Advanced User's Manual

GR716B

LEON3FT Microcontroller

User's Manual

Frontgrade Gaisler AB
Kungsgatan | SE-411 19 | Goteborg | Sweden
+46 31 7758650 | frontgrade.com/gaisler

1 Introduction

1.1 Scope

This document is the advanced user's manual for the GR716B LEON3FT microcontroller. The GR716B microcontroller has been developed in an activity initiated by the European Space Agency.

This document provides detailed information of processors, analog/digital interfaces and information related to software development. This user's manual is common for all the three package versions (CQFP, PBGA and SIP) of GR716B devices.

Separate data sheets are available for the different package versions, providing package, pin, and other details relevant to each individual package.

1.2 Datasheet limitations

Note that this document is an advanced datasheet:

- Advanced user's manual - Product in development
- Preliminary user's manual - Shipping prototype
- User's manual - Shipping space-grade product

1.3 Updates and feedback

Updates are available at https://www.gaisler.com/gr716b

Feedback: support@gaisler.com

For commercial questions please contact sales@gaisler.com

1.4 Software support

The GR716B LEON3FT microcontroller design is supported by standard toolchains provided by Frontgrade Gaisler. Toolchains can be downloaded from https://www.gaisler.com.

1.5 Development board

Information regarding the GR716B development boards are available at Frontgrade Gaisler website.

1.6 Reference documents

[AMBA]	AMBA Specification, Rev 2.0, ARM Limited
[CCSDS]	Time Code Formats, CCSDS 301.0-B-4, Blue Book, Issue 4, November 2010, https://ccsds.org
[FPGA]	GR716B Application note for FPGA scrubbing, TN-7, Issue 0.1
[GRLIB]	GRLIB IP Library User's Manual, Frontgrade Gaisler, www.frontgrade.com/gaisler
[GRIP]	GRLIB IP Core User's Manual, Frontgrade Gaisler, www.frontgrade.com/gaisler
[GRMON4]	GRMON4 User's Manual, Frontgrade Gaisler, www.frontgrade.com/gaisler
[LEON-REX]	LEON-REX Instruction Set Extension, GRLIB-TN-0001, Frontgrade Gaisler, www.frontgrade.com/gaisler
[RMAP]	Space engineering: SpaceWire - Remote memory access protocol, ECSS-E-ST-50-52C, February 2010
[SPARC]	The SPARC Architecture Manual, Version 8, SPARC International Inc.
[SPW]	Space engineering: SpaceWire - Links, nodes, routers and networks, ECSS-E-ST-50-12C
[SPWCUC]	High Accuracy Time Synchronization over SpaceWire Networks, SPWCUC-REP-0003, Version 1.1, September 2012
[SPWTDP]	Spreadsheet to configure the GRSPWTDP timer, Issue 1
[V8E]	SPARC-V8 Supplement, SPARC-V8 Embedded (V8E) Architecture Specification, SPARC-V8E, Version 1.0, SPARC International Inc.
[FTMCTRL]	FTMCTRL: BCH EDAC with multiple 8-bit wide PROM and SRAM banks, GRLIB-AN-0003, Frontgrade Gaisler, www.frontgrade.com/gaisler
[CQFP-DS]	GR716B-CQFP LEON3FT microcontroller - Data Sheet, Frontgrade Gaisler, www.frontgrade.com/gaisler
[PBGA-DS]	GR716B-PBGA LEON3FT microcontroller - Data Sheet, Frontgrade Gaisler, www.frontgrade.com/gaisler
[SIP-DS]	GR716B-SIP LEON3FT microcontroller - Data Sheet, Frontgrade Gaisler, www.frontgrade.com/gaisler
[DS]	The different CQFP, PBGA and SIP data sheets are commonly referred as [DS] when required.

1.7 Document revision history

Change record information is provided in table 1.

Table 1. Change record

Version	Date	Sections	Note
0.0	July 2020		First internal release
0.1	November 2020		Update after PDR review
0.2	March 2022		Release for DDR review
0.3	November 2022		Release for CDR review
0.4	November 2022	**figure 7**	Improved figure for motor control
		figure 117	Corrected APWG block input/states
		54.1.4	Corrected number of inputs/states
		55.4.4	Updated APWM-DAC Chapter
		Table 761	Corrected reference current
		Table 760	Added Table for ADC 13/14 Bit mode
		2.2.10	Added text for FPGA Configuration and Supervision
0.5	December 2022	34	First public release
0.6	September 2023	All	Frontgrade format
		Section 12.2.2	Added Minimum track time required for ADCs
		Section 58	Updated Section 58.1 AMR, 58.2 ROC and 58.9 Reference Voltages and Currents Electrical Characteristics
		Section 2.12	Updated Memory map
0.7	April 2024		Major updates
0.8	December 2024	Table 7	Updated IO matrix table Updated register for Brownout Updated ADC, ACOMP and DAC sections Updated RTA sections with Ticklines and general RTM descriptions Updated GRDMAC2, one DMA core available Updated Memory map Updated Interrupt table with APWM interrupts Updated Register interface for APWM blocks Added information about usable maximum baudrate information for AHBUART
0.9	March 2025	Section 53	Updated APWM_G, G*, Clock detector, Alarm protection matrix, timestamp blocks

Table 1. Change record

Version	Date	Sections	Note
0.10	November 2025	All sections	Datasheet and User's Manual split. This document is a common User manual for all three devices.
		Several Sections	Renamed GR716 to GR716B
		Section 22 Table 7 17 18	Fault Tolerant PROM/SRAM Memory Interface memory controller signals (Pin shared) are renamed from MEM_* to MEM0_*
		Table 7	Note 16 17 18 and 19 added to map the interfaces with the corresponding core functionality
		Table 7	Corrected numbering of ALARM outputs
		Table 546	Included missing register information in AHBUART- MPSTAT
		Table 671	Corrected input state reset value
		Table 7	Note 5 updated providing information about Test Enable Register
		Several sections	Removed internal memory test since functionality is not implemented
		Section 53.11	Corrected APWM Synchronization Register
		Section 17.6.6	Corrected %asr16 register bit fields
		Section 30.4.13	Corrected address offset for Logical OR
		Section 30.4.14	Corrected Logical set clear address offset and description
		Section 21.3	Corrected on-chip memory scrubber information
		Table 7	Information regarding the availability of LVDS_TX port 2, 3, 4 and 5 on individual pins on PBGA and SIP packages updated.
		Table 20	Corrected memory map for on-chip data and instruction memory for RTA's
		Table 22	Information regarding the additional bootstrap signals required for second memory controller available in PBGA/SIP devices are included
		Section 23	Information regarding the second memory controller available in PBGA/SIP devices are included
		Section 31 32	Packetwire interface removed, errata and possible workaround updated
		Section 56	FS_OUT from LVDS receivers are connected to the LVDS IO core which can trigger interrupt and status can be monitored
		Section 2.9	The modified fields have been documented to allow software to identify the Revision 1 device
		Table 844	Corrected alarm information, asynchronous clock detector alarm lines are included
		Section 52.3	Information regarding RTA interrupts updated
		Section 57.2	Erratas are listed along with identified workaround
		Table 685	Corrected timeout values programmed by Boot ROM
		Section 46.3.2	Clarified clock generation scaler values for SPI Memory Controller
		Section 2.2.18	Renamed RS232 to UART
		Section 51.7.1	Corrected information regarding BCH EDAC Protection
		Table 571	Corrected information regarding event counters for on-chip memory
		Section 49.2.3	Corrected information regarding event flags from on-chip memory
		Section 7.3.1	Added note regarding analog test bus
		Table 83	Corrected information regarding System Clock Error (SC) status bit
		Table 64	Corrected information for several bit fields
		Section 2.5	Note about APWMDAC outputs expanded.
		Section 2.6	Added usage notes about DAC_CLK as an APWMDAC input.
		Section 2.7	Added recommendations about pull-up and pull-down resistors for boot interfaces.
		Section 2.10	APWMDAC numbering changed from A-D to 0-3 in memory map.

Table 1. Change record

Version	Date	Sections	Note
		Section 12	Text of most ADC sections rewritten Block diagrams updated New block diagrams of analog architecture and control unit Register descriptions updated and registers renamed Added minimum setup to sample ADC Added information about maximum sample rate and capacitor precharge
		Section 15	Text of most sections rewritten Added block diagram of control unit Added detailed description of synchronization and ramp generation Updated register description, changed register names, documented additional registers
		Figure 16	Figure updated with correct register field, packetwire clocks removed, note added for clock with schmitt trigger input
		Section 45	Corrected information in SPI for Space Slave Controller
		Section 2.2.6	Updated information regarding the FTMCTRL1
		Sections 27 and 28	Merged into a single section. Changed APB address offsets to global AMBA addresses. Corrected APWMDAC clock enable bit descriptions.
		Section 34	Added missing information in SPWTDP
		Section 54	Sections reordered. Rewrite of overview and operation. Added new sections about I/O switch matrix and clock and tick generation. Register addresses corrected and previously missing registers added. Registers renamed and description expanded. Changed block diagram of APWMDAC internals.
		Sections 16, 28	Split FIR filter documentation into a separate chapter from ACOMP.
		Section 16	FIR moved to separate chapter. Block diagrams updated Renamed registers IPOL0/1->IEDGE0/1 New section on GPIO config Rewrote introduction Interrupt generation documented
		Section 28	FIR registers renamed, descriptions expanded and addresses corrected Added bus, filter and clocking block diagrams Added table of input selection for all FIR filters Added text for detailed operation of FIR filters and their clock generators
		Section 2.10	Renamed COMP units to ACOMP for consistency Changed description from comparators A/B to 0-11/12-19 for consistenct Removed “APWM” from FIR filter description PWM DAC now numbered 0-3 instead of A-B ADC interrupts for all sampling buffers added
		Sections 2.8.3 and 2.8.4	Added sections about the need for analog mode on analog pins and recommendations for unused pins.
		Table 94, Table 96, Table 112, Table 700, Table 701, Table 702, Table 703	Added cross-references to where sources of RTA ticks and ADC/DAC synchronization sources are documented.

Table 1. Change record

Version	Date	Sections	Note
		Section 21	Block diagram, text and memory map now includes the RTA LRAM. Added missing register fields MECNT, MS, RB, ES and AC to the LRAMCFG register description. SCTRUBCTRL.ADDR occupies bits 13:2, not 14:2. Expanded description of the SCRUBDATA, SCRUBCTRL and SCRUBCFG registers. Expanded the description of the diagnostic access and error injection via the scrubber. Scrubber subsection now has numbered subsections of its own. Previously the subsections were unnumbered.
		1.6	Non-functional links removed. Consistent website references. Added GRLIB-TN-0001 document number for [LEON-REX]
		2.5	Moved comparator connection table out of the notes and into a new table in the same section showing all analog GPIO connections. Analog signal names changed to AINA0-7, AINB0-7, and AOUT0-3. APWMDAC signals changed from DAC* to PDAC*. Reformatted table to avoid frequent line breaks. Reordered and rewrite all notes. Corrected SPI_SEL0 to SPI_SLV0_0 for SPI master with LVDS I/O.
		7.1	Expanded and rephrased description of SYS.CFG.LVDS0-1 registers. Clarified conditions under which internal pulldown and pullup are enabled.

1.8 Acronyms

Table 2. Acronyms

Acronym	Comment
AHB	Advanced High-performance bus, part of [AMBA]
AMBA	Advanced Microcontroller Bus Architecture
APB	Advanced Peripheral Bus, part of [AMBA]
BCH	Bose–Chaudhuri–Hocquenghem, class of error-correcting codes
CAN	Controller Area Network, bus standard
CPU	Central Processing Unit, used to refer to one LEON3FT processor core.
DMA	Direct Memory Access
DSU	Debug Support Unit
EDAC	Error Detection and Correction
FIFO	First-In-First-Out, refers to buffer type
FPU	Floating Point Unit
Gb	Gigabit, 10^9 bits
GB	Gigabyte, 10^9 bytes
GiB	Gibibyte, gigabinary byte, 2^{30} bytes, unit defined in IEEE 1541-200
I/O	Input/Output
ISR	Interrupt Service Routine
JTAG	Joint Test Action Group (developer of IEEE Standard 1149.1-1990)
kB	Kilobyte, 10^3 bytes
KiB	Kibibyte, 2^{10} bytes, unit defined in IEEE 1541-2002
Mb, Mbit	Megabit, 10^6 bits
MB, Mbyte	Megabyte, 10^6 bytes
MiB	Mebibyte, 2^{20} bytes, unit defined in IEEE 1541-2002
MVT	Multi Vector Trapping
PROM	Programmable Read Only Memory
RAM	Random Access Memory
RTA	Real Time Accelerator
SEE	Single Event Effects
SEL/SEU/ SET	Single Event Latchup/Upset/Transient
SPARC	Scalable Processor ARChitecture
SVT	Single Vector Trapping
SW	Software
UART	Universal Asynchronous Receiver/Transmitter

1.9 Definitions

This section and the following subsections define the typographic and naming conventions used throughout this document.

1.9.1 Bit numbering

The following conventions are used for bit numbering:

- The most significant bit (MSb) of a data type has the leftmost position
- The least significant bit of a data type has the rightmost position
- Unless otherwise indicated, the MSb of a data type has the highest bit number and the LSb the lowest bit number

1.9.2 Radix

The following conventions is used for writing numbers:

- Binary numbers are indicated by the prefix "0b", e.g. 0b1010.
- Hexadecimal numbers are indicated by the prefix "0x", e.g. 0xF00F
- Unless a radix is explicitly declared, the number should be considered a decimal.

1.9.3 Data types

Byte (BYTE)	8 bits of data
Halfword (HWORD)	16 bits of data
Word (WORD)	32 bits of data

1.10 Register descriptions

An example register, showing the register layout used throughout this document, can be seen in table 3. The values used for the reset value fields are described in table 4, and the values used for the field type fields are described in table 5. Fields that are named RESERVED, RES, or R are read-only fields. These fields can be written with zero or with the value read from the same register field.

Table 3. <Address> - <Register acronym> - <Register name>

31 24	23 16	15 8	7 0
EF3	EF2	EF1	EF0
<Reset value for EF3>	<Reset value for EF2>	<Reset value for EF1>	<Reset value for EF0>
<Field type for EF3>	<Field type for EF2>	<Field type for EF1>	<Field type for EF0>

31: 24 Example field 3 (EF3) - <Field description>
23: 16 Example field 2 (EF2) - <Field description>
15: 8 Example field 1 (EF1) - <Field description>
7: 0 Example field 0 (EF0) - <Field description>

Table 4. Reset value definitions

Value	Description
0	Reset value 0.
1	Reset value 1. Used for single-bit fields.
0xNN	Hexadecimal representation of reset value. Used for multi-bit fields.
0bNN	Binary representation of reset value. Used for multi-bit fields.
NR	Field not reset
*	Special reset condition, described in textual description of the field. Used for example when reset value is taken from a pin.
-	Don’t care / Not applicable

Table 5. Field type definitions

Value	Description
r	Read-only. Writes have no effect.
w	Write-only. Used for a writable field in a register where the field’s read-value has no meaning.
rw	Readable and writable.
rw*	Readable and writable. Special condition for write, described in textual description of field.
wc	Write-clear. Readable, and cleared when written with a 1
cas	Readable, and writable through compare-and-swap. Only applies to SpaceWire Plug-and-Play registers.

2 Architecture

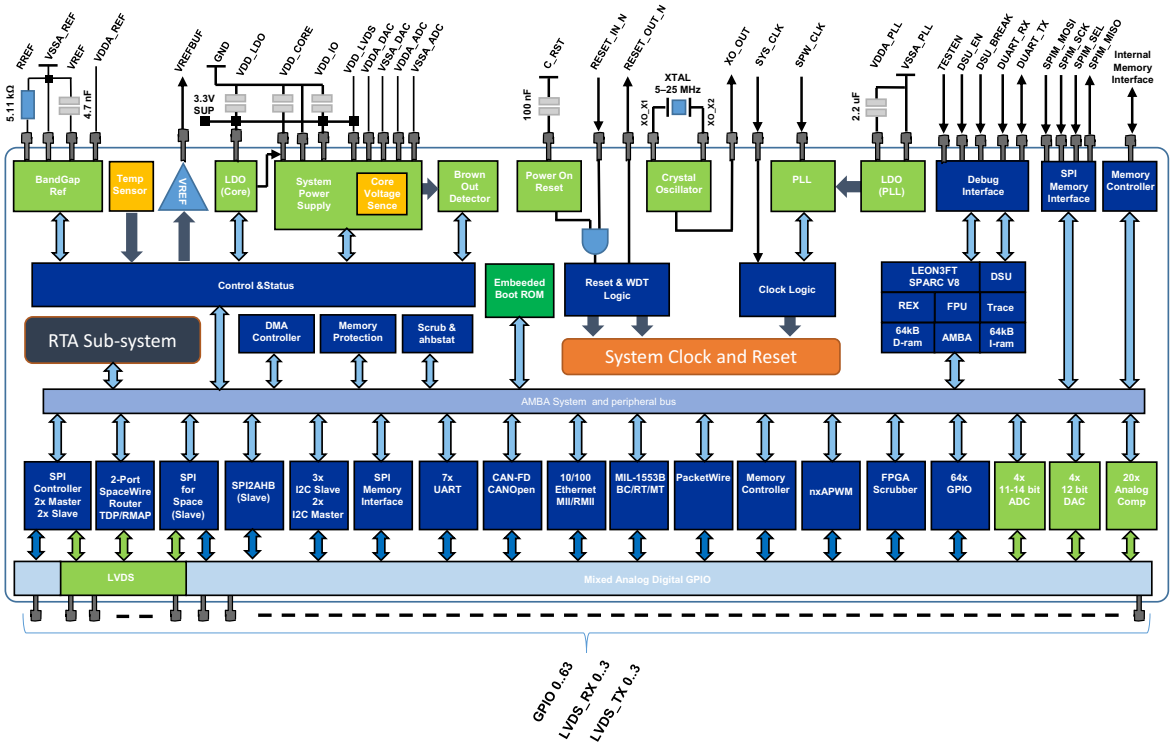


Figure 1. GR716B 132-pin block diagram

The microcontroller is a single core LEON3FT SPARC V8 processor, with advanced interface protocols, that has been optimized for real-time systems and deterministic software execution. Features such as SPARC V8E Alternate Window Pointer, interrupt zero jitter latency, SPARC V8E multiply step instructions and the possibility to run software (including interrupt handlers) from local RAM are supported to increase the determinism and responsiveness in the system. The LEON-REX instruction set extension is also supported by the microcontroller and is further described in [LEON-REX].

The architecture has multiple instances of the AMBA Advanced High-speed Bus (AHB), to which the LEON3FT processor and other high-bandwidth units are connected. Low bandwidth peripherals/functions are connected to the AMBA Advanced Peripheral Bus (APB) which is accessed through an AHB to APB bridge. The use of multiple processor buses also enables non-intrusive debugging and the possibility to have direct access to on-board memory without interrupting or involving the LEON3FT processor.

64 external CMOS pins, 5 LVDS transmitters and 4 LVDS receivers are configurable from software via configuration registers. Pre-defined pin configurations are defined in the boot software and can be enabled by using pull-up/pull-down resistors on external pins during reset. Pre-defined configuration of external pins are useful in cases when the microcontroller should boot from external memories or remote controlled via SpaceWire, CAN-FD, UART and SPI after reset. The program controlling the microcontroller needs to set appropriate direction and functionality on all pins after reset depending on the environment that the microcontroller is used in. On-chip LVDS transceivers with support for extended common-mode, cold-spare and fail-safe for SpaceWire and SPI for Space and dedicated pins for external SPI boot ROM boot are available and can optionally be used.

The microcontroller has a high level of integrated analog functions. Analog functions integrated on-chip include analog to digital converters (ADC), digital to analog converters (DAC), fast analog comparators (ACOMP), precision voltage reference (VREF), crystal oscillator (XO), phase-locked loop oscillator (PLL), brownout detection (BO), power-on and reset functionality, and supply voltage regulators (LDO) to support single 3.3V supply.

2.1 Key features

- Core
  - Fault-tolerant SPARC V8 processor with 31 register windows and support for LEON-REX.
  - Double precision IEEE-754 floating point unit.
  - Memory protection units with eight zones and individual access control of APB peripherals for memory protection.
  - Advanced on-chip debug support unit with trace buffers and statistic unit for software profiling.
  - Single cycle instructions execution and data fetch from tightly coupled memory.
  - Deterministic instruction execution time and interrupt latency.
  - Fast context switching (Partial write %PSR, AWP, Register file partitioning, interrupt mapping, MVT).
  - Single Vector Trap support.
  - Interrupt zero jitter delay.
- Memories
  - 128KiB EDAC protected tightly coupled memory with single cycle access from processor.
  - Embedded ROM with boot loader for initializing and remote access.
  - Dedicated SPI memory interface with boot ROM capability, and 4-byte addressing support
  - 8-bit SRAM/ROM (FTMCTRL) with support up to 16 MiB ROM and 32 MiB SRAM.
  - Support for system in package with embedded SRAM/PROM (FTMCTRL).
  - Scrubber with programmable scrub rate for all embedded memories and external PROM/SRAM and SPI memories.
- System
  - On-chip voltage regulators for single supply support. Capability to sense core voltage for trimming of the embedded voltage regulator for low power applications.
  - Power-on-reset, brownout detection and dual watchdogs for safe operation. External reset signal generation for reseting companion chips.
  - Crystal oscillator support.
  - PLL for System and SpaceWire clock generation. In-application programming of system clock and peripheral clocks. System and SpaceWire clocks switches glitch free.
  - Low power mode and individual clock gating of functions and peripherals.
  - Temperature, precision voltage reference and core voltage sensor.
  - Precision voltage reference output for analog high-precision bias and measurement.
  - Programmable real-time accelerators for real-time execution capability
  - Enhanced programmable DMA controllers with support if-else statements. DMA transfers can be triggered on events such as interrupts or bits/register changing value.
  - Timer units with seven 32-bit timers including watchdog.
  - Multiple bus structures for non-intrusive debug, DMA transfers and memory scrubbers.
  - Atomic access support for GPIO control registers (AND, OR, XOR, Set&Clear).
  - Support for NVRAM (SRAM and/or PROM) embedded in package. Support for software boot and execution from embedded RAM for future package options.

- Peripheral access control.
- Embedded trace and statistics unit for profiling of the system.

• Peripherals
- SpaceWire Router with support for RMAP and Time Distribution Protocol.
- Redundant MIL-STD-1553B BRM (BC/RT/BM) interface.
- CAN-FD bus controller with CANOpen support for remote boot.
- Six UART ports, with 16-byte FIFO.
- Two SPI master/slave serial ports.
- SPI in Space hardware support for SPI Slave protocol 0,1 and 2
- Two I2C master/slave serial ports.
- 10/100 Ethernet interface
- Up to 64 general purpose input and outputs (GPIO) with external interrupt capability, pulse generation and sampling.
- Four single ended Digital to Analog Converters (DAC), 12-bit, 3 MS/s, digital ramp generation support up to 25 MS/s.
- Four Analog to Digital Converters (ADC), 11/14-bit, 500/80 kS/s, differential pre-amplifier, 4 differential or 8 single ended MUX inputs per ADC, 16 analog GPIO inputs in total, and digital support for oversampling. Independent/simultaneous sampling on the four ADCs supported.
- Hardware FPGA programmer and scrubber
- Latch-up detection, to support e.g. space usage of commercial FPGAs and other COTS
- Analog PWM controller for switching power converters and motor control
- Four units for pulse-width-modulation ΔΣ DAC (APWMDAC) with 24-bit input and PWM clock up to 25 MHz.
- System clock supervision

• I/O
- Configurable I/O selection matrix with support for mixed signals, internal pull-up/pull-down resistors, schmitt-trigger input.
- LVDS transceivers with extended common-mode, cold-spare and fail-safe support for SpaceWire or SPI.
- 20 Programmable analogue comparators, pin-shared with ADC inputs and DAC outputs.
- Dedicated SPI boot ROM support for configuration.

• Supply
- Single 3.3V supply or separate Core Voltage 1.8V, and I/O voltage 3.3V.

• Radiation tolerance
- Technology: 180 nm process, UMC Taiwan
- Library: DARE+ Library version 5.7, IMEC
- SEU: Proven tolerance with hardened flip-flops and error corrections on all on-chip and external memories
- CQFP-132 and PBGA-400
- Total Ionizing Dose (TID) guaranteed up to 100 krad (Si).
- Single Event Latch-up (SEL) immune (LETth > 118 MeV*cm^2/mg).

  - SIP-400
  - Total Ionizing Dose (TID) guaranteed up to 100 krad (Si) (TBC).
  - Single Event Latch-up (SEL) immune (LETth > 70 MeV*cm^2/mg) (TBC).
- Package
  - 132-lead CQFP, 0.635 mm pitch, 25mm x 25mm, hermetically sealed with flat pins and insulating lead-frame for customer trim and form.
  - 400-pin PBGA package with lead-free balls (SAC305 material), 21mm x 21mm.
- Boot ROM and boot options
  - Remote boot directly via SpaceWire, CANopen, UART, SPI or I2C.
  - Direct software execution from on-chip RAM, external SRAM, PROM or SPI memory.
  - Direct software execution from in package embedded memory.
  - Application Software Container (ASW) for boot software integrity check.
  - Boot via ASW from external SRAM, PROM or SPI memory.
  - Boot from redundant memory.
  - Fast boot option.
- System configuration
  - Reset and boot status.
  - Individual reset and clock control for digital and analog peripherals.
  - Remote reset and boot control.
  - Clock source and divide control for the core system, SpaceWire, ADC, DAC, APWM-DAC, FIR, MIL-1553 and FPGA scrubber clock domain.
  - Support for external system reset.
  - Support for external clock source for the system, SpaceWire, Ethernet, SPI4SPACE and MIL-STD-1553.
  - Oscillator shutdown if oscillator not used.
  - Individual programmable brown-out levels.
  - Programmable LDO output level for low power mode.

2.2 Digital Architecture Overview

The system is built around three 32-bit AMBA AHB buses; the 32-bit Main AHB bus, the 32-bit DMA AHB buses and the 32-bit Debug AHB bus. The main bus connects the LEON3FT core with all other peripheral cores in the design as well as the external memory controllers. Several peripherals are connected through AMBA AHB/APB bridges where one of the bridges is integrated with the DMA controller.

The Debug AMBA AHB bus connects a UART serial debug communications link to the debug support unit and also to the rest of the system through an AMBA AHB bridge. A simplified block diagram is provided below, detailed memory map and AMBA AHB/APB bridge information are available in section 2.10.

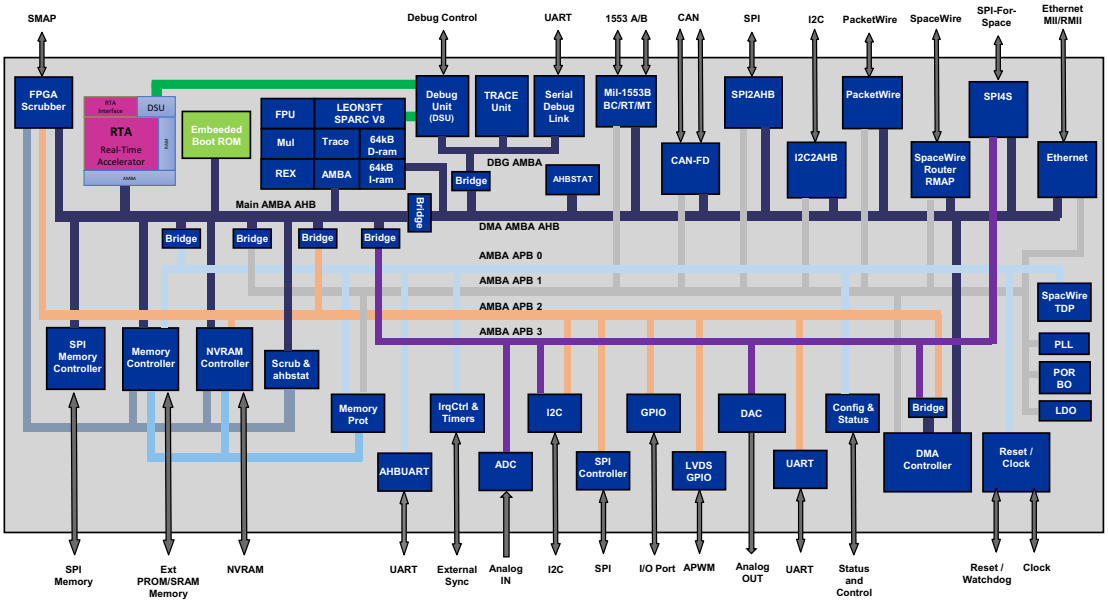


Figure 2. Simplified architecture and functional block diagram of the microcontroller (TBC)

2.2.1 Processor core and memory subsystem

The microcontroller implements a LEON3FT 32-bit processor core conforming to the IEEE-1754 (SPARC V8) architecture. The microcontroller is designed for embedded applications, combining high performance with low complexity and low power consumption. The LEON3FT core has the following main features: 7-stage pipeline with Harvard architecture, hardware multiplier and divider and on-chip debug support. The LEON3FT processor is enhanced with fault tolerance against SEU errors. The fault tolerance is focused on the protection of the on-chip RAM, processor register file and protection of external memory interfaces.

The LEON3FT integer pipeline is implemented with 31 register windows, SEU protection of register file with zero impact on software timing, and hardware multiply and divide units. The multiplier is a 32x32 hardware multiplier that performs operations in a single cycle. Floating-point operations are supported by a hardware floating-point unit.

Memory protection units are located on the AMBA system bus and on AMBA DMA bus. Each protection unit monitors access on the AHB bus. When an access is made to a protected area then the protection unit will assert a signal to the memory controller that will annul the operation and respond to the AMBA access with an AMBA ERROR response. Four areas can be protected on the system bus and four areas can be protected on the DMA bus.

Exclusive write permission can be enforced for individual APB peripherals to protect interfaces from erroneous writes.

To protect tightly coupled instruction and data memory directly connected to the processor core from software the LEON3FT hardware watchpoints (located within the processor integer unit) can function as memory protection registers for both the instruction and data RAM.

2.2.2 Deterministic Operation

Several features are supported in the architecture in order to enhance it for embedded microcontroller applications:

- Support for SPARC V8E write partial %psr
- Support for SPARC V8E Alternative Window Pointer
- Support of the SPARC V8E Multiply step instructions

The microcontroller program execution is deterministic due to the microcontroller being cache-less, and AMBA accesses made by the processor being unaffected by other AMBA masters in the microcontroller. The processor uses separate EDAC protected instruction and data memories with fixed latencies. The instruction memory latency is 1 system clock and the delay for the data memory is 1 system clock. The local instruction and data memory in the system have the same latency and behaviour in the corrected as in the uncorrected case. This also applies to the CPU, so dynamic SEU handling schemes such as the LEON3FT pipeline restart on error options is not be used.

The microcontroller has 64 KiB of shared data RAM and 64 KiB of tightly coupled instruction memory connected to the processor. The tightly coupled instruction and data RAM can be accessed via the AMBA buses. This AMBA access can be used to upload new software into the instruction memory or read/write data to/from any AMBA master in the system. The access to the data memory will not affect or delay any access made by the processor on the AMBA bus. 64KiB of the instruction/data memory are available for sharing with the real-time accelerators.

The processor or any AMBA master can access the external PROM/SRAM or SPI memory controller for program execution or reading/writing data. The external SRAM memory can be protected by the scrubber located on the main system bus. The scrubber connected to the main system bus will block access for the processor to the external memories during scrub execution. The scrub rate can be configured and should be set to an acceptable rate for the mission. The scrubber access will not block the AMBA bus since masters and slaves on the main system bus support split transactions.

2.2.3 DMA controller

The microcontroller has one DMA controller. The GRDMAC2 core provides a flexible direct memory access controller. The DMA controller can perform burst transfers of data between AHB and APB peripherals at aligned or unaligned memory addresses. The GRDMAC2 core has multiple AHB master interfaces for access to AHB peripheral bus and direct access to all APB slaves. The GRDMAC is able to perform programmable sequences of data transfers between slaves in AMBA address space. The controller is able to transfer data between peripherals and memory and between memory areas. If the accessed memory is internal or external does not matter, as long as the memory is mapped into AMBA address space reachable from the AHB bus where the DMA controller is mapped.

The DMA controller configuration registers are accessible through an APB interface. Each DMA controller can be flexibly configured by means of two descriptor chains residing in main memory: a Memory to Buffer (M2B) chain and a Buffer to Memory (B2M) chain. Each chain is composed of a linked list of descriptors, where each descriptor specifies an AHB address and the size of the data to read/write, supporting a scatter/gather behavior.

Once enabled, the DMA controller will proceed in reading the descriptor chains, then reading memory mapped addresses specified by the M2B chain and filling its internal buffer. It will then write the content of the buffer back to memory-mapped addresses by elaborating the B2M descriptor chain.

The DMA controller supports a simplified mode of operation. In this mode of operation only one descriptor is present for each of the M2B and B2M chains. These two descriptors are written directly in the core's register via APB.

The DMA controller will offload the CPU and provide DMA capabilities to peripherals and communication interfaces in the microcontroller design that do not have an internal DMA engine. The DMA controller can be programmed to initiate DMA transfers on events, such as interrupts, to the GRDMAC controller to achieve timely readouts of values. An example of use can be found the detailed description of the DMA controller in section 29.

2.2.4 Interrupt handling

The microcontroller supports interrupt time stamping and interrupt handling mechanism to ensure that a fixed number of clock cycles occurs between the assertion of an interrupt and the processor's jump to the trap table. Depending on the software application, several types of time stamping can be of interest:

- Timestamp when interrupt line is raised from peripheral IP core. This time is of particular importance when time needs to be synchronized with an external event.
- Timestamp when processor acknowledges the interrupt. This stamp is primarily of interest in system characterization where users may want to measure the time it takes for the processor to divert execution flow to the interrupt service routine after the processor has discovered the pending interrupt.
- Timestamp when software enters ISR. This timestamp is typically taken by software by reading a timer register when the ISR is entered.

Interrupt time stamping is controlled via the Interrupt Timestamp Control register(s) described in section 40. Each Interrupt Timestamp Control register contains a field (TSTAMP) that contains the number of timestamp register sets that the core implements. A timestamp register sets consist of one Interrupt Timestamp Counter register, one Interrupt Timestamp Control register, one Interrupt Assertion Timestamp register and one Interrupt Acknowledge Timestamp register.

Software enables time stamping for a specific interrupt via an Interrupt Timestamp Control Register. When the selected interrupt line is asserted, software will save the current value of the interrupt timestamp counter into the Interrupt Assertion Timestamp register. When the processor acknowledges the interrupt, the Interrupt Timestamp Control register will be set and the current value of the timestamp counter will be saved in the Interrupt Acknowledge Timestamp Register. The difference between the

Interrupt Assertion timestamp and the Interrupt Acknowledge timestamp is the number of system clock cycles that was required for the processor to react to the interrupt and divert execution to the trap handler.

2.2.5 Reset and software boot

The reset default behavior for all included cores, except the LEON3FT processor, is to enter an idle state upon reset. The internal reset signal will be asserted as a result of power-on. In the idle state the cores do not initiate any transactions and does not keep any signals in output mode. This is of particular concern for bidirectional signals to prevent contention.

The LEON3FT processor will normally start executing from a predefined start address 0x0000000 at reset. The start of execution can be prevented by assertion of an external break signal. If the break signal is asserted then the processor will enter power-down mode after reset. This will allow software upload from an external entity that can then start the processor at a dynamically specified address, by writing to the interrupt controller's register interface. Processor can optionally be forced via bootstraps to be forced to start from external PROM, SRAM, MRAM, or SPI memory. This mode could be used if the application requires separate boot code than the one existing in the LEON3FT microcontroller boot ROM. Boot addresses for external PROM and SPI memory are defined in section 2.10.

A boot ROM application is placed at address 0x00000000 and is normally executed after reset. The boot application supports system functions controllability via external bootstrap registers. The application always starts executing after reset and checking the value of external bootstrap signals. Based on these signals the processor performs tasks such as load software to internal RAM from an external memory device, enable remote access via SpaceWire, CAN-FD, SPI, UART or I2C. See section 3.1 for more information about bootstrap options for the boot ROM.

A protocol to guard against the system trying to boot using a corrupt boot image is implemented using a protected image format containing an image header, boot code, data checksum and header checksum, see section 51. Extra protection can be enabled via bootstraps by reading identical images from redundant memories but needs to be configured before booting via an external boot strap.

Self-test and diagnostic test of the CPU and internal RAMs can be enabled via bootstraps. The internal ROM will check for Stuck-At and Transition errors in local instruction and data ram. Stuck-At or Transition error(s) will result an error reported in the boot report, see 51.2.5.

2.2.6 Direct boot from external memory

Custom boot options are supported via bootstrap options to bypass the internal boot ROM code. The LEON3FT microcontroller can be configured to boot directly from external ROM, external SRAM, external SPI Memory or internal NVRAM in SIP package.

2.2.7 Remote access and control

The microcontroller can be accessed and controlled by an external control unit via SpaceWire (RMAP), CAN FD (CANOpen), UART, SPI or I2C without using processor support. Full access, except for debug features on the debug AMBA bus, will be granted to SpaceWire, CAN-FD, UART, SPI or I2C if enabled at startup via bootstraps after reset:

- SpaceWire: Remote Memory Access Protocol (RMAP) provides full remote access to the entire AMBA address space of the microcontroller. See section 33 for more information
- CAN-FD: Support for remote access via CANOpen PDOs provides full remote access to the entire AMBA address space of the microcontroller. See section for GRCANFD for more information
- UART: Support for reading and writing to register via special protocol over UART provides full remote access to the entire AMBA address space of the microcontroller. See section for AHBUART for more information

- SPI: Support for reading and writing to register via special protocol over SPI provides full remote access to the entire AMBA address space of the microcontroller. See section for SPI2AHB for more information
- I2C: Support for reading and writing to register via special protocol over I2C provides full remote access to the entire AMBA address space of the microcontroller. See section for I2C2AHB for more information

All the communication interfaces above can be implemented to be functional directly after the microcontroller leaves reset, no initialisation from the processor is required. The communication links can also be disabled by the processor, a feature that can be required for safety.

When debugging the microcontroller, the DSU is used to load software and initiate the program counter. In the case when new software is remotely updated via SpaceWire, CAN-FD, UART, SPI or I2C, a special feature in the interrupt handler is implemented to restart the system and to start execution of new software. For more information see section 40.2.7 to 40.2.9.

2.2.8 Pin sharing

An I/O switch matrix allows most of the GR716B microcontroller pins to be shared between several peripherals. The I/O switch matrix provides a flexible solution where each pin can be individually configured to one of up to 16 different functions.

The microcontroller contains on-chip ADC, DAC and analog comparator (ACOMP) peripherals. They require special mixed digital and analog I/Os. The mixed digital and analog I/O are controlled via configuration registers and need to be set to analog mode when an ADC, DAC or analog comparator (ACOMP) is going to be used. See sections 2.8.3, 12.3.1, 15.4.1, and 16.2.1.

SPI4SPACE supports on-chip LVDS transceivers and CMOS I/Os. The redundant SPI4SPACE channel can be accessed via CMOS pins and the primary SPI4SPACE channel is accessed via on-chip LVDS.

2.2.9 Integrated ADC and DAC

The ADC digital control logic supports functions to control the on-chip ADC and to offload the processor. Support for automatic oversampling on all channels, sample sequencer and digital level comparators are examples of features integrated to offload the processor. The integrated DMA controller can also be used to off-load the processor by automatic transfers of sample values to/from the integrated data memory and ADC.

The DAC digital control logic supports functions to synchronize DAC output to events and generate programmable output waveforms. The main purpose of the programmable waveform feature is to support DC/DC applications.

2.2.10 FPGA Configuration and Supervision

The FPGA configuration and supervisor in GR716B can program an external FPGA, and scrub its configuration memory to prevent accumulation of errors in the configuration memory. The controller is compatible with the Kintex UltraScale and Virtex-5 Xilinx FPGA families.

The controller can be set to scrub the entire FPGA configuration memory or just a defined memory area. It accesses the FPGA configuration memory externally through the SelectMap (SMAP) interface, which provides better performance in comparison with JTAG, due to the parallel data access. The controller accesses through an AMBA bus a Golden memory that can be available in external SPI FLASH (accessed via SPI Memory Controller). The original configuration bitstream is stored in the Golden memory, and it is used both to configure the FPGA at start-up and to repair the FPGA configuration memory in case of errors. The Golden memory also stores the mask data and the Cyclic Redundancy Check (CRC) codes used to check the configuration bitstream integrity.

For more information see reference document [FPGA] and section 55.

2.2.11 Real-time accelerator and system overview

The two real-time accelerators can handle stringent cycle-by-cycle real-time applications which is useful in, e.g., switching power and motor control applications. The real-time accelerators (RTAs) provides user flexibility to support a wide variety of real-time critical applications addition, general hardware – directly accessible from the RTAs – is implemented to execute all real-time functions that are time critical downto system-clock cycle level, such as PWMs, ADC, DAC, UART, I2C, SPI and GPIOs

Each Real-Time Accelerator (RTA) implements a LEON3FT 32-bit processor configured with minimum number of register windows, which is complemented by a RTA Task Manager (RTM) and a RTA time Stamp Manager (RSM). The RTA Task Manager (RTM) enables time critical applications which interrupt/event depended to execute code in the RTAs LEON3FT processor in one execution level", never using interrupts. The RTA time Stamp Manager (RSM) capture consecutive events and automatically calculates the difference in time between events. Examples of real-time applications targeted by the Real-Time Accelerator (RTA):

- Multiple switching power converters such as the buck and boost topology
- Complex switching power converters such as full-bridge topologies
- Motor and magnetorquer control
- High voltage and high current generators
- Power system monitoring
- Latch-up detection, to support e.g. space usage of commercial FPGAs and other COTS

The LEON3FT in the RTA can still be programmed to use the standard software execution flow and interrupts to work in systems with lower or no real-time requirements. Examples of applications with lower or no real-time requirements targeted by the Real-Time Accelerator (RTA):

- Simple Motor control
- Latch-up detection, to support e.g. space usage of commercial FPGAs and other COTS
- Advanced DMA transfers
- Interface emulation such as UART or I2C via direct access to GPIO control registers from the RTA

The real-time system is built around two data buses and an event bus. The real-time system is separated from the AMBA system to guarantee fast access by the RTA applications executed in the Real-Time Accelerator (RTA), is presented in Figure 3. The real-time events generated in any function or interface connected are distributed on the RTA Event Bus across the chip. Each Event will be configurable and distributed to all necessary real-time functions in any real-time blocks where it may be needed to provide synchronization on clock cycle-by-cycle level. Real-time data and time-stamps are distributed on the RTA data bus. The RTA data bus has no defined protocol and it is up to the application to configure the transmitter function/interface and to interpret the information received.

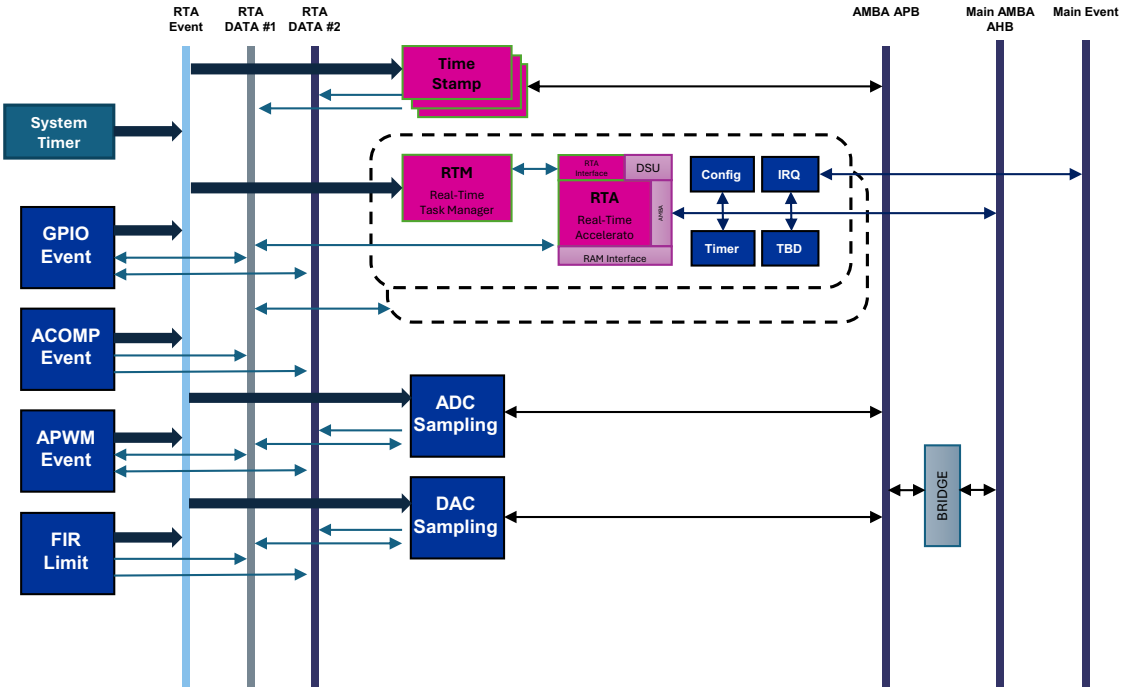


Figure 3. Simplified architecture and functional block diagram of the real-time system

Functions and interfaces connected to the real-time system:

- Analog comparators including FIR filters
- Analog to Digital Converter
- APWM blocks (PWM for Analog applications, including APWMDAC)
- Digital to Analog Converters
- UART/I2C/SPI/GPIO

The RTA processor will be kept in reset with the clock disabled after reset. The RTA needs to be reset and enabled from a user application on the main processor or via remote access before software can be executed on the RTA. The RTA has a local interrupt controller with a special feature to restart the real-time system and to start execution of new software on the RTA processor.

When debugging the microcontroller, each individual RTA's debug support unit (DSU) can be used to load software and initiate the program counter. For more information see section 40.2.7 to 40.2.9. Each RTA has a separate DSU unit.

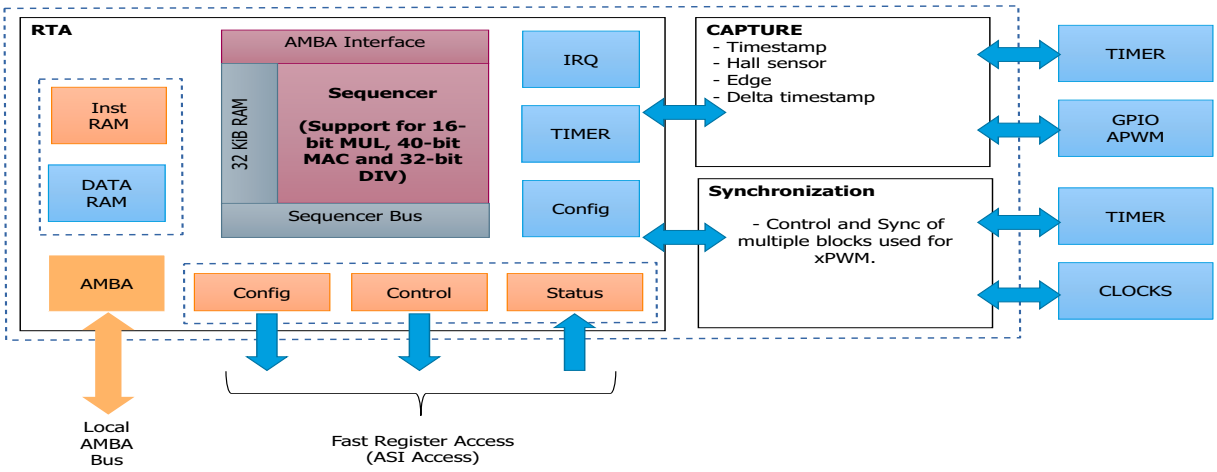


Figure 4. Simplified architecture and functional block diagram of the RTA and connections

Each Real-Time-Accelerator (RTA) includes:

- LEON3FT processor core with 2 register windows and support for 32*32 to 64 integer multiplication (UMUL/SMUL SPARC v8 instructions) and 16*16 to 40-bit integer multiply accumulate instructions (MAC SPARC v8e instruction). FPU is not available in RTA.
- Separate Data (16 KiB) and instruction (16 KiB) EDAC-protected memory for single cycle execution of instructions.
- Local System timer (GPTIMER)
- Local IRQMP interrupt controller with start/stop/reset control
- Separate Debug Unit with instruction trace
- RTA Task Manager (RTM) and interrupt time stamp functionality
- The RTA has access to some parts of the AMBA bus through bridges and some other parts cannot be accessed. The section 52.1 describes where all RTA has the access.

Each RTA module is an independent LEON3FT subsystem and can execute software in parallel with the main LEON3FT processor in GR716B. The RTA subsystem can access 32 KiB of tightly coupled memory protected by EDAC to ensure single cycle instruction execution. The RTA implements 2 sets of register window and supports interrupts.

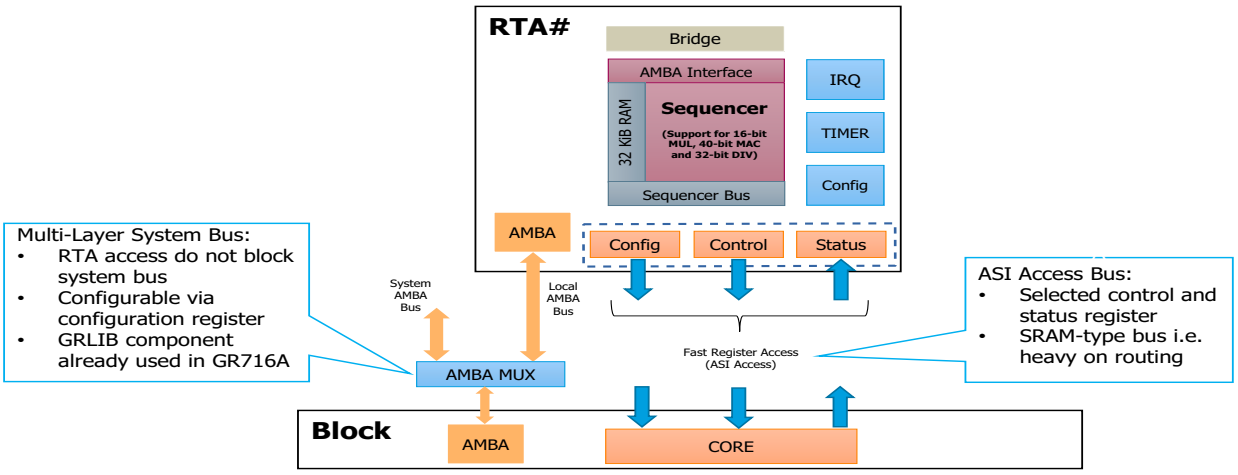


Figure 5. Real-Time-Accelerator (RTA) provides fast access to peripherals via processor ASI interface. Real-Time-Accelerator (RTA) can also access peripherals via system bus if granted by the system.

Fast access to peripherals from the RTA / LEON3FT IP CORE is performed via parallel AMBA bus structure. The benefit from using parallel bus structure is that the each RTA can make independent and parallel accesses to peripherals in the system without having to wait for access.

Another benefit from using shared or parallel buses is that all processors uses the same register address space, and all processors can use the same programming model already supported by BCC version 2.2.0 or later.

2.2.12 Switching power applications

The overall architecture real-time execution capability for DC/DC applications is secured by tightly integration between the Real-Time-Accelerator (RTA), see 2.2.11, and the hardware functionality described in this chapter 2.2.12. It supports at least 4 independent DC/DC converters running at power switch frequency up to 500kHz. Peak-current control mode is supported by on-chip analog comparator, and the voltage regulation is controlled by user-specified software in RTA.

As great user flexibility as possible is supported for a wide variety of power-supply applications, which is fulfilled by RTAs user-specified software. And generally configurable PWM hardware, directly accessible from the RTAs, is implemented to execute all real-time functions that are time critical downto system-clock cycle-by-cycle level. Further details of these implementations are presented in section 53.

2.2.13 Motor control applications

In motor control applications, hardware support for complex switching power converts can be combined with the ability to configure ADC measurements on 3 analog channels with simultaneous sampling time point for optimum motor regulation.

Architecture support for motor applications:

- At least 4 BLDC/MSM motors in PWM mode is supported
- Up to 4 analog channels with simultaneous sampling time point for optimum motor control
- Resolver interface, digital generation of a sine-wave
- Precision reference voltage for high precision measurements

Further details of motor control i presented in section 53.

2.2.14 External synchronization and alarm generation

GR716B can generate an external synchronization signal to other devices, or synchronize internal logic and counter to a received synchronization input. The external synchronization generation capabilities enables the user to synchronize multiple GR716B devices in switching-power and motor-control applications, etc.

For more information see section 53.6 and 53.7.

2.2.15 Fast analogue comparators and Latch-up detection support

The GR716B has 20 analog comparators connected to the 16 ADC input pins and the 4 DAC output pins. The trigger level of each comparator can be individually configured to one of 7 internally generated voltage, or be taken from a GPIO pin. Latches at the comparator output can catch short-duration analog trigger events, see figure 6. For more information about comparators, see section 16.

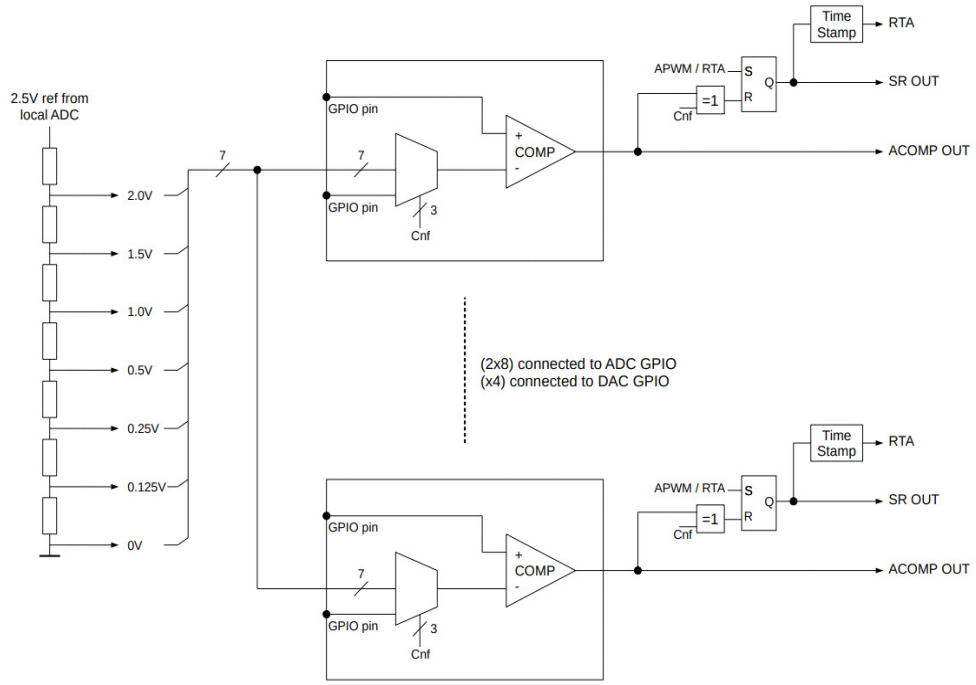


Figure 6. Functional schematic of ACOMP, which can be used for catching analog events

The analogue comparator outputs may be used as input for the following functions:

- PWM peak control in switching power applications
- General FIR filter for latch-up detection applications

The general FIR filter with binary taps has application areas such as:

- Flexible programmable latch-up detectors
- Basic FIR filter functions
- Flexible patterns recognition in communications, sensors, etc
- Continuous background-monitoring with alarm trig on wave patterns, sensors, etc.

For more information see section 16.

2.2.16 Digital Modulator for Analog Precision DAC Outputs (APWMDAC)

The APWMDAC block provides a digitally modulated output signal, to be low-pass (LP) filtered on PCB. It gives an analog precision signal after the LP-filter that has higher resolution than the 12-bit current-output DAC presented in sections 15 and data sheet section 2.7 [DS]. The maximum bandwidth will be lower, e.g. in the range 0.1 to 100 kHz, depending on priority of bandwidth, analog ripple, filter complexity, etc, in the design of the PCB LP-filter. The modulator can be configured as a first or second order modulator, see figure 7, and requires a second or third order LP-filter, respectively.

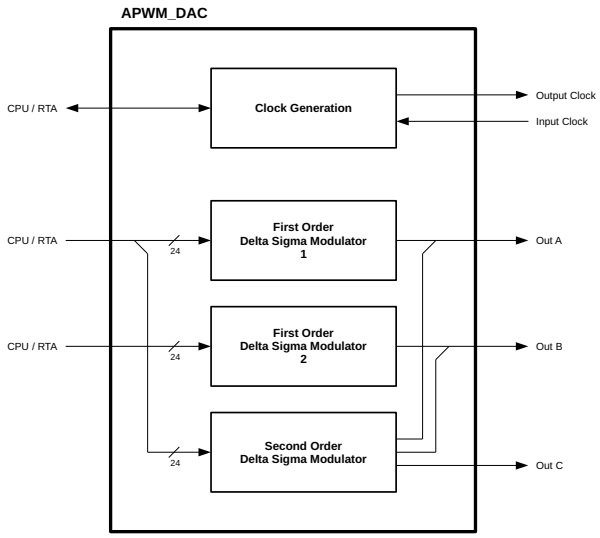


Figure 7. Functional diagram of APWM_DAC

In the simplest form, the LP-filter can be two cascaded RC-links directly connected to the GPIO pin on GR716B, which means that VDD_IO acts as analog reference voltage. If better accuracy is required, a CMOS buffer can be added on PCB, with VDD connected to a precision reference voltage.

The GR716B on-chip implementation supports full 24-bit DAC resolution, meaning that it is the PCB analog implementation that sets the analog performance limits. E.g., re-clocking the GR716B modulator output in a clocked CMOS register on PCB, with VDD connected to a precision reference voltage, will further improve the analog performance. And to implement the CMOS registers with differential signal path, going into an op-amp based differential LP-filter, will provide excellent performance.

In general, the better analog performance that is required, the more complexity will be needed in the PCB filter implementation. See section 54 for a more detailed presentation of APWMDAC with application examples of PCB filters.

2.2.17 System clock fault detection support

To detect failure on the system clock, which can affect control applications a system clock supervisor has been built-in to monitor and detect loss of system clock. The detector can be optional configured to alert the main processor or to disable regulation loops. This detector uses independent on-chip clock generator (ring-oscillator) as frequency reference.

The upper and lower frequency limits, to be checked by the Clock Detector, are configurable by user. This may be useful in cases where the frequency needs to be checked to be within a more narrow span than the temperature drift of the on-chip ring-oscillator. In such cases, the user monitors the temperature, and continuously configures the limits accordingly. Alternatively, in cases where a larger detec-

tion frequency span can be allowed, these limits can be set wide enough to allow for the whole temperature drift.

Finally, health status of the on-chip ring-oscillator can also be monitored. It is done by polling the internal status of the Clock Detector block.

For more information see sections 4 and 53.6.2.

2.2.18 Debug and statistics

An external debug host can access the microcontroller Debug Support Unit (DSU) via UART (Debug UART). The DSU can be used to access instruction trace buffers and registers of the LEON3FT processor. The DSU has also support for tracing AHB accesses that can be used for performance monitoring. For more information about the functionality see section 20. Since the DSU is connected to an AMBA AHB bus and is accessed via debug communication links also connected to AMBA AHB, all debug accesses will generate traffic over AMBA AHB. In order for the debugging to be completely non-intrusive this debug traffic is separated from the non-debug AHB traffic.

The microcontroller includes a LEON3 statistics unit that allows the debugger to count a wide range of events without interrupting or controlling execution. See section 41 for more information about the LEON3 statistics unit.

The GR716B microcontroller have one dedicated Serial Debug interface. The Serial Debug unit is directly connected to the AMBA debug bus. The Serial Debug unit have a unique AMBA address described in chapter 2.10.

The debug interface is intended to be used during software development and have direct access to the internal state of the processor and trace buffers. This interface can be disabled during mission via external pin configuration i.e. tie DSU_EN to low.

The Serial Debug interface unit is fully described in section 48

2.2.19 AMBA Error detection

The microcontroller includes status registers to store information about AMBA AHB accesses triggering an error response on the Main and DMA AMBA bus. Error response on the AMBA main bus is stored in either the memory scrubber unit or AHB Status unit 2. Error response triggered on the DMA bus is stored in the AHB Status unit 1.

The Main AMBA bus can be configured to fetch all AMBA error responses in the memory scrubber, see chapter 7.3.3. The system default configuration is to only fetch AMBA errors from the external memory controllers in the memory scrubber. All other AMBA error responses on the Main bus will be fetched in the AHB Status unit 2.

2.2.20 RTA Debug and AMBA Error detection

The external debug link a debug host can access and debug the RTA unit via separate Debug Support Unit (DSU). The RTAs Debug Support Unit (DSU) has limited instruction trace and support hardware break points.

The RTA Debug Support Unit (DSU) is connected to the internal AHB debug bus.

Each RTA also includes a control unit to monitor and detects AMBA Errors.

2.2.21 Internal communication

Triggers, events and synchronization signals that require immediate response are distributed outside the internal AMBA bus structure. This section explains the different connections next to the internal AMBA structure.

Signal connections are visually shown in figure 8 and described in table 6 in this section.

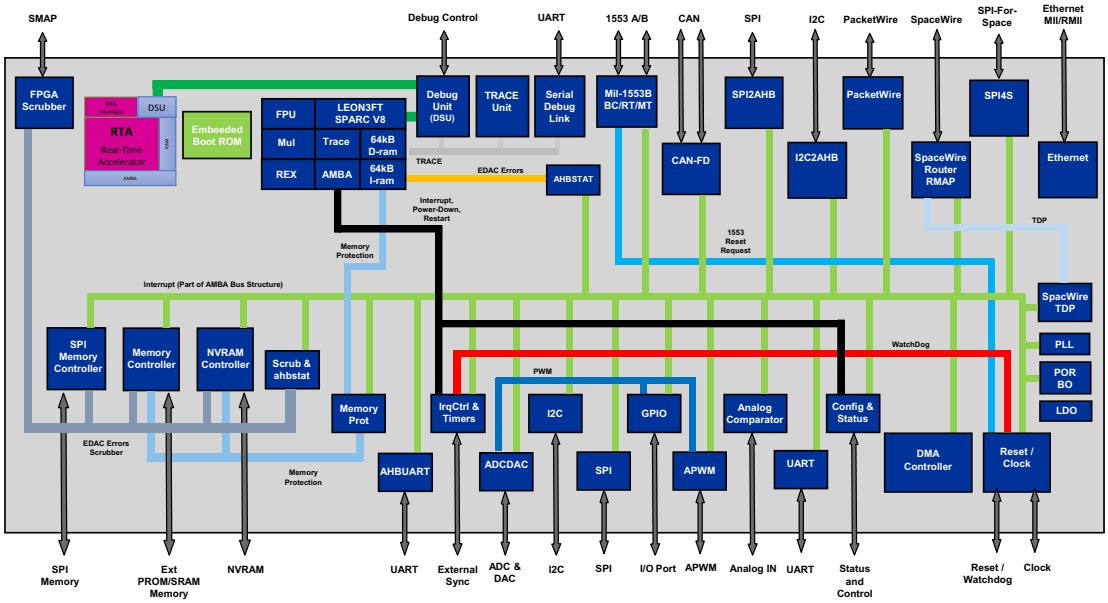


Figure 8. Internal communication paths outside AMBA bus structure (TBC)

Table 6. Internal communication paths outside the AMBA bus structure

Internal bus name	Connecting functional blocks	Description
EDAC Error	AMBA status, local instruction memory and local data memory	Connection for monitoring of correctable errors signaled from the internal data and instruction memory.
EDAC Error Scrubber	AMBA status functionality in scrubber, external memory controller and NVRAM controller	Connection for monitoring of correctable errors signaled from the memory controller and NVRAM controller.
Interrupt Bus	All blocks connected to the internal AMBA structure	Connection for distributing events from/to all peripherals and digital functionality. The internal interrupt bus distributes all 64 unique interrupts IDs in table 21. The interrupt bus is used to program event driven functions e.g. the DMA channel 0 to respond to a specific Interrupt ID in table 21.
Memory protection	Protection unit, external memory controller, NVRAM controller, local instruction memory and local data memory	Connection for blocking write access to protected areas. Protection unit grants or denies the ongoing AMBA access via the memory protection bus.

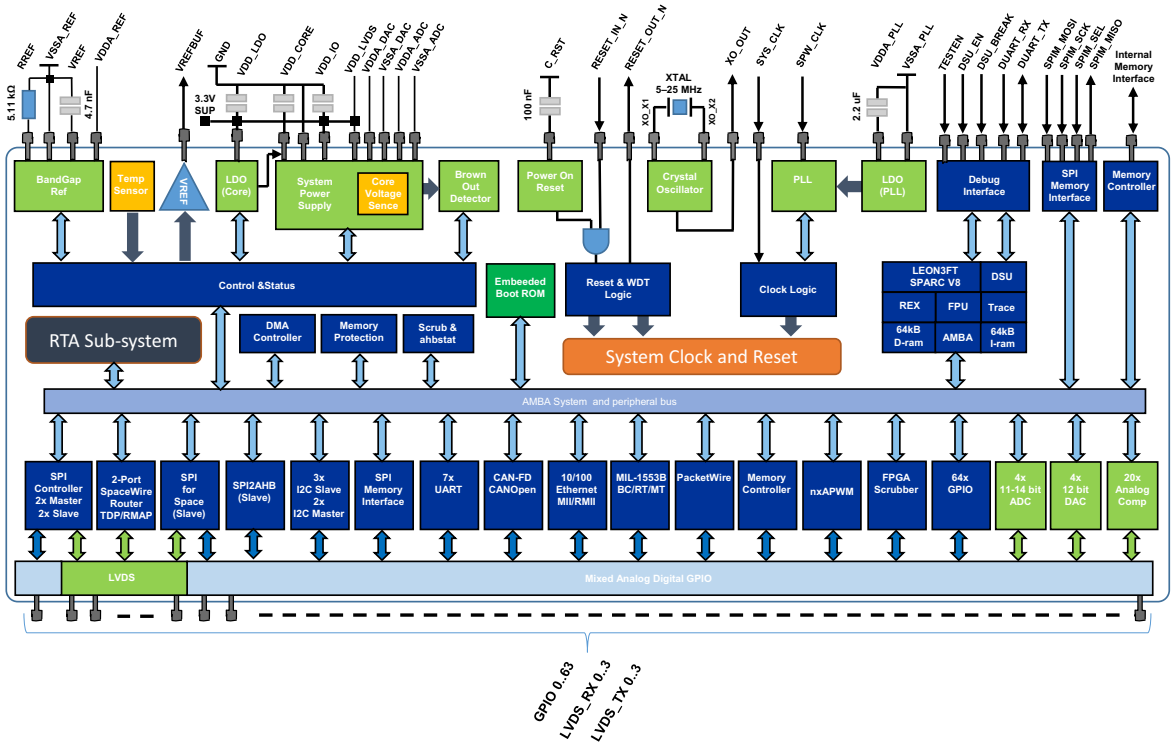
Table 6. Internal communication paths outside the AMBA bus structure

Internal bus name	Connecting functional blocks	Description
Processor Interrupt, Power Down and Restart	LEON3FT, Interrupt controller and Primary Clock gating unit.	The interrupts generated on the interrupt bus are all forwarded to the interrupt controller. The interrupt controller prioritizes, masks and propagates the interrupt with the highest priority to the processor. This bus is also used for request for Power-Down of the processor and restart of the processor. Power down request from the processor is described in section 17.2.16 and reboot is described in section 40.2.7.
Watch Dog	Timer unit 0 and reset request logic	Watch dog timer unit drives a watchdog signal on this bus to request restart of the system. Watch dog functionality is described in section 35. User can override reset request with control register described in section 7.3.
1553 Reset Request	MIL-1553 peripheral interfaces and reset request logic	MIL-1553B codec request for reset of MIL-1553B interface support.
TDP	MIL-1553B and SpaceWire	Internal bus for communication between the SpaceWire Time Distribution Protocol core and the SpaceWire interface or the MIL-1553B interface. For more information see section 34.
DSU	DSU and LEON3FT	Debug interface for direct access and control of the LEON3FT processor from debug interface.
APWM	APWM, GPIO, DAC and ADC	APWM synchronization tick outputs. Ticks or events can be programmed individually for each PWM to be generated at PWM compare points, PWM period match, or not generated at all. PWM ticks are distributed in the system to synchronize events to the PWM output.
L3STAT	To LEON3 Statistical Unit	Connection for counting events in the system defined in table 571 under section "Implementation specific events" and in section "Events generated from REQ/GNT signals". Bus is only passively listening.
TRACE	From AMBA infrastructure to Trace buffer	Main and DMA AMBA buses are routed to the trace buffer. Trace buffer is passively listening to signals.

2.3 Analog Architecture Overview

The analog/mixed signal and power-supply IP blocks are presented here. In figure 9, a simplified block diagram shows these blocks and their analog and power interconnections in the GR716B microcontroller.

Figure 9. Simplified block diagram of the analog/mixed and power-supply IPs in the GR716B microcontroller.



Generally, note that when the crystal oscillator (XO) and PLL are used to generate the GR716B microcontroller clocks, these two blocks must be correctly connected and configured to obtain correct digital functionality of the GR716B microcontroller. Moreover, to obtain correct analog functionality of the GR716B microcontroller, e.g., the internal LDOs, the voltage and current references, set by VREF and RREF, must be correctly connected and configured, since they provide the GR716B microcontroller with the internal references and bias currents required by the whole chip in figure 9. Therefore, the VREF capacitor and RREF resistor should always be connected.

2.3.1 Reset and Brownout detector

The Reset and Brownout detector blocks supervise the supply voltages as shown in figure 9. The Reset block provides reset of the internal GR716B microcontroller logic. The internally generated reset is available externally as a 3.3V CMOS output, RESET_OUT_N, which is low when V_{DD_CORE} is too low. There is also a reset release delay at power up, starting to count when V_{DD_CORE} goes above its reset threshold level.

The Brownout detectors are intended to be used as pre-warnings to the GR716B microcontroller that a supply voltage has started to go down, so the CPU can perform a well-controlled system shutdown before the reset level is activated. The Brownout detectors are implemented as one detector per supply to be supervised, and they have individually programmable threshold levels. Each Brownout detector can generate interrupt, which typically shuts down the system in a controlled way.

For more information, see section 8.

2.3.2 Crystal Oscillator (XO)

The XO is supplied by the LEON3FT microcontroller core voltage, V_{DD_CORE} (1.8V), and the oscillator output is a 3.3V CMOS output and is available on external pin. When the XO output is used as system clock, the power-on reset block must have long enough reset-release delay, set by C_RST, to ensure that the XO has started before the reset is released. Therefore, the recommended value of C_RST is at least $100nF_{nom}$, see figure 9. For more information see section 9.

2.3.3 PLL

The PLL is supplied by 1.8V from an internal LDO, which should have an external decoupling capacitor between the PLL supply pins. This supply pin shall be left open, with exception of this decoupling capacitor. The PLL provides several internal clock outputs, typically used as clock for the SpaceWire interface, internal system clock, etc. The PLL input clock is a 3.3V CMOS input, SPW_CLK, to which the XO-oscillator clock output can be directly connected, or any other clock signal generated on PCB fulfilling the electrical specification of this input. The PLL input clock is allowed to be asynchronous to any other clocks in the GR716B microcontroller, and it has no guaranteed phase relation to the internal clocks output from the PLL. For more information see sections 4 and 10.

2.3.4 Precision Voltage Reference

The internal analog reference generation is supplied by V_{DDA_REF} and V_{SSA_REF}. This supply needs to have the best voltage integrity on the chip. Therefore, no fast load-current steps are allowed on this supply. It is essential that especially this supply has good PCB decoupling/filtering (connected directly across V_{DDA_REF} and V_{SSA_REF}) in order to not feed external disturbances from PCB supplies into this supply. The analog internal references are generated in two steps. First, a reference voltage is generated by an on-chip bandgap reference, which should have an external decoupling capacitor, 4.7nF, on the VREF pin to V_{SSA_REF}. Second, this reference voltage is buffered and put out on the VREFBUF pin. The VREFBUF pin voltage can be internally sensed by one of the on-chip ADCs.

The internal reference voltage is also used by an internal current generator, which puts this voltage across an external reference resistor, RREF on PCB, to generate a precision reference current. Since this reference current is used to generate both the current reference to each DAC and the internal bias currents required by several other internal blocks, RREF *cannot* be chosen arbitrarily to get any desired DAC full-scale value. It needs to be 5.11kohm (or 4.64kohm + 464ohm, which are more frequently used standard values).

For more information see section 11.

2.3.5 Internal ADC

There are four independent ADC blocks (ADC0-3). Each ADC can operate as 11/14bit at 500/80kSps, and has an analog MUX on its input, which means that one MUX channel at a time can be measured per ADC. ADC0-2 share the same 8 analog input pins (GPIO37-44). while ADC3 is connected to a different set of 8 pins (GPIO51-58). All GR716B ADC inputs are also analog comparator (ACOMP) inputs. Pin connections and connections to on-chip sensors/sources are shown in Figure 10 and Figure 11.

Each ADC can be programmed for single-ended measurement on one analog input pin (range 0V to 2.5V), and for differential-input measurement (differential range -2V to 2V) using two input pins per channel. In-between the ADC and MUX, there is a fully differential pre-amplifier, which has three programmable gain settings (x1, x2, x4), or can be by-passed. It is to be used together with the differential ADC mode.

The supply (V_{DDA_ADC} and V_{SSA_ADC} for ADC0-2, and V_{DD_IO} and GND for ADC3), needs to be well decoupled/filtered at high frequencies (>1MHz) to not degrade the ADC performance. VSSA_ADC & GND for CQFP, and VSSA_ADC & GND_ADC3 for PBGA, must be locally connected to same PCB ground as VSSA_REF, to maintain ADC functionality and performance.

For a detailed description of the internal ADCs, see section 12.

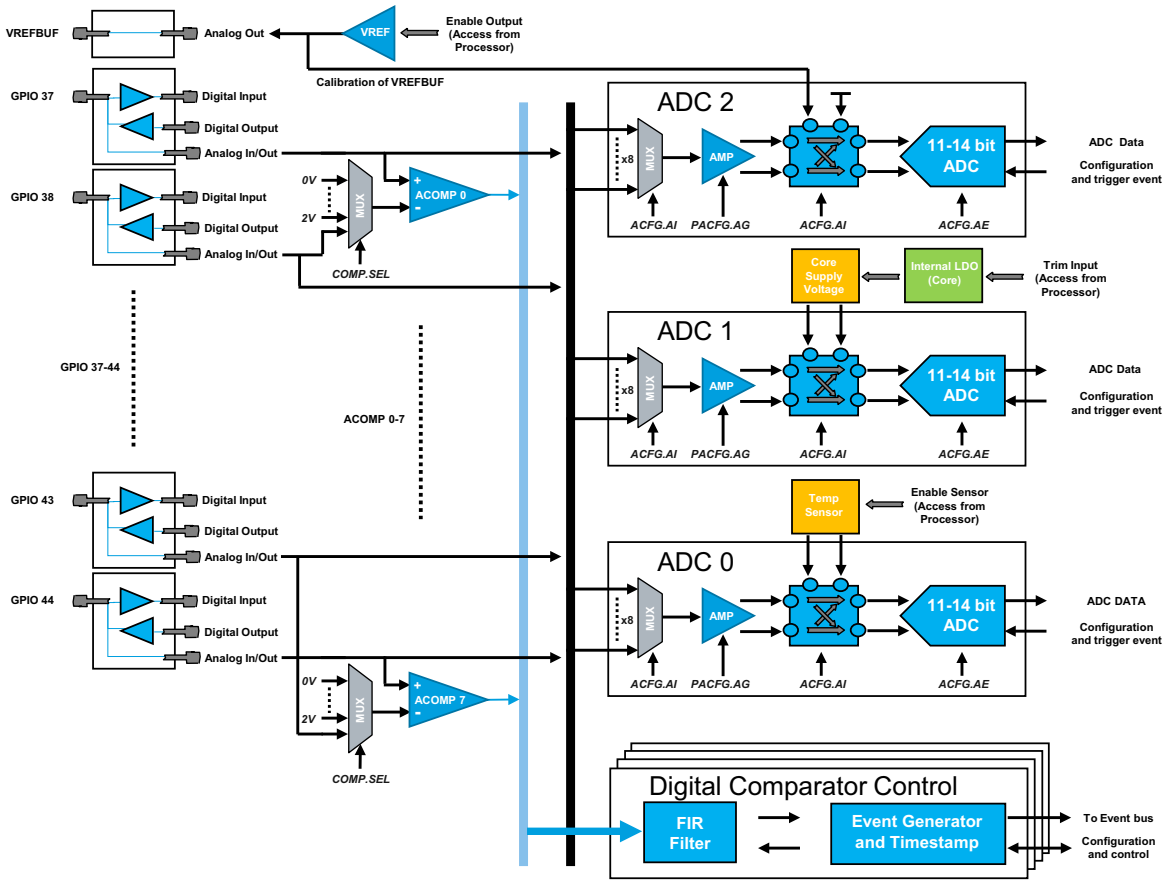


Figure 10. Shared external connections and internal fast analog comparator connections for ADC0, ADC1 and ADC2.

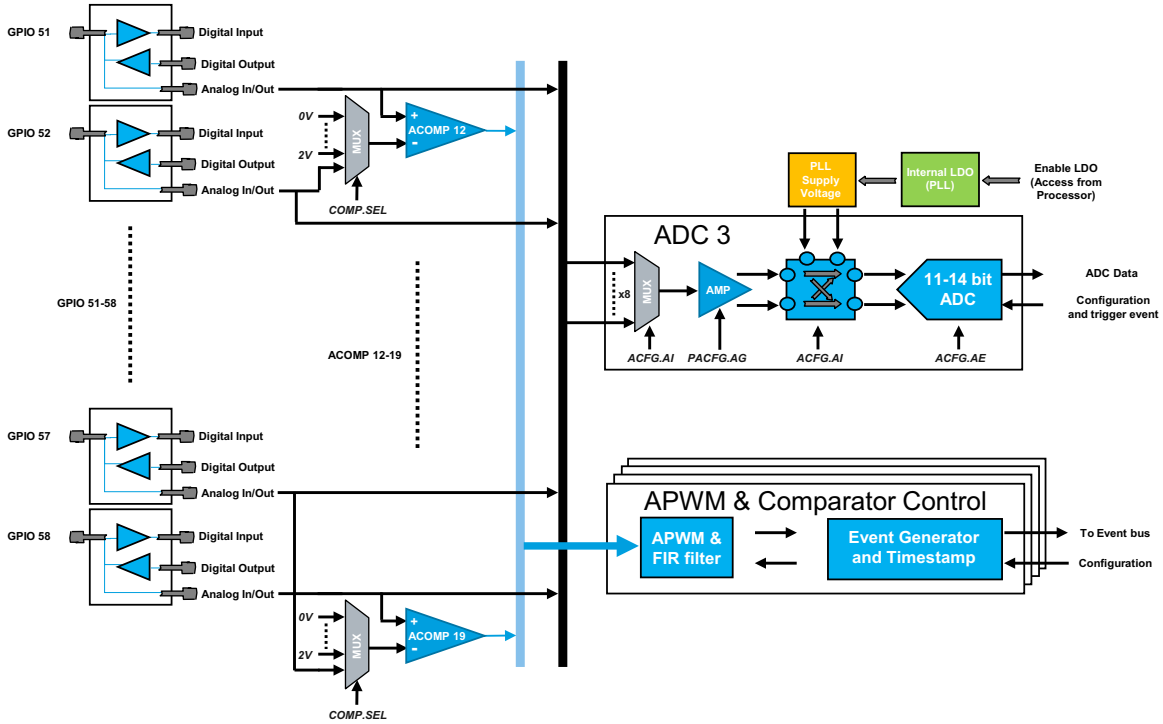


Figure 11. External connection and internal fast analog comparator connections for ADC3.

2.3.6 Internal DAC

There are four independent 12-bit/3 MSps (ramp generation up to 25 MS/s) internal DACs connected to external pins (GPIO45-48). The DAC output is a sourcing-current single-ended output of 0 to 4 mA, typically to be loaded by virtual ground generated by an op-amp on PCB, or by a passive impedance connected to PCB ground providing the output voltage directly across this impedance. The DAC output pins can be used as reference voltage source for four on-chip analog comparators (ACOMP 8-11). Pin connections are shown in Figure 12.

These four DACs are supplied by V_{DDA_DAC} and V_{SSA_DAC}. Good decoupling of the supply is required at high frequencies (>1MHz).

For a detailed description of the internal DACs, see section 15.

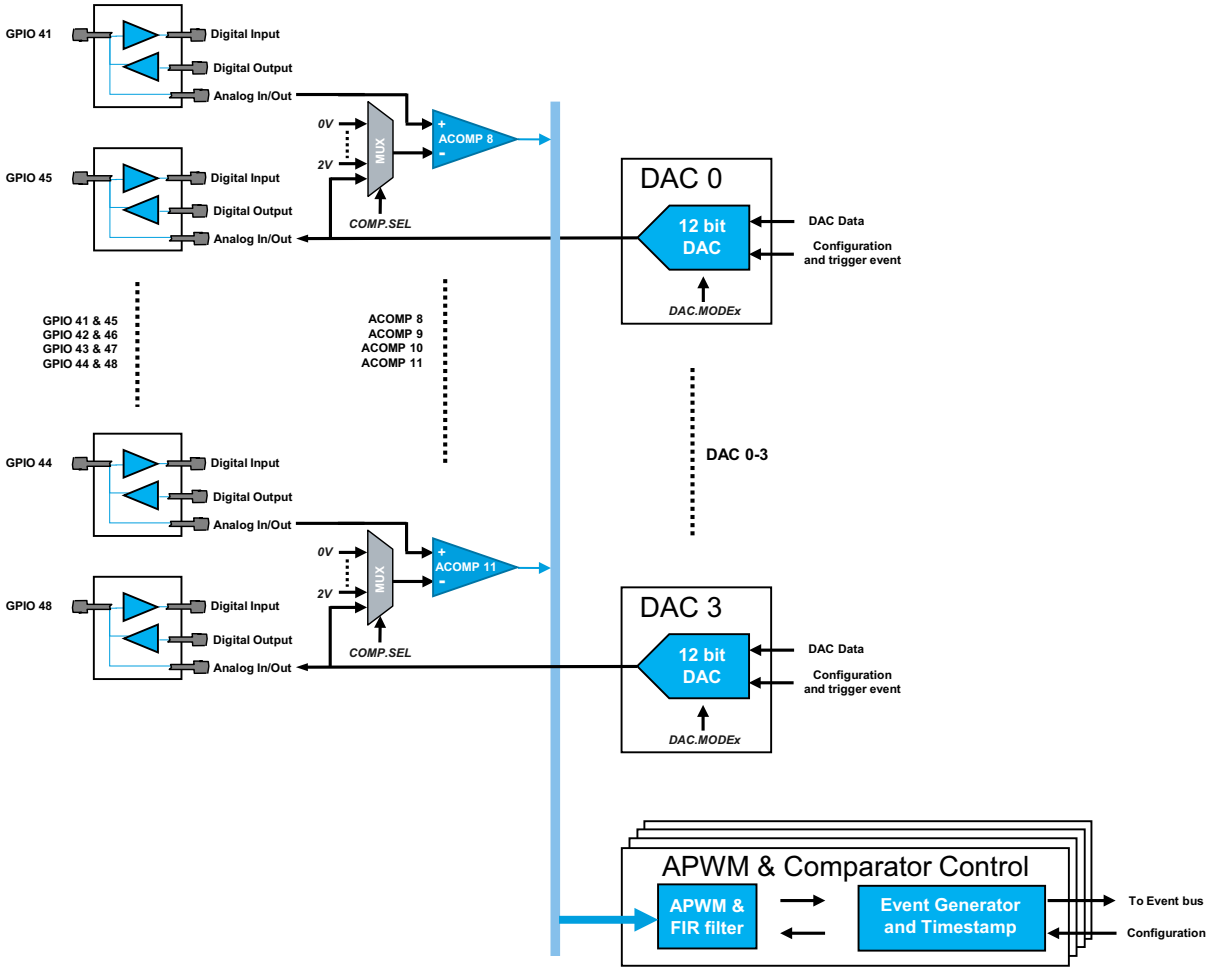


Figure 12. DAC connection to external pin and internal fast analog comparator

2.3.7 Internal LDO

The LDO provides V_{DD_CORE} with a regulated 1.8V, and needs a 3.3V input supply, V_{DD_LDO}, see figure 9. The LDO can be by-passed, then the V_{DD_CORE} pins are directly fed with 1.8V regulated supply voltage from PCB. In this case, the V_{DD_LDO} pins must not be connected to any low-impedance node other than V_{DD_CORE}. In any supply case, all V_{DD_CORE} pins must be decoupled on PCB with a small capacitor, on the order of 10 nF, directly at each V_{DD_CORE}/GND pin pair.

The LDO will cause additional internal power dissipation: the core average current times the LDO voltage drop. It will further increase the junction temperature, which should be carefully checked especially when the LDO is in use at the same time as core current is high.

For more information, see section 13.

2.3.8 Temperature Sensor

There is a temperature sensor implemented on the GR716B microcontroller chip. Its output signal is a monotonic voltage versus temperature, and is measured by internal ADC0. Its output is not threshold detected or used in any other internal function, so if a chip over-temperature protection is desired, the user needs to measure the sensor and take adequate actions in the system application at hand, e.g. a digital trigger level can be programmed. The temperature sensor can also be as precharge source by ADC0. For more information see sections 14 and 12.3.4.

2.3.9 Core Supply Voltage Monitor

The core voltage, V_{DD_CORE}, can be monitored via internal ADC1. The voltage measured can be used by the application to trim the core voltage when the internal LDO is in use. The default LDO trim value is maximum voltage, to always guarantee functionality in worst case corners at maximum supply current for the LDO. For low power applications the core voltage can be decreased to optimum level in order to minimize power consumption. The core voltage can also be used as a precharge source by ADC1. See also section 12.3.4.

2.3.10 Reference Output Voltage Monitor

The VREFBUF output voltage can be monitored via internal ADC2. It can be used as calibration measurement of VREFBUF in bridge measurements such as in the thermistor measurement example in Figure 13. VREFBUF can also be used as a precharge source by ADC2. See also section 12.3.4.

2.3.11 PLL Supply Voltage Monitor

The PLL supply voltage, V_{DDA_PLL}, can be monitored via internal ADC3. This supply voltage is generated by an internal LDO, and can be monitored for abnormal long-term drifts. The PLL supply voltage can also be used as a precharge source for ADC3. See also section 12.3.4.

2.3.12 Fast analogue comparator

Fast analog comparators (ACOMP) are integrated into the analog IO of GR716B, and are connected to digital control and filter logic. Each analog comparator can be individually programmed with one input to a fixed-voltage trig level: 0, 0.125, 0.25, 0.5, 1.0, 1.5 or 2.0 [V]. Alternatively, both ACOMP inputs can be made available on package pins. The pins with comparator input capability are also used as ADC inputs and DAC outputs. See sections 2.3.5, 2.3.6 and 2.5.

2.3.13 Precision reference output and thermistor bridge measurement

The precision reference voltage output, VREFBUF, can be enabled and used to facilitate applications such as thermistor measurements. The drive strength for VREFBUF output is 20mA sourcing current, which makes it suitable to drive many thermistors from one GR716B.

VREFBUF can be calibration measured by ADC2. Hence, if ADC2 is used to also measure the thermistor channel (any of GPIO 37-44), then these two measurements can give equivalent accuracy after software calculation as a bridge measurement provides, which considerably improves result accuracy.

See sections 12 and 14 for more information.

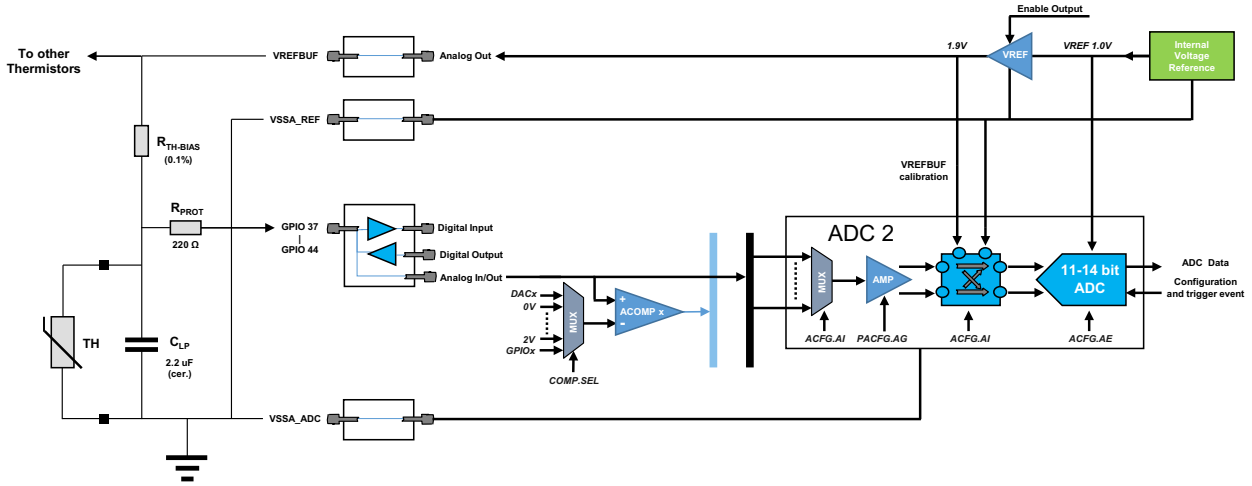


Figure 13. Thermistor measurement based on bridge measurement by VREFBUF calibration

2.3.14 LVDS Common-mode, Cold-Spare and Fail-safe support

The extended common mode support makes GR716B suitable also for cable communication, and cold-spare function ensures LVDS transceiver high impedance when supply is tied to ground, or LVDS transmitter is disabled from software.

The LVDS receivers feature an internal fail-safe function, to ensure a known receiver state in case of input connection failure.

2.3.15 Analog test bus

There are separate analog test bus structures and test output buffers implemented in GR716B. Each individual test buffer can be made accessible via dedicated GPIO pins in analog test mode. The main purpose of the analog test bus is to measure and characterize the device in production. For more information contact support@gaisler.com.

2.4 Signal Overview

The GR716B microcontroller has 64 external general purpose user input and outputs, dedicated SPI memory interface, 4 LVDS receivers and 4 LVDS transmitters in CQFP (6 LVDS transmitters available in PBGA and SIP devices). Almost all 64 external inputs and outputs and LVDS transceivers have multiple functionality. Functionality is selected by the application software during startup and configuration. During startup i.e. after reset all user input and outputs are configured as inputs.

2.5 I/O switch matrix overview

This section provides a introduction to the I/O switch matrix and gives a presentation to the predefined set of pin configuration.

The I/O switch matrix provides access to several I/O units. When an interface is not activated, its pins automatically become general purpose I/O. After reset, all I/O switch matrix pins are defined as inputs until programmed otherwise. Configuration and assigning of functions to external I/O is flexible and is controlled by software via registers described in section 7.1.

Table 7 shows a listing of all external CMOS pins in the I/O switch matrix and what functions can be assign to external pins. Table 7 also shows configuration registers to assign specific function or pin to external I/O. To assign a specific function or pin to an external interface the “column” value should be written into the table “row” I/O configuration register and bit field. E.g. register SYS.CFG.R0.GP5 described in section 7.1.

All analog connections are present regardless of I/O switch matrix configuration in SYS.CFG register. The usage of the analog features are not controlled centrally, but within the control interface of each analog block. The connections between analog blocks and GPIOs are given in Table 8.

Table 7. I/O switch matrix configuration options per pin for GPIO0-63, LVDS_RX0-3 and LVDS_TX0-5.

REG		0x0	0x1 [6]	0x2 [7]	0x3	0x4	0x5	0x6 [11]	0x7 [12]	0x8	0x9 [14]	0xA [14]	0xB [14]	0xC [15]	0xD	0xE	0xF
SYS.CFG.R0	GP0	GPIO0 [1]	UART_RTSN0	MEM0_ADDR0	1553_RXENA		CAN_TX0		SPIM_SLV1			SYNC_1	ALARM_0	tACOMP_FAST			
	GP1	GPIO1 [1]	UART_CTNS0	MEM0_ADDR1	1553_TXA		CAN_RX0		SPIM_SCK1		AB0_A	CF0_C	G_0				
	GP2	GPIO2 [1]	UART_TX0	MEM0_ADDR2	1553_RXA		CAN_SEL0	I2CM_SDA0	SPIM_MOSI1		AB0_B	CF0_D	G_1	tDAC_I_D0			
	GP3	GPIO3 [1]	UART_RX0	MEM0_ADDR3	1553_RXNA		CAN_RX1	I2CM_SCL0	SPIM_MISO1		AB1_A	CF0_E	G_2	tDAC_I_D1			
	GP4	GPIO4 [1]	UART_CTSN1	MEM0_ADDR4	1553_TXNA		CAN_TX1	I2CM_SDA1	SPI_SCK0		AB1_B	CF0_F	G_3	tDAC_I_D2			
	GP5	GPIO5 [1]	UART_RTSN1	MEM0_ADDR5	1553_TXINHA		CAN_SEL1	I2CM_SCL1	SPI_MISO0		A0_0	CF1_C	G*_0	tDAC_I_D3			
	GP6	GPIO6 [1]	UART_TX1	MEM0_ADDR6	1553_RXB			I2CS_SDA0	SPI_MOSI0		A0_1	CF1_D	G*_1	tDAC_I_D4			
	GP7	GPIO7 [1]	UART_RX1	MEM0_ADDR7	1553_RXNB			I2CS_SCL0	SPI_SEL0		A0_2	CF1_E	G*_2	tDAC_I_D5			
SYS.CFG.R1	GP0	GPIO8 [1]	UART_CTSN2	MEM0_ADDR8	1553_RXENB			I2CS_SDA1	SPI_SLV0_0		A0_3	CF1_F	G*_3	tDAC_I_D6			
	GP1	GPIO9 [1]	UART_RTSN2	MEM0_ADDR9	1553_TXB			I2CS_SCL1	SPI_SLV0_1		PDAC0_CLK [13]	CF2_C	G*_4	tDAC_I_D7			
	GP2	GPIO10 [1]	UART_TX2	MEM0_ADDR10	1553_CLK			I2CS_SDA2	SPI_SLV0_2		PDAC0_A [13]	CF2_D	G*_5	tDAC_I_D8			
	GP3	GPIO11 [1]	UART_RX2	MEM0_ADDR11	1553_TXNB			I2CS_SCL2	SPI_SLV0_3		PDAC0_B [13]	CF2_E	G*_6	tDAC_I_D9			
	GP4	GPIO12 [1]		MEM0_ADDR12	1553_TXINHB						PDAC0_C [13]	CF2_F	G*_7	tDAC_I_D10			
	GP5	GPIO13 [1]	UART_CTSN3	MEM0_ADDR13			CAN_TX0		SPI_SCK1		PDAC1_CLK [13]	G_0	G*_8	tDAC_I_D011			
	GP6	GPIO14 [1]	UART_RTSN3	MEM0_ADDR14			CAN_RX0		SPI_MISO1		PDAC1_A [13]	G_1	G*_9	tADC_O_D0			
	GP7	GPIO15 [1]	UART_TX3	MEM0_ADDR15			CAN_SEL0		SPI_MOSI1		PDAC1_B [13]	G_2	G*_10	tADC_O_D1			
SYS.CFG.R2	GP0	GPIO16 [1]	UART_RX3	MEM0_ADDR16			CAN_RX1		SPI_SEL1		PDAC1_C [13]	G_3	G*_11	tADC_O_D2			
	GP1	GPIO17 [1]	UART_RTSN4	MEM0_ADDR17			CAN_TX1		SPI_SLV1_0			ALARM_1	SYNC_0	tADC_O_D3			
	GP2	GPIO18 [1]	UART_TX4	MEM0_ADDR18			CAN_SEL1		SPI_SLV1_1		AB2_A	G*_0	CF0_C	tADC_O_D4			
	GP3	GPIO19 [1]	UART_RX4	RAM0_CSN0							AB2_B	G*_1	CF0_D	tADC_O_D5	TDP_SETET	tPLL_LOCK [15]	
	GP4	GPIO20 [1]	UART_CTSN4	RAM0_CSN1							AB3_A	G*_2	CF0_E	tADC_O_D6	TDP_E_ET_I	tSCRUB_CLK [15]	
	GP5	GPIO21 [1]	UART_CTSN5	RAM0_CSN2					SPI_SLV1_2		AB3_B	G*_3	CF0_F	tADC_O_D7		tMIL_CLK [15]	
	GP6	GPIO22 [1]	UART_RTSN5	RAM0_CSN3					SPI_SLV1_3		PDAC2_CLK [13]	G*_4	CF1_C	tADC_O_D8		tSPW_CLK [15]	
	GP7	GPIO23 [1]	UART_TX5	ROM0_CSN0							PDAC2_A [13]	G*_5	CF1_D	tADC_O_D9		tSYS_CLK [15]	
SYS.CFG.R3	GP0	GPIO24 [1]	UART_RX5	ROM0_CSN1							PDAC2_B [13]	G*_6	CF1_E	tADC_O_D10			
	GP1	GPIO25 [1]		MEM0_DATA0				I2CM_SDA0	SPI_SCK0		PDAC2_C [13]	G*_7	CF1_F	tADC_O_EOC		SCRUB_INITN	
	GP2	GPIO26 [1]		MEM0_DATA1				I2CM_SCL0	SPI_MISO0		PDAC3_CLK [13]	G*_8	CF2_C	tADC_O_CMP		SCRUB_DONE	
	GP3	GPIO27 [1]		MEM0_DATA2				I2CM_SDA1	SPI_MOSI0		PDAC3_A [13]	G*_9	CF2_D	tBO_O		SCRUB_DATA0	
	GP4	GPIO28 [1]		MEM0_DATA3				I2CM_SCL1	SPI_SEL0		PDAC3_B [13]	G*_10	CF2_E	tCLKDET_OK		SCRUB_DATA1	
	GP5	GPIO29 [1]		MEM0_DATA4				I2CS_SDA0	SPI_SLV0_0		PDAC3_C [13]	G*_11	CF2_F	tCLKDET_RO0		SCRUB_DATA2	tRINGOSC0
	GP6	GPIO30 [1]		MEM0_DATA5				I2CS_SCL0	SPI_SLV0_1		ALARM_2	SYNC_0	AB0_A	tCLKDET_E0		SCRUB_DATA3	
	GP7	GPIO31 [1]		MEM0_DATA6				I2CS_SDA1	SPI_SLV0_2		SYNC_1	ALARM_3	AB0_B	tCLKDET_RO1		SCRUB_DATA4	tRINGOSC1
SYS.CFG.R4	GP0	GPIO32 [1]		MEM0_DATA7				I2CS_SCL1	SPI_SLV0_3		A0_0	ALARM_4	SYNC_1	tCLKDET_E1		SCRUB_DATA5	
	GP1	GPIO33 [1]	UART_CTSN3	MEM0_OEN				I2CM_SDA0			A0_1	PDAC0_CLK [13]	AB1_A	tLVDS_FSRX		SCRUB_DATA6	
	GP2	GPIO34 [1]	UART_RTSN3	MEM0_WRN				I2CM_SCL0			A0_2	PDAC0_A [13]	AB1_B			SCRUB_DATA7	
	GP3	GPIO35 [1]	UART_TX3	ROM0_CSN0				I2CS_SDA0			A0_3	PDAC0_B [13]	MEM0_BRDYN [7]			SCRUB_PROG	
	GP4	GPIO36 [1]	UART_RX3	ROM0_CSN1				I2CS_SCL0			ALARM_5	PDAC0_C [13]	MEM0_BEXCN [7]			SCRUB_RDWR	
	GP5	GPIO37 [2]	UART_RTSN4		1553_RXENA		CAN_TX0		SPI_SLV0_3	AINA0 [2]				tACOMP_SET0		SCRUB_CSIN	
	GP6	GPIO38 [2]	UART_TX4		1553_TXA		CAN_RX0		SPI_SLV0_2	AINA1 [2]						SCRUB_SCLK	
	GP7	GPIO39 [2]	UART_RX4		1553_RXA		CAN_SEL0	I2CS_SDA1	SPI_SLV0_1	AINA2 [2]							
SYS.CFG.R5	GP0	GPIO40 [2]	UART_CTSN4		1553_RXNA			I2CS_SCL1	SPI_SLV0_0	AINA3 [2]							
	GP1	GPIO41 [2]	UART_CTSN5	RAM0_CSN2	1553_TXNA		CAN_TX0	I2CM_SDA0	SPI_SCK0	AINA4 [2]	AB0_LEB [2]						
	GP2	GPIO42 [2]	UART_RTSN5	RAM0_CSN3	1553_TXINHA		CAN_RX0	I2CM_SCL0	SPI_MISO0	AINA5 [2]	AB1_LEB [2]						
	GP3	GPIO43 [2]	UART_TX5	ROM0_CSN0	1553_RXB		CAN_SEL0	I2CS_SDA0	SPI_MOSI0	AINA6 [2]	AB2_LEB [2]						
	GP4	GPIO44 [2]	UART_RX5	ROM0_CSN1	1553_RXNB		CAN_RX1	I2CS_SCL0	SPI_SEL0	AINA7 [2]	AB3_LEB [2]						
	GP5	GPIO45 [2]	UART_RX0		1553_RXENB		CAN_RX1		SPI_SLV1_1	AOUT0 [2]							ETH_TXCLK
	GP6	GPIO46 [2]	UART_TX0		1553_TXB				SPI_SLV1_0	AOUT1 [2]							ETH_RXCLK
	GP7	GPIO47 [2]	UART_CTSN0		1553_CLK		CAN_TX1	I2CM_SDA1	SPI_SCK1	AOUT2 [2]							ETH_MDIO

Table 7. I/O switch matrix configuration options per pin for GPIO0-63, LVDS_RX0-3 and LVDS_TX0-5.

	REG	0x0	0x1 [6]	0x2 [7]	0x3	0x4	0x5	0x6 [11]	0x7 [12]	0x8	0x9 [14]	0xA [14]	0xB [14]	0xC [15]	0xD	0xE	0xF
SYS.CFG.R6	GP0	GPIO48 [2]	UART_RTSN0		1553_TXNB		CAN_SEL1	I2CM_SCL1	SPI_MISO1	AOUT3 [2]							ETH_MDC
	GP1	GPIO49 [1]	UART_CTSN0	MEM0_ADDR19	1553_TXINHB			I2CS_SDA2	SPI_MOSI1				SPIM_SLV1 [12]	AHBUART_TX [6]	TDP_SETET		ETH_INT
	GP2	GPIO50 [1]	UART_TX0	MEM0_ADDR20				I2CS_SCL2	SPI_SEL1				SPIM_SCK1 [12]	AHBUART_RX [6]	TDP_E_ET_I		ETH_RXD3
	GP3	GPIO51 [2]	UART_RX0	MEM0_ADDR21	1553_RXENA				SPI_MOSI1	AINB0 [2]	AB0_TEB [2]		SPIM_MOSI1 [12]		TDP_PULSE0		ETH_RXD2
	GP4	GPIO52 [2]	UART_CTSN1	MEM0_ADDR22	1553_TXA			I2CM_SDA0	SPI_SEL1	AINB1 [2]	AB1_TEB [2]		SPIM_MISO1 [12]		TDP_PULSE1		ETH_RXD1
	GP5	GPIO53 [2]	UART_RTSN1		1553_RXA			I2CM_SCL0	SPIS_SCK0	AINB2 [2]	AB2_TEB [2]		SPI4S_SCK_R [12]				ETH_RXD0
	GP6	GPIO54 [2]	UART_TX1		1553_RXNA			I2CM_SDA1	SPIS_MISO0	AINB3 [2]	AB3_TEB [2]		SPI4S_MISO_R [12]				ETH_RXDV
	GP7	GPIO55 [2]	UART_RX1		1553_TXNA			I2CM_SCL1	SPIS_MOSI0	AINB4 [2]		A0_LEB [2]	SPI4S_MOSI_R [12]	TIMER32_TICKGEN			ETH_RXER
SYS.CFG.R7	GP0	GPIO56 [2]	UART_CTSN2		1553_TXINHA			I2CS_SDA0	SPIS_SLV0	AINB5 [2]		A1_LEB [2]	SPI4S_CS_R [12]	tEVENT_TRIG			ETH_COL
	GP1	GPIO57 [2]	UART_RTSN2		1553_RXB			I2CS_SCL0	SPI_SCK0	AINB6 [2]		A2_LEB [2]		tDNCNT24			ETH_CRS
	GP2	GPIO58 [2]	UART_TX2		1553_RXNB		CAN_TX0	I2CS_SDA1	SPI_MISO0	AINB7 [2]		A3_LEB [2]		tDNCNT12			ETH_TXD3
	GP3	GPIO59 [5]	UART_RX2		1553_RXENB		CAN_RX0	I2CS_SCL1	SPI_MOSI0					tTIMER32	tACOMP_SET1		ETH_TXD2
	GP4	GPIO60 [5]	UART_CTSN3		1553_TXB		CAN_SEL0	I2CS_SDA2	SPI_SEL0					tTIMER27	TDP_PULSE2		ETH_TXD1
	GP5	GPIO61 [5]	UART_RX3		1553_CLK		CAN_RX1	I2CS_SCL2	SPI_SLV0_0					tTIMEALARM	TDP_PULSE3		ETH_TXD0
	GP6	GPIO62 [5]	UART_TX3	ROM0_CSN0	1553_TXNB		CAN_TX1		SPI_SLV0_1					tPARITYERR	TDP_PULSE4		ETH_TXEN
	GP7	GPIO63 [1]	UART_RTSN3	ROM0_CSN1	1553_TXINHB		CAN_SEL1		SPI_SLV0_2					tAPWM_OUT	TDP_PULSE5		
SYS.CFG.LVDS0-1	LVDS0.TX0 [3]	SPW_TXD0		SPI_SCK0 [8] [12]		LVDS_OUT0 [10]				Disabled							
	LVDS0.TX1 [3]	SPW_TXS0		SPI_SLV0_0 [8] [12]		LVDS_OUT1 [10]				Disabled							
	LVDS0.TX2 [3] [4]	SPW_TXD1	SPI4S_MISO_N [12]	SPI_MOSI0 [8] [12]	SPI_MISO0 [9] [12]	LVDS_OUT2 [10]				Disabled							
	LVDS0.RX0 [3]	SPW_RXD1	SPI4S_SCK_N [12]	SPI_MISO0 [8] [12]	SPI_SCK0 [9] [12]	LVDS_IN0 [10]				Disabled							
	LVDS0.RX1 [3]	SPW_RXD0	SPI4S_MOSI_N [12]		SPI_MOSI0 [9] [12]	LVDS_IN1 [10]				Disabled							
	LVDS0.RX2 [3]	SPW_RXS0	SPI4S_CS_N [12]		SPI_SEL0 [9] [12]	LVDS_IN2 [10]				Disabled							
	LVDS0.TX3 [3] [4]	SPW_TXS1				LVDS_OUT3 [10]				Disabled							
	LVDS0.RX3 [3]	SPW_RXS1				LVDS_IN3 [10]				Disabled							
	LVDS1.TX4 [3] [5]	SPW_TXD1	SPI4S_MISO_N [12]	SPI_MOSI0 [8] [12]	SPI_MISO0 [9] [12]	LVDS_OUT4 [10]				Disabled							
	LVDS1.TX5 [3] [5]	SPW_TXS1				LVDS_OUT5 [10]				Disabled							

Note 1: These pins have programmable Schmitt trigger input mode selectable in the SYS.CFG.SCHMITT0/1 register. See section 7.1. Note that some functions take their input from the Schmitt trigger path independently of the I/O switch matrix configuration. Whenever this is the case, it is documented at the function description.

Note 2: These pins have analog capability and are connected to comparator inputs (ACOMP) and ADC inputs or DAC outputs. See Table 8 for analog connections and sections 12, 15 and 16 for detailed functionality. Before applying an analog voltage level on one of these pins, it should be placed in analog mode as described in section 2.8.3. The AIN*, AOUT* and *_LEB/*_TEB mode selections result in analog mode. LEB/TEB refers to Analogue PWM Leading Edge Blanking / Trailing Edge Blanking signal. Signal will drive 'Low' or 'HiZ'. Used with ACOMP on the same pin. See section 53.2 and 53.3 for more information.

Note 3: LVDS pin assignment depends on package option. Refer to datasheet section 3.2 [DS].

Note 4: Not available in CQFP package. Only available in PBGA and SIP packages.

Note 5: In the CQFP package, LVDS_TX4 and LVDS_TX5 are available via GPIO pins 59/60 and 61/62. In the PBGA and SIP packages, LVDS_TX4 and LVDS_TX5 are available on dedicated pins. On all packages when LVDS port is enabled using SYS.CFG.LVDS1.TX4 or SYS.CFG.LVDS1.TX5 then the CMOS outputs of the respective GPIO pins are set to 'HiZ'. When GPIO 59/60 or 61/62 are utilized, then the corresponding SYS:CFG.LVDS1.TX4/5 must be disabled.

Note 6: UART_* signals are from APBUART units (section 19, UART Serial interface). AHBUART_* signals are from AHBUART units (section 48, Serial and Remote Access Interface). The two are not interchangeable.

Note 7: MEM0_*, RAM0_*, and ROM0_* signals are from the FTMCTRL0 memory controller (section 22, Fault Tolerant PROM/SRAM Memory Interface). They are distinct from the dedicated MEM1_* signals available in some package types (section 23).

Note 8: SPI controller 0 in master mode can use LVDS I/O (LVDS mux mode 0x2). SPI controller 0 is described in section 44, SPI Controller.

Note 9: SPI controller 0 in slave mode can use LVDS I/O (LVDS mux mode 0x3). SPI controller 0 is described in section 44, SPI Controller.

Note 10: LVDS I/O may be used in GPIO mode (LVDS_OUT/LVDS_IN) for general purpose input and output (LVDS mux mode 0x4). See section 56.

Note 11: The I2C* signals connect to I2C controllers of three different kinds:
I2CM_SCL0/SDA0 and I2CM_SCL1/SDA1 connect to I2CMST0 and I2CMST1, respectively (section 38, I2C master).
I2CS_SCL0/SDA0 and I2CS_SCL1/SDA1 connect to I2CSLV0 and I2CSLV1, respectively (section 39, I2C slave).
I2CS_SCL2/SDA2 connect to I2C2AHB (section 37, I2C to AHB bridge)

Frontgrade Gaisler AB
Kungsgatan | SE-411 19 | Goteborg | Sweden
+46 31 7758650 | frontgrade.com/gaisler

Table 7. I/O switch matrix configuration options per pin for GPIO0-63, LVDS_RX0-3 and LVDS_TX0-5.

REG	0x0	0x1 [6]	0x2 [7]	0x3	0x4	0x5	0x6 [11]	0x7 [12]	0x8	0x9 [14]	0xA [14]	0xB [14]	0xC [15]	0xD	0xE	0xF

Note 12: The SPI* signals connect to SPI controllers of four different kinds:

SPIM_* signals in the I/O switch matrix connect to SPIMCTRL1 (section 46, SPI Memory Controller). SPIMCTRL0 has dedicated pins.
SPI_* signals connect to the SPICTRL0 and SPICTRL1 (section 44, SPI Controller). They may be operated either in master or slave mode. SPICTRL0 has the option of using LVDS I/O. See Note 8 and Note 9 above.
SPI4S_* signals connect to the SPI4S (section 45, SPI for Space Slave Controller). The nominal interface (*_N) can use LVDS I/O.

Note 13: The PDAC* signals are digital outputs of the APWMDAC units, refer to sections 2.6 and 54. When used as an output, PDAC0_CLK can be output on either GPIO 9 or GPIO 33. When used as an input, PDAC0_CLK can only be taken from GPIO[9]. For all PDAC_CLK signals, muxing the GPIO as PDAC_CLK makes the pin an output. To use PDAC*_CLK as an input, the pin must be muxed as an input (for example using mode 0x0).

Note 14: Except where mentioned in notes above, I/O switch matrix modes 0x9, 0xA, and 0xB are used for APWM signals (section 53, Analog Applications Pulse Width Modulation).

Note 15: Signals prefixed with a lower “t” are special purpose digital test outputs and inputs. Some are proprietary. Some are available for diagnostic use. For example one set of test outputs are the internal PLL clocks and lock which can be enabled vi the Test Clock Enable register (Table 81).

Table 8. Analog connections of analog-capable input and output GPIO pins (GPIO37-44, GPIO45-48, and GPIO51-58). The leftmost column contains one or (for compactness) two analog units per row. For compactness, the connections of ACOMP0-7 and ACOMP12-19 share some table rows.

GPIO	37	38	39	40	41	42	43	44	45	46	47	48	51	52	53	54	55	56	57	58
ADC0	AINA0	AINA1	AINA2	AINA3	AINA4	AINA5	AINA6	AINA7												
ADC1	AINA0	AINA1	AINA2	AINA3	AINA4	AINA5	AINA6	AINA7												
ADC2	AINA0	AINA1	AINA2	AINA3	AINA4	AINA5	AINA6	AINA7												
ADC3													AINB0	AINB1	AINB2	AINB3	AINB4	AINB5	AINB6	AINB7
DAC0									AOUT0											
DAC1										AOUT1										
DAC2											AOUT2									
DAC3												AOUT3								
ACOMP0/12	C0P	C0N											C12P	C12N						
ACOMP1/13		C1P	C1N											C13P	C13N					
ACOMP2/14			C2P	C2N											C14P	C14N				
ACOMP3/15				C3P	C3N											C15P	C15N			
ACOMP4/16				C4P	C4N											C16N	C16P			
ACOMP5/17					C5N	C5P											C17N	C17P		
ACOMP6/18						C6N	C6P											C18N	C18P	
ACOMP7/19							C7N	C7P											C19N	C19P
ACOMP8					C8P				C8N											
ACOMP9						C9P				C9N										
ACOMP10							C10P				C10N									
ACOMP11								C11P				C11N								

2.6 I/O switch for APWMDAC output signals

This section lists external GPIO pins with special purpose when used for PWM DAC output applications. See chapter 54 for a detailed description of the APWMDAC units.

Table 9. External APWM DAC pin configuration

Pin Name	Interface Name	APWMDAC #	Functional description
GPIO[9]	PDAC0_CLK 1)	0	External APWM DAC 0 pins. Pins needs external filter to function. See figure 14 and 15 for single or combined PWM DAC output examples. The PDAC0 A, B and C signals may also be output on GPIO 34, 35, and 36. See Table 7.
GPIO[10]	PDAC0_A	0	
GPIO[11]	PDAC0_B	0	
GPIO[12]	PDAC0_C	0	
GPIO[13]	PDAC1_CLK 1)	1	External APWM DAC 1 pins. Pins needs external filter to function. See figure 14 and 15 for single or combined PWM DAC output examples
GPIO[14]	PDAC1_A	1	
GPIO[15]	PDAC1_B	1	
GPIO[16]	PDAC1_C	1	
GPIO[22]	PDAC2_CLK 1)	2	External APWM DAC 2 pins. Pins needs external filter to function. See figure 14 and 15 for single or combined PWM DAC output examples
GPIO[23]	PDAC2_A	2	
GPIO[24]	PDAC2_B	2	
GPIO[25]	PDAC2_C	2	
GPIO[26]	PDAC3_CLK 1)	3	External APWM DAC 3 pins. Pins needs external filter to function. See figure 14 and 15 for single or combined PWM DAC output examples
GPIO[27]	PDAC3_A	3	
GPIO[28]	PDAC3_B	3	
GPIO[29]	PDAC3_C	3	

Note 1: PDAC_CLK can be used either as an input (external clock) or an output (self-generated clock). When used as an output, the corresponding GPIO pin should be muxed for PDAC_CLK function in the I/O switch matrix (see section 2.5). But this mux option must not be used when PDAC_CLK is used as an input. To use PDAC_CLK as a clock input to the APWMDAC, the I/O switch matrix must configure the corresponding GPIO pin as an input. One option is to use mux mode 0x0 and set the pin as an input via the GRGPIO core (chapter 30).

Note 2: During reset and until software has enabled APWMDAC functions in the I/O switch matrix, the GPIO pins will be tristated by the GR716B. External pull resistors should be considered if a well-defined state of the outputs is needed during startup.

Figure 14. Single output PWM DAC filter.

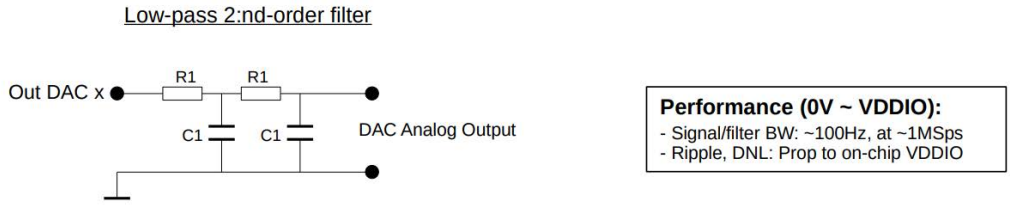
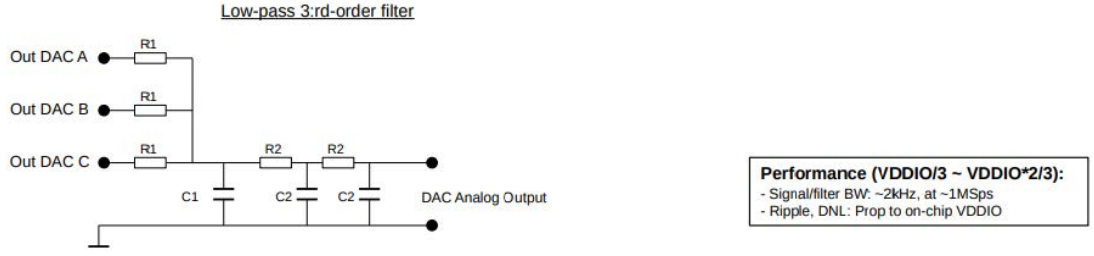


Figure 15. Combined output PWM DAC filter.



2.7 I/O switch default configurations for bootstraps

This chapter lists external pin connection for all valid boot strap options.

2.7.1 External pin configuration for SpaceWire remote access

This section describes valid bootstrap configuration for SpaceWire remote access.

Table 10. CQFP Remote SpaceWire pin configurations

Pin Name	Interface Name	Functional description
LVDS_RX[1]	SPW_RXD0	SpaceWire receiver data interface
LVDS_RX[2]	SPW_RXS0	SpaceWire receiver strobe interface
LVDS_TX[0]	SPW_TXD0	SpaceWire transmitter data interface
LVDS_TX[1]	SPW_TXS0	SpaceWire transmitter strobe interface
LVDS_RX[0]	SPW_RXD1	SpaceWire receiver data interface
LVDS_RX[3]	SPW_RXS1	SpaceWire receiver strobe interface
LVDS_TX[4] 2)	SPW_TXD1	SpaceWire transmitter data interface
LVDS_TX[5] 2)	SPW_TXS1	SpaceWire transmitter strobe interface

Note 1: Remote SpaceWire interface uses LVDS type interface

Note 2: Available vi GPIO Mux

Table 11. PBGA/SIP Remote SpaceWire pin configurations

Pin Name	Interface Name	Functional description
LVDS_RX[1]	SPW_RXD0	SpaceWire receiver data interface
LVDS_RX[2]	SPW_RXS0	SpaceWire receiver strobe interface
LVDS_TX[0]	SPW_TXD0	SpaceWire transmitter data interface
LVDS_TX[1]	SPW_TXS0	SpaceWire transmitter strobe interface
LVDS_RX[0]	SPW_RXD1	SpaceWire receiver data interface
LVDS_RX[3]	SPW_RXS1	SpaceWire receiver strobe interface
LVDS_TX[2]	SPW_TXD1	SpaceWire transmitter data interface
LVDS_TX[3]	SPW_TXS1	SpaceWire transmitter strobe interface
LVDS_TX[4] 2)	SPW_TXD1	SpaceWire transmitter data interface
LVDS_TX[5] 2)	SPW_TXS1	SpaceWire transmitter strobe interface

Note 1: Remote SpaceWire interface uses LVDS type interface

Note 2: Available vi GPIO Mux

2.7.2 External pin configuration for UART remote access

This section describes valid bootstrap configuration for UART remote access. Refer to section 48 for details on the communication protocol.

Table 12. Remote UART access pin configurations

Pin Name	Interface Name	Functional description
GPIO[49]	AHBUART_TX	UART transmitter interface. External pull-up or pull-down is recommended.
GPIO[50]	AHBUART_RX	UART receiver interface
Note 1:	Interface uses CMOS type interface	
Note 2:	The GR716B GPIOs are tristated during reset. The bootstrap option only takes effect after reset has been deasserted. To ensure that AHBUART_TX has a valid state during reset use an external pull-up or pull-down resistor. A high state corresponds to an idle TX line. A low state that persists for a sufficiently long time signals a break condition.	

2.7.3 External pin configuration for I2C remote access

This section describes valid bootstrap configuration for I2C remote access. Refer to section 39 for details on the communication protocol.

Table 13. Remote I2C access pin configurations

Pin Name	Interface Name	Functional description
GPIO[49]	I2C_SDA	I2C Serial Data interface. External pull-up is required.
GPIO[50]	I2C_SCL	I2C Serial Clock interface. External pull-up is required.
Note 1:	Interface uses CMOS type interface	
Note 2:	I2C address should be set using external boot strap pin number 15, 62 and 0	

2.7.4 External pin configuration for SPI remote access

This section describes valid bootstrap configuration for SPI remote access. Refer to section 43 for details on the communication protocol.

Table 14. Remote SPI access pin configurations

Pin Name	Interface Name	Functional description
GPIO[53]	SPIS_SCK	SPI Slave Clock interface
GPIO[54]	SPIS_MISO	SPI Master input Slave output interface. External pull resistor recommended.
GPIO[55]	SPIS_MOSI	SPI Master output Slave input interface
GPIO[56]	SPIS_SLV	SPI Slave Select interface
Note 1:	Interface uses CMOS type interface	
Note 2:	During reset and whenever the GR716B is deselected by SPIS_SLV, SPIS_MISO will be tristated by the GR716B. An external pull-up or pull-down resistor is needed to keep SPIS_MISO from floating in this state.	

2.7.5 External pin configuration for CAN-FD (CANOpen) remote access

This section describes valid bootstrap configuration for CAN-FD remote access. Refer to section 26 for details on the communication protocol.

Table 15. Remote SPI access pin configurations

Pin Name	Interface Name	Functional description
GPIO[58]	CAN_TX0	CAN-FD Transmitter 0 output. External pull-up resistor recommended.
GPIO[59]	CAN_RX0	CAN-FD Reciever 0 input
GPIO[60]	CAN_SEL0	CAN-FD Output 0 select. External pull-up or pull-down is recommended.
GPIO[61]	CAN_TX1	CAN-FD Transmitter 1 output. External pull-up resistor recommended.
GPIO[62]	CAN_RX1	CAN-FD Reciever 1 input
GPIO[63]	CAN_SEL1	CAN-FD Output 1 select. External pull-up or pull-down is recommended.

Note 1: Interface uses CMOS type interface and requires external CAN transceivers for connection to a CAN bus.

Note 2: CAN FD (CANOpen) parameters must be configured: (TBD)

Note 3: The GR716B GPIOs are tristated during reset. The bootstrap option only takes effect after reset has been deasserted. To ensure that CAN_TX0 and CAN_TX1 are high (to signal a recessive state to the transceiver), external pull-up resistors are required. If they are allowed the float, the CAN transceiver(s) may output a dominant state when the GR716B is in reset, preventing any communication on the shared bus by other connected nodes.

2.7.6 External pin configuration for external SPI boot memory

This section describes valid bootstrap configuration for external SPI memory. Refer to section 46 for a detailed description of the SPI memory controllers.

Table 16. SPI memory pin configurations

Pin Name	Interface Name	Functional description
SPIM_MOSI	SPIM_MOSI	SPI Memory master output slave input
SPIM_SCK	SPIM_SCK	SPI Memory master clock output
SPIM_SEL	SPIM_SEL	SPI Memory slave select output
SPIM_MISO	SPIM_MISO	SPI Memory master input slave output
GPIO[2]	SPIM_MOSI1	Redundant SPI Memory master output slave input
GPIO[1]	SPIM_SCK1	Redundant SPI Memory master clock output
GPIO[0]	SPIM_SEL1	Redundant SPI Memory slave select output
GPIO[3]	SPIM_MISO1	Redundant SPI Memory master input slave output

Note 1: Interface uses CMOS type interface

Note 2: SPIM_MOSI, SPIM_SCK, and SPIM_SEL are bootstrap pins (section 3.1) and must have external pull-resistors to configure the boot mode. SPIM_MISO and SPIM_MISO1 are inputs to the GR716B and since SPI memories typically tristate their data outputs when deselected it is recommended to use external pull-up or pull-down resistor to prevent floating CMOS inputs. For SPIM_MOSI1, SPIM_SCK1, and SPIM_SEL1 external pull-ups are recommended to prevent them from floating during reset. The GR716B GPIO pins are tristated during reset. The bootstrap configuration only takes effect after reset has deasserted.

2.7.7 External pin configuration for external SRAM boot memory

This section describes valid bootstrap configuration for external SRAM. Refer to chapter 22 for a detailed description of the SRAM memory controller.

Table 17. SRAM memory pin configurations

Pin Name	Interface Name	Functional description
GPIO[0]	MEM0_ADDR[0]	Memory address interface
GPIO[1]	MEM0_ADDR[1]	
GPIO[2]	MEM0_ADDR[2]	
GPIO[3]	MEM0_ADDR[3]	
GPIO[4]	MEM0_ADDR[4]	
GPIO[5]	MEM0_ADDR[5]	
GPIO[6]	MEM0_ADDR[6]	
GPIO[7]	MEM0_ADDR[7]	
GPIO[8]	MEM0_ADDR[8]	
GPIO[9]	MEM0_ADDR[9]	
GPIO[10]	MEM0_ADDR[10]	
GPIO[11]	MEM0_ADDR[11]	
GPIO[12]	MEM0_ADDR[12]	
GPIO[13]	MEM0_ADDR[13]	
GPIO[14]	MEM0_ADDR[14]	
GPIO[15]	MEM0_ADDR[15]	
GPIO[16]	MEM0_ADDR[16]	
GPIO[17]	MEM0_ADDR[17]	
GPIO[18]	MEM0_ADDR[18]	
GPIO[33]	MEM0_OEN	Output interface. Pull-up resistor on PCB is recommended.
GPIO[34]	MEM0_WRN	Writen enable interface. Pull-up resistor on PCB is recommended.
GPIO[25]	MEM0_DATA[0]	Bidirectional Data interface Pull-up or pull-down resistors should be used to prevent the bus from floating when the memory is not accessed. Internal weak pull-up or pull-down can be enabled from software after boot and may be sufficient if leakage currents on the bus are not too large.
GPIO[26]	MEM0_DATA[1]	
GPIO[27]	MEM0_DATA[2]	
GPIO[28]	MEM0_DATA[3]	
GPIO[29]	MEM0_DATA[4]	
GPIO[30]	MEM0_DATA[5]	
GPIO[31]	MEM0_DATA[6]	
GPIO[32]	MEM0_DATA[7]	
GPIO[19]	RAM0_CSN[0]	Chip Select. Pull-up resistor on PCB is recommended.
GPIO[20]	RAM0_CSN[1]	Redundant Chip Select. Pull-up resistor on PCB is recommended.

Note 1: Interface uses CMOS type interface

Note 2: MEM0_ADDR[22:19] remain configured as GPIO with this bootstrap option until reconfigured by software. If MEM0_ADDR[22:19] are used in the boot memory interface they need external pull-down resistors to keep them in a well-defined state during the first stage of the boot.

Note 3: When used, MEM0_OEN, MEM0_WRN, RAM0_CSN[0] and RAM0_CSN[1] should have pull-up resistors on PCB to ensure they have a valid deselected state during reset. The GR716B GPIO pins are tristated during reset. The bootstrap option only takes effect after reset has deasserted.

2.7.8 External pin configuration for external PROM/FLASH boot memory

This section describes valid bootstrap configuration for external PROM/FLASH. Refer to chapter 22 for a detailed description of the PROM/FLASH memory controller.

Table 18. PROM/FLASH memory pin configurations

Pin Name	Interface Name	Functional description
GPIO[0]	MEM0_ADDR[0]	Memory address interface
GPIO[1]	MEM0_ADDR[1]	
GPIO[2]	MEM0_ADDR[2]	
GPIO[3]	MEM0_ADDR[3]	
GPIO[4]	MEM0_ADDR[4]	
GPIO[5]	MEM0_ADDR[5]	
GPIO[6]	MEM0_ADDR[6]	
GPIO[7]	MEM0_ADDR[7]	
GPIO[8]	MEM0_ADDR[8]	
GPIO[9]	MEM0_ADDR[9]	
GPIO[10]	MEM0_ADDR[10]	
GPIO[11]	MEM0_ADDR[11]	
GPIO[12]	MEM0_ADDR[12]	
GPIO[13]	MEM0_ADDR[13]	
GPIO[14]	MEM0_ADDR[14]	
GPIO[15]	MEM0_ADDR[15]	
GPIO[16]	MEM0_ADDR[16]	
GPIO[17]	MEM0_ADDR[17]	
GPIO[18]	MEM0_ADDR[18]	
GPIO[33]	MEM0_OEN	Output interface. Pull-up resistor on PCB is recommended.
GPIO[34]	MEM0_WRN	Writen enable interface. Pull-up resistor on PCB is recommended.
GPIO[25]	MEM0_DATA[0]	Data interface Pull-up or pull-down resistors should be used to prevent the bus from floating when the memory is not accessed. Internal weak pull-up or pull-down can be enabled from software after boot and may be sufficient if leakage currents on the bus are not too large.
GPIO[26]	MEM0_DATA[1]	
GPIO[27]	MEM0_DATA[2]	
GPIO[28]	MEM0_DATA[3]	
GPIO[29]	MEM0_DATA[4]	
GPIO[30]	MEM0_DATA[5]	
GPIO[31]	MEM0_DATA[6]	
GPIO[32]	MEM0_DATA[7]	
GPIO[35]	ROM0_CSN[0]	Chip Select. Pull-up resistor on PCB is recommended.
GPIO[36]	ROM0_CSN[1]	Redundant Chip Select. Pull-up resistor on PCB is recommended.

Note 1: Interface uses CMOS type interface

Note 2: MEM0_ADDR[22:19] remain configured as GPIO with this bootstrap option until reconfigured by software. If MEM0_ADDR[22:19] are used in the boot memory interface they need external pull-down resistors to keep them in a well-defined state during the first stage of the boot.

Note 3: When used, MEM0_OEN, MEM0_WRN, ROM0_CSN[0] and ROM0_CSN[1] should have pull-up resistors on PCB to ensure they have a valid deselected state during reset. The GR716B GPIO pins are tristated during reset. The bootstrap option only takes effect after reset has deasserted.

2.8 I/O switch matrix options, considerations and limitations

This chapter lists options and limitations when using different interfaces in the IO switch.

2.8.1 SPI interfaces

The SPI interface can switch from being a Master to Slave interface and vice versa. In general this is not a problem and adds flexibility to the I/O mux concept except for the ’slave select’ signal. The ’slave select’ signal will change direction when switching from Slave i.e. slave select input signal to Master interface i.e. slave select output. In the worst scenario this can permanently damage the internal driver and receiver. To mitigate this problem the Master Slave select output and Slave Select input has been assigned to different I/Os.

In a situation where the application board only requires the SPI interface to either be Slave or Master the option is given to the designer to assign the extra pin to another system interface.

2.8.2 External Memory interface

The external PROM/SRAM interface occupies many external I/Os due parallel data and address buses. The system should only allocate the number of address and chip-select pins needed for the application. E.g. a system that only requires 256KiB of memory only need to allocate and use 18 external address lines and 1 chip-select i.e. the system can assign 4 pins to another system interface.

There is also a potential saving to make if the functionality ’bus ready’ and ’bus exception’ isn’t used.

2.8.3 GPIO with analog functions

The GR716B has 20 GPIO pins (GPIO37-48 and GPIO51-58) with analog capability (ADC, DAC, ACOMP) in addition to digital input and output capability. Before an analog voltage level is applied to any of these pins, they must be placed in analog mode. Placing a pin in analog mode has two effects. Firstly, the digital output driver is disabled, placing the pin in a high impedance state. Secondly, the digital receiver circuit is disabled. The latter is necessary to prevent large internal currents in the receiver circuit when the voltage is in the transition region between VIL and VIH. A GPIO pin not in analog mode, even if configured as a digital input, will be stressed by long-term exposure to analog voltage level due to this internal current.

Analog-capable GPIO pins can be placed in analog mode by setting their I/O switch matrix configurations to 0x8 (see table Table 7 and section 7.1). For comparators used in APWM_AB or APWM_A applications, the input GPIO can also be configured for leading/trailing edge blanking (LEB/TEB) in the I/O switch matrix. Configuration 0x9 for APWM_AB LEB/TEB, and configuration 0xA for APWM_A LEB. In LEB/TEB mode the GPIO will be in analog mode except for a short duration when the pin is blanked by the APWM unit, which enables the digital output driver with low output logic level, while keeping the input buffer disabled.

When in analog mode, the internal pulldown and pullup resistors will be disabled, regardless of the configuration in the SYS.CFG.PULLDOWN0/1 and SYS.CFG.PULLUP0/1 registers.

GPIO mode, clock gating and reset: TDB

2.8.4 Recommendations for unused GPIO and LVDS pins

GPIO pins that are unused in an application should not be allowed to float unless placed in analog mode (section 2.8.3). If a GPIO pin floats, leakage currents will tend to bring the pin voltage to within the transition region (between VIL and VIH) which causes an internal I/O supply current consumption of order 1 mA per pin if the digital receiver circuitry. For pins with Schmitt trigger input capability in addition to the normal GPIO input circuit, the Schmitt trigger circuit can contribute an additional current of several mA per pin when in this state.

The GR716B Schmitt trigger input circuits cannot be disabled and are always active. The only way to place such a pin a low-consumption state is to ensure a digital low or high voltage level on the pin.

One individual pull resistor per pin can achieve this passively. In applications that seek to minimize component count pull resistors can be avoided either by enabling weak internal pull or configuring the pin as an output from software. However, in this case the pin will still float and have a high internal current consumption during reset until software has performed the configuration.

For pins without Schmitt trigger inputs the same methods can be used. But additionally, such pins have the option of being placed in analog mode.

LVDS: TBD

2.9 Cores

The design is based on the following cores from the GRLIB IP Library:

Table 19. Used IP cores

Core	Function	Vendor	Device
AHB2AHB	Bi-directional AHB/AHB bridge	0x01	0x020
AHBROM	Generic AHB ROM	0x01	0x1B
AHBSTAT	AHB Status Register	0x01	0x052
AHBTRACE	AHB trace buffer	0x01	0x017
AHBUART	Serial/AHB Debug interface	0x01	0x007
APBCTRL	AHB/APB bridge	0x01	0x006
APBUART	8-bit UART with FIFO	0x01	0x00C
DSU3	LEON3 Debug Support Unit	0x01	0x004
FTMCTRL	8/16/32-bit memory controller with EDAC	0x01	0x054
GPTIMER	Modular timer unit with watchdog	0x01	0x038
GR1553B	MIL-STD-1553B / AS15531 interface	0x01	0x04D
GRAPWM	Analog PWM system	0x01	n/a
GRCANFD	CAN Flexible Data Rate Controller	0x01	0x0B5
GRCLKGATE	Clock gating unit	0x01	0x02C
GRDMAC2	DMA Controller with internal AHB/APB bridge	0x01	0x0C0
GRETH	10/100 Ethernet Media Access Controller (MAC)	0x01	0x01D
GRGPIO	General Purpose I/O Port	0x01	0x01A
GRGPREG	General purpose register	0x01	0x087
GRMEMPROT	Memory protection	0x01	0x1F1
GRSPWROUTER	SpaceWire Router switch	0x01	0x03E
I2C2AHB	I2C to AHB bridge	0x01	0x00B
I2CMST	I2C master	0x01	0x028
I2CSLV	I2C slave	0x01	0x03E
IRQ(A)MP	Multiprocessor interrupt controller with AMP extensions	0x01	0x00D
L3STAT	LEON3 statistical unit	0x01	0x098
LEON3FT	LEON3 SPARC V8 32-bit processor	0x01	0x053
LRAM	Local on-chip SRAM with EDAC and AHB interface	0x01	0x0A3
MEMSCRUB	Memory scrubber	0x01	0x057
RSTGEN	Reset generator	N/A	N/A
RTA	Real-Time Accelerator subsystem	0x01	n/a
SPI2AHB	SPI to AHB bridge	0x01	0x05C
SPICTRL	SPI controller	0x01	0x02D
SPIMCTRL	SPI memory controller	0x01	0x045
SPISLAVE	SPI for space slave	0x01	0x0A7

The information in the last two columns is available via plug'n'play information in the system and is used by software to detect peripherals and to initialize software drivers.

Revision 0 of the ASIC has the device ID 0x071610BA. This value can be obtained in software by accessing address 0xFFEFFFF0.

Revision 1 of the ASIC has the device ID 0x071210BA. This value can be obtained in software by accessing address 0xFFEFFFF0. The modification is performed in Revision 1 of the ASIC to allow software to identify the device revision.

2.10 Memory map

The memory map of the internal AHB and APB buses as seen from the processor cores can be seen below.

The column 'RTA Access' in the Memory map table indicates if the AMBA peripheral is accessible by the RTA's.

Table 20. AMBA memory map, as seen from processors

Core		Address range	Area	RTA Access
AHBROM		0x00000000 - 0x000FFFFF	Internal Boot PROM	No
FTMCTRL		0x01000000 - 0x01FFFFFF	External PROM	No
SPIMCTRL0		0x02000000 - 0x03FFFFFF	SPI Memory 0 mapped 3-byte address area	No
SPIMCTRL1		0x04000000 - 0x05FFFFFF	SPI Memory 1 mapped 3-byte address area	No
SPIMCTRL0		0x10000000 - 0x17FFFFFF	SPI Memory 0 mapped 4-byte address area	No
SPIMCTRL1		0x18000000 - 0x1FFFFFFF	SPI Memory 1 mapped 4-byte address area	No
DLRAM		0x30000000 - 0x300FFFFF	Processor local data memory	No
ILRAM		0x31000000 - 0x310FFFFF	Processor local instruction memory	No
FTMCTRL		0x40000000 - 0x4FFFFFFF	External SRAM Memory	No
NVRAM		0x50000000 - 0x50FFFFFF	Reserved space for internal NVRAM	No
RTA0	DLRAM	0x60000000 - 0x60003FFF	Local RTA data memory	Yes
	ILRAM	0x61000000 - 0x61003FFF	Local RTA instruction memory	Yes
	IRQMP	0x62000000 - 0x620000FF	Local interrupt controller	Yes
	GPTIMER	0x62010000 - 0x620100FF	Local Watchdog timer and timer unit	Yes
	DLRAM CFG	0x62020000 - 0x620200FF	Local data memory configuration	Yes
	ILRAM CFG	0x62030000 - 0x620300FF	Local instruction memory configuration	Yes
	STAT	0x62040000 - 0x620400FF	RTA Status and mailbox register	Yes
	RTM	0x62060000 - 0x620600FF	RTA Task Manager 0	Yes
	RTM	0x62070000 - 0x620700FF	RTA Task Manager 1	Yes
	RTM	0x62080000 - 0x620800FF	RTA Task Manager 2	Yes
	AHBSTAT	0x62090000 - 0x620900FF	RTA AHB Status Register for local bus	Yes
RTA1	DLRAM	0x70000000 - 0x70003FFF	Local RTA data memory	Yes
	ILRAM	0x71000000 - 0x71003FFF	Local RTA instruction memory	Yes
	IRQMP	0x72000000 - 0x720000FF	Local interrupt controller	Yes
	GPTIMER	0x72010000 - 0x720100FF	Local Watchdog timer and timer unit	Yes
	DLRAM CFG	0x72020000 - 0x720200FF	Local data memory configuration	Yes
	ILRAM CFG	0x72030000 - 0x720300FF	Local instruction memory configuration	Yes
	STAT	0x72040000 - 0x720400FF	RTA Status and mailbox register	Yes
	RTM	0x72060000 - 0x720600FF	RTA Task Manager 0	Yes
	RTM	0x72070000 - 0x720700FF	RTA Task Manager 1	Yes
	RTM	0x72080000 - 0x720800FF	RTA Task Manager 2	Yes
	AHBSTAT	0x72090000 - 0x720900FF	RTA AHB Status Register for local bus	Yes

Table 20. AMBA memory map, as seen from processors

Core		Address range	Area	RTA Access
APBCTRL0	FTMCTRL	0x80000000 - 0x800000FF	Memory controller with EDAC	No
	DLRAM	0x80001000 - 0x800010FF	On-chip Data memory control registers	No
	IRQAMP	0x80002000 - 0x800023FF	Multi-processor Interrupt Ctrl.	No
	GPTIMER	0x80003000 - 0x800030FF	Modular Timer Unit 0 with Watchdog support	No
	GPTIMER	0x80004000 - 0x800040FF	Modular Timer Unit 1	No
	MEMPROT	0x80005000 - 0x800051FF	Memory Protection Unit for system bus	No
	GRCLKGATE	0x80006000 - 0x800060FF	Clock gating configuration register unit 0	No
	GRCLKGATE	0x80007000 - 0x800070FF	Clock gating configuration register unit 1	No
	GRGPREG	0x80008000 - 0x800080FF	Configuration and test registers	No
	L3STAT	0x80009000 - 0x800093FF	LEON3 Statistics Unit	No
	AHBSTAT	0x8000A000 - 0x8000A0FF	AHB Status Register for DMA AMBA bus	No
	ILRAM	0x8000B000 - 0x8000B0FF	On-chip Instruction memory control registers	No
	GRSPWTDP	0x8000C000 - 0x8000C1FF	CCSDS TDP / SpaceWire I/F	No
	GRGPRBANK	0x8000D000 - 0x8000D0FF	IO Mux configuration register	No
	GRGPREG	0x8000E000 - 0x8000E0FF	Test register and system control register	No
	AHBUART	0x8000F000 - 0x8000F0FF	Slave UART configuration for remote access	No
APBCTRL1	GRSPWROUTER	0x80100000 - 0x801000FF	GRSPWROUTER SpaceWire Router	No
	GR1553B	0x80101000 - 0x801010FF	MIL-STD-1553B Interface	No
	GRCANFD	0x80102000 - 0x801023FF	CANFD Controller with DMA	No
	Not used	0x80103000 - 0x801033FF	-	No
	SPI2AHB	0x80104000 - 0x801040FF	SPI to AHB Bridge	No
	I2C2AHB	0x80105000 - 0x801050FF	I2C to AHB Bridge	No
	GRDMAC2	0x80106000 - 0x801061FF	Stand alone DMA unit 0	No
	GRGPRBANK	0x80108000 - 0x801081FF	LVDS IO configuration registers	No
	GRETH	0x80109000 - 0x801091FF	GRETH 10/100 Ethernet MAC with AHB	No
	MEMPROT	0x8010A000 - 0x8010A0FF	Memory protection for DMA bus	No
	PLL	0x8010B000 - 0x8010B0FF	System clock control register	No
	BO	0x8010C000 - 0x8010C0FF	Brown-Out detection control registers	No
	PLL	0x8010D000 - 0x8010D0FF	PLL control registers	No

Table 20. AMBA memory map, as seen from processors

Core		Address range	Area	RTA Access
	APBUART	0x80300000 - 0x803000FF	Generic UART 0	Yes
	APBUART	0x80301000 - 0x803010FF	Generic UART 1	Yes
	APBUART	0x80302000 - 0x803020FF	Generic UART 2	Yes
A	APBUART	0x80303000 - 0x803030FF	Generic UART 3	Yes
P	APBUART	0x80304000 - 0x803040FF	Generic UART 4	Yes
B	APBUART	0x80305000 - 0x803050FF	Generic UART 5	Yes
C	AHBSTAT	0x80306000 - 0x803060FF	AHB Status Register for MAIN AMBA bus	Yes
T	NVRAM	0x80307000 - 0x803070FF	Memory controller with EDAC (NVRAM)	Yes
R	-	0x80308000 - 0x803080FF	Unused	-
L	SPICTRL	0x80309000 - 0x803090FF	SPI Controller 0	Yes
2	SPICTRL	0x8030A000 - 0x8030A0FF	SPI Controller 1	Yes
	LVDS GPIO	0x8030B000 - 0x8030BFFF	LVDS GPIO Mode Control	Yes
	GRGPIO	0x8030C000 - 0x8030CFFF	General Purpose I/O port 0 to 31	Yes
	GRGPIO	0x8030D000 - 0x8030DFFF	General Purpose I/O port 32 to 64	Yes
	I2CMST	0x8030E000 - 0x8030E0FF	I2C-master 0	Yes
	I2CMST	0x8030F000 - 0x8030F0FF	I2C-master 1	Yes
	ADC	0x80400000 - 0x804000FF	On-chip ADC interface 0	Yes
	ADC	0x80401000 - 0x804010FF	On-chip ADC interface 1	Yes
	ADC	0x80402000 - 0x804020FF	On-chip ADC interface 2	Yes
A	ADC	0x80403000 - 0x804030FF	On-chip ADC interface 3	Yes
P	SCRUBBER	0x80404000 - 0x804040FF	FPGA Scrubber	Yes
B	-	0x80405000 - 0x804050FF	Unused	-
C	-	0x80406000 - 0x804060FF	Unused	-
T	-	0x80407000 - 0x804070FF	Unused	-
R	DAC	0x80408000 - 0x804080FF	On-chip DAC interface 0	Yes
L	DAC	0x80409000 - 0x804090FF	On-chip DAC interface 1	Yes
3	DAC	0x8040A000 - 0x8040A0FF	On-chip DAC interface 2	Yes
	DAC	0x8040B000 - 0x8040B0FF	On-chip DAC interface 3	Yes
	I2CSLV	0x8040C000 - 0x8040C0FF	I2C-slave 0	Yes
	I2CSLV	0x8040D000 - 0x8040D0FF	I2C-slave 1	Yes
	GRSPI4	0x8040F000 - 0x8040F0FF	SPI for Space Slave	Yes

Table 20. AMBA memory map, as seen from processors

Core		Address range	Area	RTA Access
	TIMER32	0x81000000 - 0x810000FF	APWM system timer and synchronization	Yes
	TIMER32	0x81001000 - 0x810010FF	APWM system timer and synchronization	Yes
	REGSYNC	0x81002000 - 0x810020FF	APWM Register synchronization	Yes
A	PROTSHDN	0x81003000 - 0x810030FF	Protection shutdown A	Yes
P	PROTSHDN	0x81004000 - 0x810040FF	Protection shutdown B	Yes
B	PROTSHDN	0x81005000 - 0x810050FF	Protection shutdown C	Yes
C	SYNC	0x81006000 - 0x810060FF	External synchronization	Yes
T	ACOMP	0x81007000 - 0x810070FF	Analog comparators ACOMP 0-11	Yes
R	ACOMP	0x81008000 - 0x810080FF	Analog comparators ACOMP12-19	Yes
L	APWMDAC	0x81009000 - 0x810090FF	PWM modulator DAC 0	Yes
4	APWMDAC	0x8100A000 - 0x8100A0FF	PWM modulator DAC 1	Yes
	APWMDAC	0x8100B000 - 0x8100B0FF	PWM modulator DAC 2	Yes
	APWMDAC	0x8100C000 - 0x8100C0FF	PWM modulator DAC 3	Yes
	CLKDET	0x8100D000 - 0x8100D0FF	Clock error detection 0	Yes
	TIMER27	0x8100E000 - 0x8100E0FF	APWM function timer and synchronization	Yes
	TIMER27	0x8100F000 - 0x8100F0FF	APWM function timer and synchronization	Yes
	REGSYNCA	0x81100000 - 0x811000FF	APWM A0 Timer and synchronization	Yes
	APWMA	0x81101000 - 0x811010FF	APWM A0 0	Yes
	APWMA	0x81102000 - 0x811020FF	APWM A0 1	Yes
A	APWMA	0x81103000 - 0x811030FF	APWM A0 2	Yes
P	APWMA	0x81104000 - 0x811040FF	APWM A0 3	Yes
B	Not used	0x81105000 - 0x811050FF	-	Yes
C	Not used	0x81106000 - 0x811060FF	-	Yes
T	Not used	0x81107000 - 0x811070FF	-	Yes
R	Not used	0x81108000 - 0x811080FF	-	Yes
L	Not used	0x81109000 - 0x811090FF	-	Yes
5	FIRCLKGEN	0x8110A000 - 0x8110A0FF	FIR filter clock and synchronization	Yes
	CLOCKDET	0x8110B000 - 0x8110B0FF	Clock error detection 1	Yes
	TIMESTAMP	0x8110C000 - 0x8110C0FF	APWM Timestamp	Yes
	Not used	0x8110D000 - 0x8110D0FF	-	Yes
	Not used	0x8110E000 - 0x8110E0FF	-	Yes
	Not used	0x8110F000 - 0x8110F0FF	-	Yes

Table 20. AMBA memory map, as seen from processors

Core		Address range	Area	RTA Access
	REGSYNCAB	0x81200000 - 0x812000FF	APWM AB0 Timer and synchronization	Yes
	APWMAB	0x81201000 - 0x812010FF	APWM AB0	Yes
	REGSYNCAB	0x81202000 - 0x812020FF	APWM AB1 Timer and synchronization	Yes
A	APWMAB	0x81203000 - 0x812030FF	APWM AB1	Yes
P	REGSYNCAB	0x81204000 - 0x812040FF	APWM AB2 Timer and synchronization	Yes
B	APWMAB	0x81205000 - 0x812050FF	APWM AB2	Yes
C	REGSYNCAB	0x81206000 - 0x812060FF	APWM AB3 Timer and synchronization	Yes
T	APWMAB0	0x81207000 - 0x812070FF	APWM AB3	Yes
R	FIR	0x81208000 - 0x812080FF	FIR filter 0	Yes
L	FIR	0x81209000 - 0x812090FF	FIR filter 1	Yes
6	FIR	0x8120A000 - 0x8120A0FF	FIR filter 2	Yes
	FIR	0x8120B000 - 0x8120B0FF	FIR filter 3	Yes
	FIR	0x8120C000 - 0x8120C0FF	FIR filter 4	Yes
	FIR	0x8120D000 - 0x8120D0FF	FIR filter 5	Yes
	FIR	0x8120E000 - 0x8120E0FF	FIR filter 6	Yes
	FIR	0x8120F000 - 0x8120F0FF	FIR filter 7	Yes
	REGSYNCCF	0x81300000 - 0x813000FF	APWM CF0 Timer and synchronization	Yes
	APWMCF	0x81301000 - 0x813010FF	APWM CF0 0	Yes
	APWMCF	0x81302000 - 0x813020FF	APWM CF0 1	Yes
A	APWMCF	0x81303000 - 0x813030FF	APWM CF0 2	Yes
P	APWMCF	0x81304000 - 0x813040FF	APWM CF0 3	Yes
B	REGSYNCCF	0x81305000 - 0x813050FF	APWM CF1 Timer and synchronization	Yes
C	APWMCF	0x81306000 - 0x813060FF	APWM CF1 0	Yes
T	APWMCF	0x81307000 - 0x813070FF	APWM CF1 1	Yes
R	APWMCF	0x81308000 - 0x813080FF	APWM CF1 2	Yes
L	APWMCF	0x81309000 - 0x813090FF	APWM CF1 3	Yes
7	Not used	0x8130A000 - 0x8130A0FF	-	Yes
	Not used	0x8130B000 - 0x8130B0FF	-	Yes
	Not used	0x8130C000 - 0x8130C0FF	-	Yes
	Not used	0x8130D000 - 0x8130D0FF	-	Yes
	Not used	0x8130E000 - 0x8130E0FF	-	Yes
	Not used	0x8130F000 - 0x8130F0FF	-	Yes

Table 20. AMBA memory map, as seen from processors

Core		Address range	Area	RTA Access
	REGSYNCCF	0x81400000 - 0x814000FF	APWM CF2 Timer and synchronization	Yes
	APWMCF	0x81401000 - 0x814010FF	APWM CF2 0	Yes
	APWMCF	0x81402000 - 0x814020FF	APWM CF2 1	Yes
A	APWMCF	0x81403000 - 0x814030FF	APWM CF2 2	Yes
P	APWMCF	0x81404000 - 0x814040FF	APWM CF2 3	Yes
B	Not used	0x81405000 - 0x814050FF	-	Yes
C	Not used	0x81406000 - 0x814060FF	-	Yes
T	Not used	0x81407000 - 0x814070FF	-	Yes
R	Not used	0x81408000 - 0x814080FF	-	Yes
L	Not used	0x81409000 - 0x814090FF	-	Yes
8	Not used	0x8140A000 - 0x8140A0FF	-	Yes
	Not used	0x8140B000 - 0x8140B0FF	-	Yes
	Not used	0x8140C000 - 0x8140C0FF	-	Yes
	Not used	0x8140D000 - 0x8140D0FF	-	Yes
	Not used	0x8140E000 - 0x8140E0FF	-	Yes
	Not used	0x8140F000 - 0x8140F0FF	-	Yes
	APWMG	0x81500000 - 0x815000FF	APWMG 0 (with One Shot Timer)	Yes
	APWMG	0x81501000 - 0x815010FF	APWMG 1 (with One Shot Timer)	Yes
	APWMG	0x81502000 - 0x815020FF	APWMG 2 (with One Shot Timer)	Yes
A	APWMG	0x81503000 - 0x815030FF	APWMG 3 (with One Shot Timer)	Yes
P	APWMG	0x81504000 - 0x815040FF	APWMG 4	Yes
B	APWMG	0x81505000 - 0x815050FF	APWMG 5	Yes
C	APWMG	0x81506000 - 0x815060FF	APWMG 6	Yes
T	APWMG	0x81507000 - 0x815070FF	APWMG 7	Yes
R	APWMG	0x81508000 - 0x815080FF	APWMG 8	Yes
L	APWMG	0x81509000 - 0x815090FF	APWMG 9	Yes
9	APWMG	0x8150A000 - 0x8150A0FF	APWMG 10	Yes
	APWMG	0x8150B000 - 0x8150B0FF	APWMG 11	Yes
	APWMG	0x8150C000 - 0x8150C0FF	APWMG 12	Yes
	APWMG	0x8150D000 - 0x8150D0FF	APWMG 13	Yes
	APWMG	0x8150E000 - 0x8150E0FF	APWMG 14	Yes
	APWMG	0x8150F000 - 0x8150F0FF	APWMG 15	Yes

Table 20. AMBA memory map, as seen from processors

Core		Address range	Area	RTA Access
APBCTRL DBG	AHBUART	0x94000000 - 0x940000FF	AHB Debug UART	No
	L3STAT	0x94001000 - 0x940013FF	LEON3 Statistics Unit	No
	GRGPREG	0x94002000 - 0x940020FF	Analog test control	No
	GRGPREG	0x94003000 - 0x94003FFF	Memory test control and status register	No
	GRGPREG	0x94004000 - 0x94004FFF	Analog test control	No
	GRGPREG	0x94005000 - 0x94005FFF	Analog test control	No
	GRGPREG	0x94006000 - 0x94006FFF	Debug configuration register 1	No
	GRGPREG	0x94007000 - 0x94007FFF	Debug configuration register 2	No
	-	0x94008000 - 0x94008FFF	Unused	-
	-	0x94009000 - 0x94009FFF	Unused	-
	-	0x9400A000 - 0x9400AFFF	Unused	-
	-	0x9400B000 - 0x9400BFFF	Unused	-
	-	0x9400C000 - 0x9400CFFF	Unused	-
	-	0x9400D000 - 0x9400DFFF	Unused	-
	-	0x9400E000 - 0x9400EFFF	Unused	-
	-	0x9400F000 - 0x9400FFFF	Unused	-
DSU3		0x90000000 - 0x905FFFFF	LEON3 Debug Support Unit	No
DSU3 RTA 0		0x96000000 - 0x96FFFFFF	LEON3 Debug Support Unit for RTA 0	No
DSU3 RTA 1		0x98000000 - 0x98FFFFFF	LEON3 Debug Support Unit for RTA 1	No
-		0x94000000 - 0x940FFFFF	APB bus DBG address space	No
AHBTRACE1		0x9ff20000 - 0x9ff3FFFF	AHB Trace Buffer	No
-		0x9FFFF000 - 0x9FFFFFFF	Configuration area for Debug bus	No
MEMSCRUB		0xFFF00000 - 0xFFF000FF	Memory scrubber registers	No
SPIMCTRL		0xFFF00100 - 0xFFF001FF	SPIMCTRL control registers 0	No
SPIMCTRL		0xFFF00200 - 0xFFF002FF	SPIMCTRL control registers 1	No
-		0xFFFFF000 - 0xFFFFFFFF	Configuration area for system main bus	No

Accesses to unused AMBA AHB address space will result in an AMBA ERROR response, this applies to the memory areas that are marked as "Unused" in the table above. Accesses to unused areas located on one of the AHB/APB bridges will not have any effect, note that these unoccupied address ranges are not marked as "Unused" in the table above. No AMBA ERROR response will be given for memory allocated to one of the APB bridges.

2.11 Atomic access

This chapter describes how atomic read and modify operations are performed in the GR716B microcontroller. The GR716B microcontroller supports atomic read-modify-write operations in hardware by mirroring the address space of the peripheral and internal data memory for different atomic operations. Atomic operations supported are OR, AND, XOR and Set&Clear.

Atomic access is only supported locally in the GPIO functionally in GR716B.

2.12 Interrupts

The table below indicates the interrupt default assignments. All interrupts are handled by interrupt controller and forwarded to LEON3FT processors. For more configuration and option see chapter 40.

Table 21. Bus Interrupt line assignments.

Interrupt ID	Interrupt Line	Core	Comment
	0	n/a	not used
1	1	Extended	Extended Interrupts for primary interrupt controller
2	2	GRPWRX	Interrupt from PacketWire RX controller
3	3	GRPWTX/GRSCRUB	Interrupt line shared between PacketWire TX controller and FPGA Scrubber.
4	4	GR1553B	Interrupt from GR1553 controller
5	5	GRSPWROUTER	Interrupt from SpaceWire router
6	6	GRDMAC	DMA controller interrupt
7	7	I2CSLV0	Interrupt from I2C Slave 0
8	8	unused	
9	9	GPTIMER0	Interrupt 1 from timer block 0
10	10	GPTIMER0	Interrupt 2 from timer block 0
11	11	GPTIMER0/Timer32_A	Interrupt line shared between interrupt 3 from timer block 0 and APWM Timer32_A
12	12	GPTIMER0/Timer32_B	Interrupt line shared between interrupt 4 from timer block 0 and APWM Timer32_B
13	13	GPTIMER0/RegSYNC	Interrupt line shared between interrupt 5 from timer block 0 and APWM Register synchronization
14	14	GPTIMER0/Timer27_0/ Timer27_1/PROTA	Interrupt line shared between interrupt 6 from timer block 0, APWM Timer27_0, APWM Timer27_1 and APWM Protection A block
15	15	GPTIMER0/PROTB	Interrupt line shared between interrupt 7 from timer block 0 (WDOG) and APWM Protection B block
16	16	PROTC	Interrupt from APWM Protection C block
17	17	GRGPIO0/SYNC	Interrupt from GPIO controller 0 Synchronization block
18	18	GRGPIO0/REGSYNCA	Interrupt line shared between GPIO controller 0 and APWMA Timer block
19	19	GRGPIO0/APWMA0	Interrupt line shared between GPIO controller 0 and APWMA0 block
20	20	GRGPIO0/APWMA1	Interrupt line shared between GPIO controller 0 and APWMA1 block
21	21	GRCANFD/APWMA2	Interrupt line shared between CANFD controller and APW-MA2 block
22	22	GRCANFD/APWMA3	Interrupt line shared between TxSYNC CANFD controller and APWMA3 block
23	23	GRCANFD/TIME-STAMP	Interrupt line shared between RxSYNC CANFD controller and APWM Timestamp
24	24	APBUART0/REGSYN-CAB0	Interrupt line shared between APBUART interface 0 and APWMAB0 Timer and synchronization
25	25	APBUART1/APWMAB0	Interrupt line shared between APBUART interface 1 and APWMAB0
26	26	DAC0/REGSYNCAB1	Interrupt line shared between on-chip DAC 0 and APWMAB1 Timer and synchronization
27	27	DAC1/APWMAB1	Interrupt line shared between on-chip DAC 1 and APWMAB1

Table 21. Bus Interrupt line assignments.

Interrupt ID	Interrupt Line	Core	Comment
28	28	ADC0 buffer 0/REGSYNCAB2	Interrupt line shared between on-chip ADC 0 sampling buffer 0 and APWMAB2 Timer and synchronization
29	29	ADC1 buffer 0/ADC0 buffer 1/APWMAB2	Interrupt line shared between on-chip ADC 1 sampling buffer 0, ADC 0 sampling buffer 1 and APWMAB2
30	30	ADC2 buffer 0/ADC1 buffer 1/ADC0 buffer 2/ REGSYNCAB3	Interrupt line shared between on-chip ADC 2 sampling buffer 0, ADC1 sampling buffer 1, ADC0 sampling buffer 2 and APWMAB3 Timer and synchronization
31	31	ADC3 buffer 0/ADC2 buffer 1/ADC1 buffer 2/ ADC0 buffer 3/ APWMAB3	Interrupt line shared between on-chip ADC 3 sampling buffer 0, ADC 2 sampling buffer 1, ADC 1 sampling buffer 2, ADC 0 sampling buffer 3 and APWMAB3
28	32	REGSYNCCF0/ADC3 buffer 1/ADC 2 buffer 2/ ADC 1 buffer 3	Interrupt line shared between APWMCF0 Timer and synchronization, ADC3 sampling buffer 1, ADC2 sampling buffer 2, and ADC1 sampling buffer 3
29	33	APWMCF0 0/FIRCCLKGEN/ADC3 buffer 2/ ADC2 buffer 3	Interrupt line shared between APWMCF0 0, FIR filter clock and synchronization, ADC 3 sampling buffer 2, and ADC 2 sampling buffer 3
30	34	APWMCF0 1/FIR0/ ADC3 buffer 3	Interrupt line shared between APWMCF0 1, FIR filter 0, and ADC 3 sampling buffer 3
31	35	APWMCF0 2/FIR1/ ADC3 buffer 4	Interrupt line shared between APWMCF0 2 and FIR filter 1, and ADC 3 sampling buffer 4
26	36	DAC2/FIR2/APWMCF0 3/ADC3 buffer 5	Interrupt line shared between on-chip DAC 2 interrupt, FIR filter 2 and APWMCF0 3, and ADC 3 sampling buffer 5
27	37	DAC3/FIR3/REGSYNCCF1/ADC3 buffer 6	Interrupt line shared between on-chip DAC 3 interrupt, FIR filter 3 and APWMCF1 Timer and synchronization, and ADC 3 sampling buffer 6
16	38	GRGPIO1/FIR4/APWMCF1 0/ADC3 buffer 7	Interrupt line shared between GPIO controller 1, FIR filter 4 and APWMCF1 0, and ADC 3 sampling buffer 7
17	39	GRGPIO1/FIR5/APWMCF1 1	Interrupt line shared between GPIO controller 1, FIR filter 5 and APWMCF1 1
18	40	GRGPIO1/FIR6/APWMCF1 2	Interrupt line shared between GPIO controller 1, FIR filter 6 and APWMCF1 2
19	41	GRGPIO1/FIR7/APWMCF1 3	Interrupt line shared between GPIO controller 1, FIR filter 7 and APWMCF1 3
3	42	APBUART2/REGSYNCCF2/CLOCKDET 0/ CLOCKDET 1	Interrupt line shared between APBUART interface interrupt 2, APWMCF2 Timer/synchronization and Clock error detection unit 0/1
4	43	GRSPWTDP/APWMDAC 0/APWMCF2 0/ APWMG* 0	Interrupt line shared between SpaceWire TDP, PWM modulator DAC 0, APWMCF2 0 and APWMG* 0
5	44	APBUART3/APWMDAC 1/APWMCF2 1/ APWMG* 1	Interrupt line shared between APBUART interface interrupt 3,PWM modulator DAC 1, APWMCF2 1 and APWMG* 1
6	45	APBUART4/APWMDAC 2/APWMCF2 2/ APWMG* 2	Interrupt line shared between APBUART interface interrupt 4, PWM modulator DAC 2, APWMCF2 2 and APWMG* 2
7	46	APBUART5/APWMDAC 3/APWMCF2 3/ APWMG* 3	Interrupt line shared between APBUART interface interrupt 5, PWM modulator DAC 3, APWMCF2 3 and APWMG* 3
8	47	I2CSLV1/I2C2AHB / ACOMP 0-11/APWMG0	Interrupt line shared between Interrupt line shared between I2C Slave Interface 1, I2C2 to AHB bridge, ACOMP 0-11 interrupt and APWMG 0
11	48	SPICTRL0/APWMG1	Interrupt line shared between SPI controller 0 and APWMG 1
12	49	SPICTRL1/APWMG2	Interrupt line shared between SPI controller 1 and APWMG 2

Table 21. Bus Interrupt line assignments.

Interrupt ID	Interrupt Line	Core	Comment
13	50	I2CM0/APWMG3	Interrupt line shared between I2C master controller 0 and APWMG 3
14	51	I2CM1/ ACOMP12-19/ APWMG4	Interrupt line shared between I2C master controller 1, ACOMP 12-19 interrupt and APWMG 4
2	52	SPIMCTRL0/ SPIMCTRL1/APWMG5	Interrupt line shared between SPI memory controllers 0, 1 and APWMG 5
20	53	GPTIMER1/APWMG6	Interrupt line shared between interrupt 1 from timer block 1 and APWMG 6
21	54	GPTIMER1/APWMG7	Interrupt line shared between interrupt 2 from timer block 1 and APWMG 7
22	55	GPTIMER1/APWMG8	Interrupt line shared between interrupt 3 from timer block 1 and APWMG 8
23	56	GPTIMER1/APWMG9	Interrupt line shared between interrupt 4 from timer block 1 and APWMG 9
24	57	GPTIMER1/APWMG10	Interrupt line shared between interrupt 5 from timer block 1 and APWMG 10
25	58	GPTIMER1/APWMG11	Interrupt line shared between interrupt 6 from timer block 1 and APWMG 11
26	59	GPTIMER1	Interrupt line shared between interrupt 7 from timer block 1
16	60	RTA0	Interrupt line shared between RTA sequencer 0
17	61	RTA1 / GRETH	Interrupt line shared between RTA sequencer 1 and Ethernet MAC
18	62	SPI2AHB / SPI4S	Interrupt line shared between SPI to AHB bridge and SPI for space slave controller
19	63	PLL/BO/AHBSTAT/ DLRAM, ILRAM/GRG-PRBANK/MEMSCRUB/ MEMPROT/LVDSIO	Interrupt line shared between PLL, Brownout detector, AHB status, Scrubbers, LVDS as General Purpose IO and I/O mux interrupt.

Note: Interrupt ID is the default mapping of the Interrupt line after device reset.

3 Signals

3.1 Bootstrap signals

The power-up and initialisation state is affected by several external signals as shown in table 22.The bootstrap signals taken via GPIO, DUART and SPIM signals are saved when the on-chip system reset is released. This occurs after deassertion of the internal power-on-reset or RESET_IN_N input and valid input clock on the SYS_CLK pin. The state of the signals are sampled and stored in a bootstrap register. See section 7.2 for boot strap register description.

Note that some pins used for bootstrapping have dual purpose can be used for normal operations after reset has been released.

Table 22. Bootstrap signals

Pin	Functional description
DSU_EN	Enables the Debug Support Unit (DSU) and other members connected to the Debug AHB bus. If DSU_EN is HIGH the DSU and the Debug AHB bus will be clocked. If DSU_EN is LOW the DSU and all members on the Debug AHB bus will be clock gated off.
DSU_BREAK	Puts processor in debug mode when asserted while DSU_EN is HIGH. When DSU_EN is LOW, BREAK is assigned to the timer enable bit of the watchdog timer and also controls if the processor starts executing after reset.
GPIO[17]	Enable bypass of internal boot ROM. Boot strapping this signal 'high' will force the processor NOT to execute the internal boot software. Normally the processor starts executing from address 0x0. But if this bootstrap is 'high' the processor will start execute from software from address selected by bootstrap signals SPIM_MOSI & SPIM_SCK & SPIM_SEL.
GPIO[0]	Determines the use of EDAC for external boot RAM when the GR716B microcontroller shall boot from external memory. Set to low for enabling EDAC and to high for disabling EDAC. Determines the use of PLL when the GR716B microcontroller shall boot via a remote source. Set to high for enabling PLL and to low for disabling PLL. Determines bit 0 of the I2C node ID (ID[0]). When using CAN-FD remote boot PLL must be disabled. CAN-FD node ID[0] is fixed to ‘0’.
GPIO[1]	Determines the selection of nominal or redundant bus for CAN-FD. Set to low for nominal bus, and high for redundant bus.
GPIO[15]	Determines bit 2 of the CAN-FD and I2C node ID (ID[2]).
GPIO[18]	When remote access is enabled (SPIM_MOSI is high), GPIO[18] enables/disables CAN-FD remote boot. Set high to enable, low to disable.
GPIO[62]	Determines bit 1 of the CAN-FD and I2C node ID (ID[1]).
GPIO[63]	Enables extra protection of external boot source or setting SpaceWire clock frequency. If boot from external RAM/ROM this pin enable the use of redundant memory if primary boot memory fails. High to enable, low to disable. If remote access via SPW this pin together with DUART_TXD is used to set the SpaceWire default speed. If remote access via CAN-FD this pin together with DUART_TXD is used to set the CAN default bit rate.

Table 22. Bootstrap signals

Pin	Functional description
DUART_TXD	If boot from external SRAM/ROM/SPI-ROM this pin are used for selecting to copy ASW image from selected external boot RAM/ROM (If not set for this option. The GR716B microcontroller will start execute from the selected external memory) If remote access via SPW is selected then this pin together with GPIO[63] is used to set the SPW default speed. Set DUART_TXD & GPIO[63] accordingly depending on external SpaceWire frequency: "00" - For 50 MHz external frequency source "01" - For 10 MHz external frequency source "10" - For 20 MHz external frequency source "11" - For 25 MHz external frequency source If remote access via CAN-FD is selected then this pin together with GPIO[63] is used to set the Bit-Rate Configuration Register of CAN-FD adapted to the internal AMBA clock frequency. Set DUART_TXD & GPIO[63] accordingly depending on internal AMBA clock frequency: "00" - For 10 MHz internal AMBA clock frequency "01" - For 20 MHz internal AMBA clock frequency "10" - For 25 MHz internal AMBA clock frequency "11" - For 50 MHz internal AMBA clock frequency The CAN-FD bit-rate when remote boot is enabled is fixed to 0.125 Mbit/s.
SPIM_MOSI	Enable remote access (high to enable, low to disable). When remote access is disabled, the processor will either (depending on DUART_TXD) start or copy a software image from the selected external boot memory. When remote access is enabled, the processor will (after initialization) be placed in a halted state but remote access to the internal bus will be enabled through SPI, SpaceWire RMAP, I2C, UART or CAN-FD.
SPIM_SCK & SPIM_SEL	This pin together with the pin SPIM_MOSI selects which source the LEON3FT microcontroller should boot from: When copy ASW boot from external source is selected (SPIM_MOSI is low) "00" - Copy software image from SPI Memory "01" - Copy software image from external SRAM "10" - Copy software image from external ROM "11" - Unused (Will result in copy of software image from SPI Memory) When boot from external source is selected (SPIM_MOSI is low) "00" - Boot from SPI Memory "01" - Boot from external SRAM "10" - Boot from external ROM "11" - Unused (Will result in boot from SPI Memory) Enable for remote access interfaces (SPIM_MOSI is high) "00" - SPI remote access "01" - SpaceWire RMAP enable "10" - I2C remote access "11" - UART remote access Dedicated pin GPIO[18] available for CAN-FD boot.
MEM1_CONFIG(0)	Boot enable, GR716B boots using the second memory controller with dedicated memory interface (memory interfaces not shared with the GPIO). Drive 'Low' to enable boot, 'High' to disable. Available only for PBGA and SIP devices.
MEM1_CONFIG(1)	Memory enable, enable the second memory controller. Drive 'Low' to enable memory interface, 'High' to disable. Available only for PBGA and SIP devices.

Table 22. Bootstrap signals

Pin	Functional description
MEM1_CONFIG(2)	Drive 'Low' for 16-bit Memory interface. 'High' is for future use. Available only for PBGA and SIP devices.
Note 1:	User should use weak pull-up/pull-downs for configuration of the GR716B microcontroller. A weak resistor is defined as resistor which require low current from the drive circuitry. The resistance should be greater or equal to 10K ohm.
Note 2:	Bootstrap signals determine state of GR716B microcontroller after reset has been released.
Note 3:	The LEON3FT processor is always enabled after reset has been released.
Note 4:	Remote access request will force the processor to power down according to 17.2.16 after initialization has been completed.
Note 5:	Remote access will enable clocks according to table: 1. SpaceWire option will enable SpaceWire core and external SpaceWire interface 2. SPI option will enable SPI for Space Slave and external SPI for Space interface 3. I2C option will enable I2C core and external I2C interface 4. UART option will enable AHBUART1 and external UART interface 5. CAN-FD option will enable CAN-FD core and external CAN interface
Note 6:	Only requested memory interface will have clock and pins enabled.
Note 7:	Watchdog timer will always be enabled and not controllable from bootstraps.

3.1.1 Boot strap configuration for remote access

This section describes valid bootstrap configuration for remote access:

Table 23. Remote bootstrap configurations

Pin								
Remote Access	Source [7]			SpaceWire Divisor		**PLL** [4] [5]	Boot Bypass	
SPIM_MOSI	SPIM_SCK	SPIM_SEL	GPIO[18]	DUART_TXD	GPIO[63]	GPIO[0]	GPIO[17]	**Functional description**
high	low	high	low	low	low	high	low	Enable SpaceWire remote access using a 50 MHz SPWCLK input clock after initialization of the processor.
high	low	high	low	low	high	high	low	Enable SpaceWire remote access using a 10 MHz SPWCLK input clock after initialization of the processor
high	low	high	low	high	low	high	low	Enable SpaceWire remote access using a 20 MHz SPWCLK input clock after initialization of the processor
high	low	high	low	high	high	high	low	Enable SpaceWire remote access using a 25 MHz SPWCLK input clock after initialization of the processor
high	low	high	low	x [2]	x [2]	low	low	Enable SpaceWire remote access using the clock direct from the SpaceWire input pin after initialization of the processor
high	low	low	low	x [2]	x [2]	x [2]	low	Enable SPI remote access after initialization of the processor.
high	high	low	low	x [2]	x [2]	x [2]	low	Enable I2C remote access after initialization of the processor.
high	high	high	low	x [2]	x [2]	x [2]	low	Enable UART remote access after initialization of the processor.
high	low	low	high	low	low	low	low	Enable CAN-FD CANOPEN remote access using a 10 MHz SYS_CLK after initialization of the processor.
high	low	low	high	low	high	low	low	Enable CAN-FD CANOPEN remote access using a 20 MHz SYS_CLK after initialization of the processor.

Table 23. Remote bootstrap configurations

Pin								
Remote Access	Source [7]			SpaceWire Divisor		**PLL [4] [5]**	Boot Bypass	
SPIM_MOSI	SPIM_SCK	SPIM_SEL	GPIO[18]	DUART_TXD	GPIO[63]	GPIO[0]	GPIO[17]	**Functional description**
high	low	low	high	high	low	low	low	Enable CAN-FD CANOPEN remote access using a 25 MHz SYS_CLK after initialization of the processor.
high	low	low	high	high	high	low	low	Enable CAN-FD CANOPEN remote access using a 50 MHz SYS_CLK after initialization of the processor.

Note 1: To enable remote access SPIM_MOSI must be bootstrapped to high.

Note 2: Configuration pin has no effect or not used for bootstrap configuration. It recommend to tie the pin to either low or high.

Note 3: Processor are forced into power down mode after processor and memory test has been completed.

Note 4: Enable internal PLL by bootstrap signal to high. When using the internal PLL the link speed will be set to 10 Mbps after reset and maximum link speed after auto negotiation of link speed is 100 Mbps.

Note 5: Disable internal PLL by bootstrap signal to low. When bypassing the internal PLL the speed will be set to input frequency of the SpaceWire clock. The PLL will be in power down mode.

Note 6: Refer section 3.1 for detailed description of bootstrap signals.

Note 7: When remote access via I2C or CAN-FD is selected, 3 bits of the I2C address and CANOPEN node ID are partially determined by bootstrap pins. The CANOPEN node ID is set to 0b0000*xxx* while I2C memory and configuration addresses are set to 0b101*xxx*0 and 0b101*xxx*1. The bits "*xxx*" are taken from bootstrap pins: GPIO[15] & GPIO[62] & GPIO[0]. (TBD)

Note 8: MEM1_CONFIG(x) pins shall be set to 'High' for PBGA and SIP devices to get remote boot.

3.1.2 Boot strap configuration for external SPI memory

This section describes valid bootstrap configuration for use of external memory options:

Table 24. External SPI memory bootstrap configurations

Pin								
Remote Access	Source			ASW 3)	Red 4)	EDAC 5)	Bypass 6)	
SPIM_ MOSI	SPIM_ SCK	SPIM_ SEL	GPIO[18]	DUART _TXD	GPIO[63]	GPIO[0]	GPIO[17]	**Functional description**
low	low	low	low	low	low	5)	low	Enable external SPI memory boot. Processor will start execute application software direct from memory after initialization of the processor.
low	low	low	low	low	low	5)	high	Enable external SPI memory boot. Processor will start execute application software direct from memory.
low	low	low	low	high	low	5)	low	Enable external ASW SPI memory boot after initialization of the processor. Processor will copy and extract ASW container before executing application software.
low	low	low	low	high	high	5)	low	Enable external ASW SPI memory with DMR protection boot after initialization of the processor. Processor will copy and extract ASW container before executing application software.

Note 1: To enable external memory access SPIM_MOSI must be bootstrapped to low.

Note 2: Configuration pin has no effect or not used for bootstrap configuration. It recommend to tie the pin either to low or high.

Note 3: Enable ASW protection. ASW protection usage is described in section 51.

Note 4: Enable dual module redundancy protection. Option only valid in combination with ASW protection. Redundant memory is expected to located at 0x04000000.

Note 5: Enable BCH EDAC protection. EDAC can be enabled and used in combination with all other options. When configuration is used external memory must included BCH check bits.

Note 6: Enable bypass of the internal boot ROM. When enabled the processor will start execute code directly from the primary memory at 0x02000000. Processor or internal memory is initialized after reset when this option is used.

Note 7: MEM1_CONFIG(x) pins shall be set to ‘High’ for PBGA and SIP devices to get SPI boot.

3.1.3 Boot strap configuration for external SRAM memory

This section describes valid bootstrap configuration for use of external memory options:

Table 25. External SRAM memory bootstrap configurations

Pin							
Remote Access	Source		ASW [3]	Red [4]	**EDAC** [5]	Bypass [6]	
SPIM_MOSI GPIO[18]	SPIM_SCK	SPIM_SEL	DUART_TXD	GPIO[63]	GPIO[0]	GPIO[17]	**Functional description**
low	low	high	low	low	**5)**	low	Enable external SRAM memory boot. Processor will start execute application software direct from memory after initialization of the processor.
low	low	high	low	low	**5)**	high	Enable external SRAM memory boot. Processor will start execute application software direct from memory.
low	low	high	high	low	**5)**	low	Enable external ASW SRAM memory boot after initialization of the processor. Processor will copy and extract ASW container before executing application software.
low	low	high	high	high	**5)**	low	Enable external ASW SRAM memory with DMR protection boot after initialization of the processor. Processor will copy and extract ASW container before executing application software.

Note 1: To enable external memory access SPIM_MOSI must be bootstrapped to low.

Note 2: Configuration pin has no effect or not used for bootstrap configuration. It recommend to tie the pin either to low or high.

Note 3: Enable ASW protection. ASW protection usage is described in section 51.

Note 4: Enable dual module redundancy protection. Option only valid in combination with ASW protection. Redundant memory is expected to be located at address allocated for external chip select signal 1.

Note 5: Enable BCH EDAC protection. EDAC can be enabled and used in combination with all other options. When configuration is used external memory must included BCH check bits.

Note 6: Enable bypass of the internal boot ROM. When enabled the processor will start execute code directly from the primary memory at 0x40000000. Processor or internal memory is initialized after reset when this option is used.

Note 7: MEM1_CONFIG(x) pins shall be set to ‘High’ for PBGA and SIP devices to get FTMCTRL0 (8-bit memory controller) SRAM boot.

3.1.4 Boot strap configuration for external PROM/FLASH memory

This section describes valid bootstrap configuration for use of external memory options:

Table 26. External PROM/FLASH memory bootstrap configurations

Pin							
Remote Access	Source		ASW 3)	Red 4)	EDAC 5)	Bypass 6)	
SPIM_MOSI GPIO[18]	SPIM_SCK	SPIM_SEL	DUART_TXD	GPIO[63]	GPIO[0]	GPIO[17]	**Functional description**
low	high	low	low	low	5)	low	Enable external PROM/ FLASH memory boot. Processor will start execute application software direct from memory after initialization of the processor.
low	high	low	low	low	5)	high	Enable external PROM/ FLASH memory boot. Processor will start execute application software direct from memory.
low	high	low	high	low	5)	low	Enable external ASW PROM/FLASH memory boot after initialization of the processor. Processor will copy and extract ASW container before executing application software.
low	high	low	high	high	5)	low	Enable external ASW PROM/FLASH memory with DMR protection boot after initialization of the processor. Processor will copy and extract ASW container before executing application software.

Note 1: To enable external memory access SPIM_MOSI must be bootstrapped to low

Note 2: Configuration pin has no effect or not used for bootstrap configuration. It recommend to tie the pin either to low or high.

Note 3: Enable ASW protection. ASW protection usage is described in section 51.

Note 4: Enable dual module redundancy protection. Option only valid in combination with ASW protection. Redundant memory is expected to be located at address allocated for external chip select signal 1.

Note 5: Enable BCH EDAC protection. EDAC can be enabled and used in combination with all other options. When configuration is used external memory must included BCH check bits.

Note 6: Enable bypass of the internal boot ROM. When enabled the processor will start execute code directly from the primary memory at 0x01000000. Processor or internal memory is initialized after reset when this option is used.

Note 7: MEM1_CONFIG(x) pins shall be set to ‘High’ for PBGA and SIP devices to get FTMCTRL0 (8-bit memory controller) PROM boot.

3.2 Configuration for flight

To achieve the intended functionality in flight, certain signals must be held at a fixed configuration:

- DUART_RXD must have a external pull-down resistor to avoid any erroneous trigger on noise injected on the debug interface.
- DSUEN can be pulled-down to disable debug capabilities in flight, if not used.

4 Clocking

The GR716B microcontroller has two dedicated clock input pins SYS_CLK, and SPW_CLK. There are multiple interface clock inputs that can be supplied via the I/O switch matrix. And several internally generated clocks. The clock inputs are listed in Table 27. The internally used clock domains are listed in Table 28.

The SYS_CLK pin is used as the main system clock, and directly drive the clock network without PLL. The SYS_CLK is selected by default as system clock. The system clocks shall always be running during reset and normal operation.

The SPW_CLK pin is the external SpaceWire clock, and it can be used to generate the internal clocks directly or multiplied with a PLL, depending on the value of the configuration registers in PLL configuration block, see section 10.

Table 27. Clock inputs

Clock input	Description	Frequency Range
SYS_CLK	Dedicated system clock input. Default source of internal system clock after reset. May also be used as clock source for MIL-STD-1553B interface (if SYS_CLK frequency is 20 MHz) and/or FPGA scrubber interface (after division).	up to 100 MHz [1)]
SPW_CLK	Dedicated PLL clock input. Source of internal SpaceWire clock at reset unless using a remote access boot mode with enabled PLL.	5 to 100 MHz [2)]
ETH_TXCLK	Interface clock for ETH_TX* signals from GRETH Ethernet MAC. Normally supplied by Ethernet PHY.	up to 25 MHz
ETH_RXCLK	Interface clock for ETH_RX* signals to GRETH Ethernet MAC. Normally supplied by Ethernet PHY.	up to 25 MHz
1553_CLK	Optional input for MIL-STD-1553B interface clock from I/O switch matrix if interface clock is not taken from SYS_CLK or PLL.	20 MHz [3)]
SPI4S_SCK	SPI for Space clock	up to 25 MHz [4)]
PDAC0_CLK, PDAC1_CLK, PDAC2_CLK, PDAC3_CLK	Optional APWMDAC external clock input. APWMDAC clock can also be generated internally by division of the system clock, in which case PDAC0-3_CLK are outputs.	up to 25 MHz [5)]

Note 1: SYS_CLK must at all times be supplied. The only exception is during part of the power-up and power-down sequence when POR is active, or at other times when RESET_IN_N is low. Other clock inputs besides SYS_CLK only need to be supplied if the clock is used by the application.

Note 2: When SPW_CLK is used directly as internal SpaceWire clock, the duty cycle shall be set to 50/50 for best jitter performance. Additionally the frequency must be a multiple of 10 MHz +- 10% (9-11 MHz, 18-22 MHz, 27-33 MHz, 36-44 MHz, or >45 MHz) to make startup bit rate 10 Mb/s +-10% possible and for link state machine timeouts to be correct. When SPW_CLK is used as PLL input, only discrete frequencies 5 MHz, 10 MHz, 12.5 MHz, 20 MHz, 25 MHz, 50 MHz, 100 or 200 MHz are allowed.

Note 3: Duty cycle for MIL-STD-1553B clock shall be at least 40%. The system clock frequency must be larger than or equal to 10 MHz for the MIL-STD-1553B interface to function.

Note 4: Frequency shall be equal or lesser than system clock frequency divided by 2.

Note 5: Frequency shall be equal or lesser than system clock frequency divided by 4.

Table 28. Internally muxed or generated clocks

Clock name	Description	Frequency Range
System clock	Clock for the majority of all logic in the GR716B. During and after reset it is taken directly from the SYS_CLK input. After software configuration, may be generated using the on-chip PLL. The System clock must be 5-100 MHz when driven from any other source than the SYS_CLK pin, such as SPW_CLK or PLL. Otherwise a clock fail detector may switch it unconditionally to the SYS_CLK pin.	up to 100 MHz
SpaceWire clock	Serial interface clock for SpaceWire. The SpaceWire TX bitrate is programmable as an integer divisor of the SpaceWire clock frequency. The maximum SpaceWire RX bitrate is limited by the SpaceWire clock frequency. Taken directly from the SPW_CLK input pin or from on-chip PLL output. Must be a multiple of 10 MHz (+-10%) to enable SpaceWire startup bitrate and link state machine timeouts to be compliant.	10 to 100 MHz
GR1553_CLK	MIL-STD-1553B interface clock. May be taken from an input pin via the I/O switch matrix (1553_CLK). May also be taken from the SYS_CLK input (if equal to 20 MHz), or by dividing the PLL output.	20 MHz
SCRUBBER_CLK	Interface clock for GRSCRUB FPGA Scrubber. This clock is generated from the internal systems clock by dividing with a factor of 4, 8, 16 or 32.	up to 25 MHz
ADC clocks	There is one ADC sampling clock per ADC. The sampling clock is generated by dividing the system clock by an integer. The clock generation is controlled by registers in the ADC peripherals, see section 12.3.8.	up to 10 MHz
DAC clocks	There is one DAC clock per DAC. The DAC clock is generated by dividing the system clock by an even integer. The clock generation is controlled from registers in the DAC peripherals, see section 15.4.5.	up to 25 MHz
APWMDAC sample clock	There is one APWMDAC clock per APWMDAC unit. It can be generated by dividing the internal system clock or be taken from an input pin via the I/O switch matrix. When generated internally, it can be output via the I/O switch matrix. See sections 2.6 and 54.2.2.	up to 25 MHz
FIR sample clock	There are eight FIR blocks divided int two groups of four with one sample clock per group. The FIR sample clock is generated by dividing the internal system clock. This is controlled by registers in the FIR unit, see section 16.2.	up to 25 MHz
PLL system clock detector ring oscillator 0 and 1	On-chip ring oscillator outputs. Used to monitor system clock when PLL used as source for system clock. See section 4.3.1.	2-4 MHz
APWM system clock detector ring oscillator 0 and 1	On-chip ring oscillator outputs. Used to monitor system clock and allow alarm to be generated asynchronously if system clock is out of expected frequency range. See section 53.6.2.	2-4 MHz

The microcontroller PLL can be used to generate frequencies required for SpaceWire, 1553B or the system clock. The lowest PLL input frequency to be used with the integrated PLL is 5 MHz to be able to meet jitter performance for SpaceWire (with ideal supply). The clock divider block, which supplies the internal SpaceWire clock, can be configured to generate a maximum clock frequency of 100 MHz. However, if a 200 MHz clock is required for the internal SpaceWire clock, it cannot be achieved using the PLL and clock divisor block. In such a case, the external SpaceWire clock must be directly supplied with a 200 MHz clock, and the PLL must be bypassed using the configuration registers described in section 10.

Clock distribution and configuration in the microcontroller is shown in figure 16. In figure 16 the 'blue', 'green' and 'grey' boxes represents logic. External pins are marked with 'names' for cross reference to the pin list in section 2.5. Control registers accessible via software or external boot-straps in order to setup and configure clocks in the system are named using the format *<register name>.<bit-field>*.

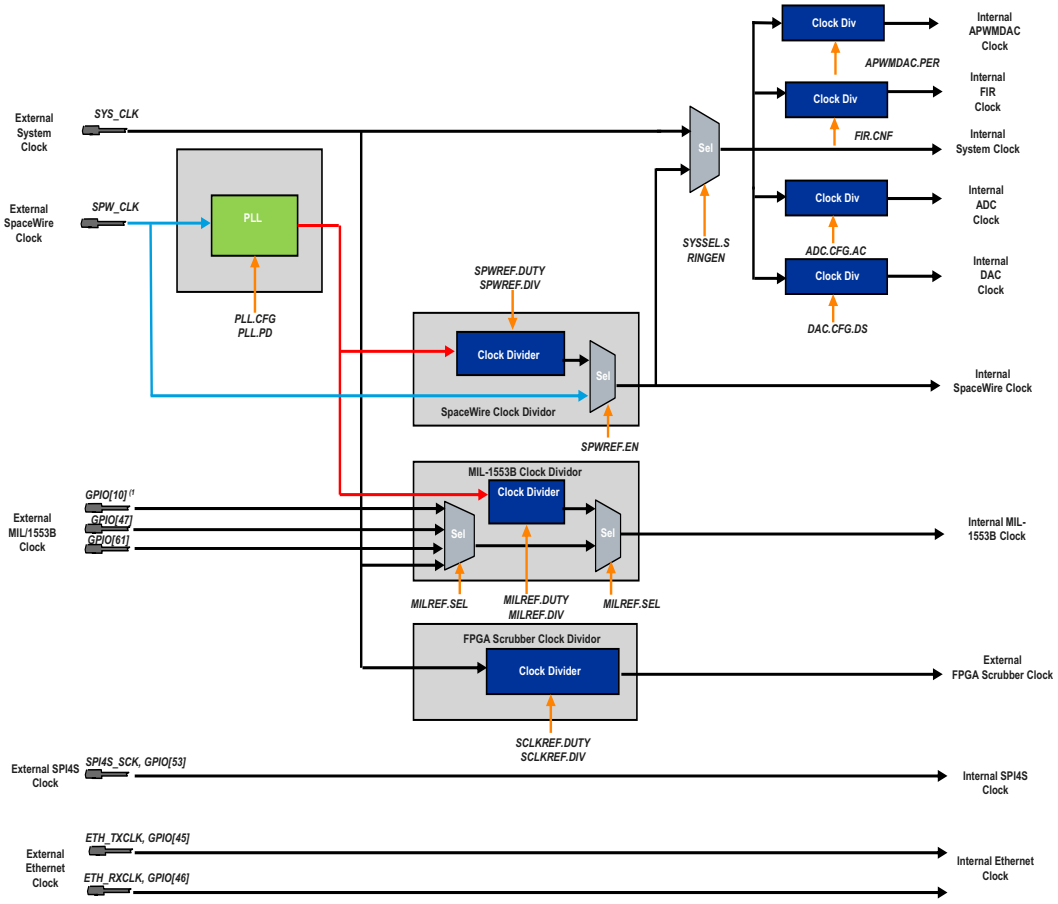


Figure 16. GR716B microcontroller clock distribution scheme and control register.

Note 1: GPIO[10] can be selected to use plain CMOS or Schmitt trigger input. Refer to Table 45.

4.1 PLL Configuration and Status

The PLL is designed to mitigate radiation effects and to always output 400 MHz. In order to lock and generate a 400 MHz output clock the PLL needs to be programmed with the input clocks frequency. The input clock frequency is set via PLL control and status registers, see section 10.

When the GR716B Microcontroller is configured to be controlled via remote access the PLL is configured automatically after reset by the hardware. The setup used is determined by configuration boot-straps specified in chapter 3.1. The input frequency needs to be known by the hardware in order to properly setup and synchronize the remote access link.

4.2 Clock Source and divisor

The system clock, SpaceWire clock, FPGA Scrubber and GR1553B clock can be generated internally from internal or external sources, see figure 16 and section 10. Clock source and divisor is selected via configuration registers described in section 10.

The clock source and divisor needs to be chosen carefully depended upon the application requirements for clock frequency, clock jitter and clock duty cycle.

4.3 System clock

The internal system clock is used to clock the processors, the AMBA buses, and all on-chip cores. This clock can be derived directly from input pin SYS_CLK or from the external pin SPW_CLK or from the internal PLL.

4.3.1 System clock source selection

The internal system clock can be configured to run slower or faster than the external SPW_CLK. Special care needs to be taken when switching system clock source in order to switch to a existing clock source. The device will automatically switch back to use the default system input clock during reset and if the system tries to switch to a disabled clock source.

4.3.2 Internal crystal oscillator (XO)

The microcontroller includes an on-chip oscillator able to provide a 5 - 25 MHz internal clock. This clock can optionally be used to generate other on-chip clocks for the processor system, SpaceWire and MIL-STD-1553B. To be able to provide a high-accuracy reference clock a crystal oscillator is implemented, where the active oscillator part is implemented on-chip and the crystal is to be connected externally.

The output from the on-chip oscillator needs to be connected outside the microcontroller device if it is to be used. Refer section 9 for further information about the on-chip crystal oscillator.

Alternatively, arbitrary clock sources with appropriate digital output voltage levels can be connected directly to the microcontroller clock inputs. In this case the XO can be left unused.

4.4 SpaceWire clock

The clock used for the SpaceWire link receiver and transmitter logic is taken from the dedicated SpaceWire clock pin SPW_CLK either directly, or multiplied with a PLL, depending on the value of the configuration register for the SpaceWire clock mux and PLL. See chapter 10 for more information. It is possible to source the internal system clock using the SPW_CLK via the internal PLL, however, the system clock must at all times be supplied to the system via the SYS_CLK.

4.5 MIL-STD-1553B clock

The 20 MHz clock for the MIL-STD-1553B codec can be taken from the IO mux (GPIO 10, 47 or 61). The clock can also be sourced from SYS_CLK. Regardless of the source of the MIL-STD-1553B clock, the system clock frequency must be greater than or equal to 10 MHz for the MIL-STD-1553B interface to function.

4.5.1 Using PLL clock as input clock for 1553B interface

The PLL output clock frequency can be used to generate a MIL-STD-1553B clock. The MIL-STD-1553B clock can be generated by dividing the PLL frequency by 20, see section 10 for details on the MIL-STD-1553B clock divisor registers.

4.6 ADC Clock

ADC clock shall match the sampling speed required by the application. Maximum sampling speed is 500 kSps i.e. maximum ADC clock frequency is 10 MHz and using minimum track time. The ADC clock is configured via registers, see section 12.

4.7 DAC Clock

DAC clock shall match the sampling speed required by the application. Maximum sampling speed is 25 MSps i.e. maximum DAC clock frequency is 25 MHz. The DAC clock is configured via registers, see section 15.

4.8 Clock gating unit

The design has a clock gating unit through which individual cores can have their clocks enabled/disabled and resets driven.

The LEON3 processor core will automatically be clock gated when the processor enters power-down or halt state. The floating-point units (GRFPU) will be clock gated when the corresponding processor has disabled FPU operations by setting the %psr.ef bit to zero, or when the processor has entered power-down/halt mode.

For more information see the chapter about the clock gating unit section 27.

4.9 Debug AHB bus clocking

All cores on the Debug AHB bus will be gated off when the DSU_EN signal is set to low.

4.10 PLL Lock and clock output

The internal PLL lock signal, SpW, MIL-1553, Scrubber and System clock can be driven out of the chip using GPIO pins for diagnostic purposes. Refer to table 81 for register enables.

5 Reset

The device has an on-chip reset generator that creates a reset signal that is fed to the rest of the system. The reset is asynchronously set and synchronously released after a delay. The delay can be controlled by connecting a external capacitance to the external pin C_RST input.

All peripherals can be reset independently while the processor continues execution. Thus giving the option to force the full device into a known state during reset mode or just applying a hard reset to selected peripherals. Peripherals are reset independently via register accessible from the processor in the microcontroller or via remote accesses via UART, SPI, CAN-FD, MIL-STD-1553B or SpaceWire interface. Remote access via MIL-STD-1553B requires external boot ram. For more information about individual reset control see chapter 27.2.

The microcontroller includes a brown-out detector to supervise the external power supply for the system to shutdown in a controlled manor. A system shutdown is requested via an interrupt to the processor by the brown-out detector in case the supply voltage falls below a specific value. The voltage level is programmable and is always set to the lowest possible value by default after reset.

5.1 IO Reset

The 64 General purpose IO and LVDS described in chapter 2.4 and 2.5 will set to high impedance mode during power-up/down, Brown detection or if a failure has been detected in the IO configuration registers described in chapter 7.1.

Table 29: Digital IO reset state table

Name	Notes	Power-up/down ramping (TBD)	State during reset	State after reset	Normal state
RESET_OUT_N		HiZ	HiZ/Dig output 0	Dig output 1	Dig output 1
RESET_IN_N	Reset input override	Input	Input	Input	Input
XO_OUT	Digital clock output	Dig output	Dig output	Dig output	Dig output
SYS_CLK	System clock	Dig input	Dig input	Dig input	Dig input
SPW_CLK	SpaceWire clock	Dig input	Dig input	Dig input	Dig input
DSU_EN	Debug Support Unit enable signal	Dig input	Dig input	Dig input	Dig input
DSU_BREAK	Debug Support Unit break signal	Dig input	Dig input	Dig input	Dig input
DUART_TXD [1)]	Debug UART, transmit data	Undefined	Bootstrap input	Dig output	Dig output
DUART_RXD	Debug UART, receive data	Dig input	Dig input	Dig input	Dig input
SPIM_MOSI [1)]	SPI Memory master output slave input	Undefined	Bootstrap input	Dig output	Dig output
SPIM_SCK [1)]	SPI Memory master clock output	Undefined	Bootstrap input	Dig output	Dig output
SPIM_SEL [1)]	SPI Memory slave select output	Undefined	Bootstrap input	Dig output	Dig output
SPIM_MISO	SPI Memory master input slave output	Dig input	Dig input	Dig input	Dig input

Table 29: Digital IO reset state table

Name	Notes	Power-up/down ramping (TBD)	State during reset	State after reset	Normal state
LVDS_RX[0]p	For functional pin description see section 2.5 and datasheet section 3.2 [DS].	HiZ	HiZ	HiZ	User defined
LVDS_RX[0]n		HiZ	HiZ	HiZ	User defined
LVDS_RX[1]p		HiZ	HiZ	HiZ	User defined
LVDS_RX[1]n		HiZ	HiZ	HiZ	User defined
LVDS_RX[2]p		HiZ	HiZ	HiZ	User defined
LVDS_RX[2]n		HiZ	HiZ	HiZ	User defined
LVDS_TX[0]p		Undefined	Outputs	HiZ	User defined
LVDS_TX[0]n		Undefined	Outputs	HiZ	User defined
LVDS_TX[1]p		Undefined	Outputs	HiZ	User defined
LVDS_TX[1]n		Undefined	Outputs	HiZ	User defined
LVDS_TX[2]p		Undefined	Outputs	HiZ	User defined
LVDS_TX[2]n		Undefined	Outputs	HiZ	User defined
LVDS_TX[3]p		Undefined	Outputs	HiZ	User defined
LVDS_TX[3]n		Undefined	Outputs	HiZ	User defined
LVDS_TX[4]p		Undefined	HiZ	HiZ	User defined
LVDS_TX[4]n		Undefined	HiZ	HiZ	User defined
LVDS_TX[5]p		Undefined	HiZ	HiZ	User defined
LVDS_TX[5]n		Undefined	HiZ	HiZ	User defined
GPIO[0] 1)	For functional pin description see section 2.5 and datasheet section 3.2 [DS].	HiZ	Bootstrap input	Dig input	User defined
GPIO[1] 1)		HiZ	Bootstrap input	Dig input	User defined
GPIO[2:14]		HiZ	Dig input	Dig input	User defined
GPIO[15] 1)		HiZ	Bootstrap input	Dig input	User defined
GPIO[16]		HiZ	Dig input	Dig input	User defined
GPIO[17] 1)		HiZ	Bootstrap input	Dig input	User defined
GPIO[18] 1)		HiZ	Bootstrap input	Dig input	User defined
GPIO[19:36]		HiZ	Dig input	Dig input	User defined
GPIO[37:61]		Undefined	Dig input	Dig input	User defined
GPIO[62] 1)		Undefined	Bootstrap input	Dig input	User defined
GPIO[63] 1)		HiZ	Bootstrap input	Dig input	User defined
TESTEN	Test enable signal	Dig input	Dig input	Dig input	Dig input

Note 1: External pin should have an external pull-up/down to ground or supply

6 Technical notes

6.1 GRLIB AMBA plug&play scanning

The bus structure in this design requires some special consideration with regard to plug&play scanning. The default behavior of GRLIB AMBA plug&play scanning routines is to start scanning at address 0xFFFF0000. If any AHB/AHB bridges or APB bridges are detected during the scan, the general scanning routine traverses the bridge and reads the plug&play information from the bus behind the bridge. In this design, the default 0xFFFF0000 address gives plug&play information only for the Processor AHB bus. For the plug&play scanning routine to get plug&play information from all AHB buses the start address 0x9FFF0000 need to be used.

6.2 Software portability

6.2.1 Instruction set architecture

The LEON3FT processor used in this design implements the SPARC V8 instruction set architecture. This means that any compiler that produces valid SPARC V8 executables can be used. Full instruction set compatibility is kept with LEON2FT and LEON3FT applications.

6.2.2 Peripherals

Standard GRLIB software drivers can be used.

For software driver development, this document describes the capabilities offered by the LEON3FT microcontroller system. In order to write a generic driver for a GRLIB IP core, that can be used on all systems based on GRLIB, please also refer to the generic IP core documentation in GRLIB IP Core User's Manual [GRIP]. Note, however, that the generic documentation may describe functionality not present in this implementation and that this data sheet supersedes any documentation found in [GRIP] for this system.

6.2.3 Plug and play

Standard GRLIB AMBA plug&play layout is used. The same software routines used for typical LEON/GRLIB systems can be used.

7 System Startup Status and General Configuration

This section describes general status register and control registers for LEON3FT microcontroller system. General status and configuration register described in this section are be used for IO function selection and peripheral configuration. GPIOs are sampled during RESET and stored in register for configuration of IO switch matrix and peripherals.

This section also describes how to get access to control signal to analog functions from external pins and how to enable memory build test and interrupt test.

7.1 Configuration Registers

The registers are mapped into AMBA address space. The register layout used for configuration of GPIO is explained in section 2.5.

System register bits affected by bootstraps are marked with a '*' in reset value field.

Table 30. System IO configuration register

AMBA address	Register	Acronym
0x8000D000	System IO configuration for GPIO 0 to 7	SYS.CFG.R0
0x8000D004	System IO configuration for GPIO 8 to 15	SYS.CFG.R1
0x8000D008	System IO configuration for GPIO 16 to 23	SYS.CFG.R2
0x8000D00C	System IO configuration for GPIO 24 to 31	SYS.CFG.R3
0x8000D010	System IO configuration for GPIO 32 to 39	SYS.CFG.R4
0x8000D014	System IO configuration for GPIO 40 to 47	SYS.CFG.R5
0x8000D018	System IO configuration for GPIO 48 to 55	SYS.CFG.R6
0x8000D01C	System IO configuration for GPIO 56 to 63	SYS.CFG.R7
0x8000D020	System IO Pullup configuration for GPIO 0 to 31	SYS.CFG.PULLUP0
0x8000D024	System IO Pullup configuration for GPIO 32 to 64	SYS.CFG.PULLUP1
0x8000D028	System IO Pulldown configuration for GPIO 0 to 31	SYS.CFG.PULLDOWN0
0x8000D02C	System IO Pulldown configuration for GPIO 32 to 64	SYS.CFG.PULLDOWN1
0x8000D030	LVDS configuration 0	SYS.CFG.LVDS0
0x8000D034	LVDS configuration 1	SYS.CFG.LVDS1
0x8000D038	System IO Schmitt trigger configuration for GPIO 0 to 31	SYS.CFG.SCHMITT0
0x8000D03C	System IO Schmitt trigger configuration for GPIO 32 to 64	SYS.CFG.SCHMITT1
0x8000D070	System IO configuration interrupt register	SYS.CFG.IRQ
0x8000D074	System IO configuration interrupt status register	SYS.CFG.IRQSTAT
0x8000D078	System IO configuration interrupt mask	SYS.CFG.MASK
0x8000D07C	System IO configuration interrupt edge	SYS.CFG.EDGE
0x8000D0E0	System IO configuration local register interface information	SYS.CFG.INFO

Table 31. 0x8000D000 - SYS.CFG.R0 - System GPIO configuration register0

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
GP7	GP6	GP5	GP4	GP3	GP2	GP1	GP0
* 1)	* 1)	* 1)	* 1)	* 1)	* 1)	* 1)	* 1)
rw	rw	rw	rw	rw	rw	rw	rw

31: 28 GPIO7 functional select (GP7) - Select functionality for GPIO pin 7. For functionality see Table 7.

27: 24 GPIO6 functional select (GP6) - Select functionality for GPIO pin 6. For functionality see Table 7.

23: 20 GPIO5 functional select (GP5) - Select functionality for GPIO pin 5. For functionality see Table 7.

Table 31. 0x8000D000 - SYS.CFG.R0 - System GPIO configuration register0

19: 16 GPIO4 functional select (GP4) - Select functionality for GPIO pin 4. For functionality see Table 7.

15: 12 GPIO3 functional select (GP3) - Select functionality for GPIO pin 3. For functionality see Table 7.

11: 8 GPIO2 functional select (GP2) - Select functionality for GPIO pin 2. For functionality see Table 7.

7: 4 GPIO1 functional select (GP1) - Select functionality for GPIO pin 1. For functionality see Table 7.

3: 0 GPIO0 functional select (GP0) - Select functionality for GPIO pin 0. For functionality see Table 7.

Note 1: Reset value is set by bootstrap, see section 2.7.

Table 32. 0x8000D004 - SYS.CFG.R1 - System GPIO configuration register1

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
GP7	GP6	GP5	GP4	GP3	GP2	GP1	GP0
* 1)	* 1)	* 1)	* 1)	* 1)	* 1)	* 1)	* 1)
rw	rw	rw	rw	rw	rw	rw	rw

31: 28 GPIO15 functional select (GP7) - Select functionality for GPIO pin 15. For functionality see Table 7.

27: 24 GPIO14 functional select (GP6) - Select functionality for GPIO pin 14. For functionality see Table 7.

23: 20 GPIO13 functional select (GP5) - Select functionality for GPIO pin 13. For functionality see Table 7.

19: 16 GPIO12 functional select (GP4) - Select functionality for GPIO pin 12. For functionality see Table 7.

15: 12 GPIO11 functional select (GP3) - Select functionality for GPIO pin 11. For functionality see Table 7.

11: 8 GPIO10 functional select (GP2) - Select functionality for GPIO pin 10. For functionality see Table 7.

7: 4 GPIO9 functional select (GP1) - Select functionality for GPIO pin 9. For functionality see Table 7.

3: 0 GPIO8 functional select (GP0) - Select functionality for GPIO pin 8. For functionality see Table 7.

Note 1: Reset value is set by bootstrap, see section 2.7.

Table 33. 0x8000D008 - SYS.CFG.R2 - System GPIO configuration register2

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
GP7	GP6	GP5	GP4	GP3	GP2	GP1	GP0
0x0	0x0	0x0	0x0	0x0	0x0	* 1)	* 1)
rw	rw	rw	rw	rw	rw	rw	rw

31: 28 GPIO23 functional select (GP7) - Select functionality for GPIO pin 23. For functionality see Table 7.

27: 24 GPIO22 functional select (GP6) - Select functionality for GPIO pin 22. For functionality see Table 7.

23: 20 GPIO21 functional select (GP5) - Select functionality for GPIO pin 21. For functionality see Table 7.

19: 16 GPIO20 functional select (GP4) - Select functionality for GPIO pin 20. For functionality see Table 7.

15: 12 GPIO19 functional select (GP3) - Select functionality for GPIO pin 19. For functionality see Table 7.

11: 8 GPIO18 functional select (GP2) - Select functionality for GPIO pin 18. For functionality see Table 7.

7: 4 GPIO17 functional select (GP1) - Select functionality for GPIO pin 17. For functionality see Table 7.

3: 0 GPIO16 functional select (GP0) - Select functionality for GPIO pin 16. For functionality see Table 7.

Note 1: Reset value is set by bootstrap, see section 2.7.

Table 34. 0x8000D00C - SYS.CFG.R3 - System GPIO configuration register 3

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
GP7	GP6	GP5	GP4	GP3	GP2	GP1	GP0
* 1)	* 1)	* 1)	* 1)	* 1)	* 1)	* 1)	0x0
rw	rw	rw	rw	rw	rw	rw	rw

31: 28 GPIO31 functional select (GP7) - Select functionality for GPIO pin 31. For functionality see Table 7.

27: 24 GPIO30 functional select (GP6) - Select functionality for GPIO pin 30. For functionality see Table 7.

23: 20 GPIO29 functional select (GP5) - Select functionality for GPIO pin 29. For functionality see Table 7.

19: 16 GPIO28 functional select (GP4) - Select functionality for GPIO pin 28. For functionality see Table 7.

15: 12 GPIO27 functional select (GP3) - Select functionality for GPIO pin 27. For functionality see Table 7.

Table 34. 0x8000D00C - SYS.CFG.R3 - System GPIO configuration register 3

11: 8 GPIO26 functional select (GP2) - Select functionality for GPIO pin 26. For functionality see Table 7.

7: 4 GPIO25 functional select (GP1) - Select functionality for GPIO pin 25. For functionality see Table 7.

3: 0 GPIO24 functional select (GP0) - Select functionality for GPIO pin 24. For functionality see Table 7.

Note 1: Reset value is set by bootstrap, see section 2.7.

Table 35. 0x8000D010 - SYS.CFG.R4 - System GPIO configuration register 4

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
GP7	GP6	GP5	GP4	GP3	GP2	GP1	GP0
0x0	0x0	0x0	* 1)	* 1)	* 1)	* 1)	* 1)
rw	rw	rw	rw	rw	rw	rw	rw

31: 28 GPIO39 functional select (GP7) - Select functionality for GPIO pin 39. For functionality see Table 7.

27: 24 GPIO38 functional select (GP6) - Select functionality for GPIO pin 38. For functionality see Table 7.

23: 20 GPIO37 functional select (GP5) - Select functionality for GPIO pin 37. For functionality see Table 7.

19: 16 GPIO36 functional select (GP4) - Select functionality for GPIO pin 36. For functionality see Table 7.

15: 12 GPIO35 functional select (GP3) - Select functionality for GPIO pin 35. For functionality see Table 7.

11: 8 GPIO34 functional select (GP2) - Select functionality for GPIO pin 34. For functionality see Table 7.

7: 4 GPIO33 functional select (GP1) - Select functionality for GPIO pin 33. For functionality see Table 7.

3: 0 GPIO32 functional select (GP0) - Select functionality for GPIO pin 32. For functionality see Table 7.

Note 1: Reset value is set by bootstrap, see section 2.7.

Table 36. 0x8000D014 - SYS.CFG.R5 - System GPIO configuration register 5

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
GP7	GP6	GP5	GP4	GP3	GP2	GP1	GP0
0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
rw	rw	rw	rw	rw	rw	rw	rw

31: 28 GPIO47 functional select (GP7) - Select functionality for GPIO pin 47. For functionality see Table 7.

27: 24 GPIO46 functional select (GP6) - Select functionality for GPIO pin 46. For functionality see Table 7.

23: 20 GPIO45 functional select (GP5) - Select functionality for GPIO pin 45. For functionality see Table 7.

19: 16 GPIO44 functional select (GP4) - Select functionality for GPIO pin 44. For functionality see Table 7.

15: 12 GPIO43 functional select (GP3) - Select functionality for GPIO pin 43. For functionality see Table 7.

11: 8 GPIO42 functional select (GP2) - Select functionality for GPIO pin 42. For functionality see Table 7.

7: 4 GPIO41 functional select (GP1) - Select functionality for GPIO pin 41. For functionality see Table 7.

3: 0 GPIO40 functional select (GP0) - Select functionality for GPIO pin 40. For functionality see Table 7.

Table 37. 0x8000D018 - SYS.CFG.R6 - System GPIO configuration register 6

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
GP7	GP6	GP5	GP4	GP3	GP2	GP1	GP0
* 1)	* 1)	* 1)	0x0	0x0	* 1)	* 1)	0x0
rw	rw	rw	rw	rw	rw	rw	rw

31: 28 GPIO55 functional select (GP7) - Select functionality for GPIO pin 55. For functionality see Table 7.

27: 24 GPIO54 functional select (GP6) - Select functionality for GPIO pin 54. For functionality see Table 7.

23: 20 GPIO53 functional select (GP5) - Select functionality for GPIO pin 53. For functionality see Table 7.

19: 16 GPIO52 functional select (GP4) - Select functionality for GPIO pin 52. For functionality see Table 7.

Table 37. 0x8000D018 - SYS.CFG.R6 - System GPIO configuration register 6

15: 12 GPIO51 functional select (GP3) - Select functionality for GPIO pin 51. For functionality see Table 7.

11: 8 GPIO50 functional select (GP2) - Select functionality for GPIO pin 50. For functionality see Table 7.

7: 4 GPIO49 functional select (GP1) - Select functionality for GPIO pin 49. For functionality see Table 7.

3: 0 GPIO48 functional select (GP0) - Select functionality for GPIO pin 48. For functionality see Table 7.

Note 1: Reset value is set by bootstrap, see section 2.7.

Table 38. 0x8000D01C - SYS.CFG.R7 - System GPIO configuration register 7

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
GP7	GP6	GP5	GP4	GP3	GP2	GP1	GP0
* 1)	* 1)	* 1)	* 1)	* 1)	* 1)	0x0	* 1)
rw	rw	rw	rw	rw	rw	rw	rw

31: 28 GPIO63 functional select (GP7) - Select functionality for GPIO pin 63. For functionality see Table 7.

27: 24 GPIO62 functional select (GP6) - Select functionality for GPIO pin 62. For functionality see Table 7.

23: 20 GPIO61 functional select (GP5) - Select functionality for GPIO pin 61. For functionality see Table 7.

19: 16 GPIO60 functional select (GP4) - Select functionality for GPIO pin 60. For functionality see Table 7.

15: 12 GPIO59 functional select (GP3) - Select functionality for GPIO pin 59. For functionality see Table 7.

11: 8 GPIO58 functional select (GP2) - Select functionality for GPIO pin 58. For functionality see Table 7.

7: 4 GPIO57 functional select (GP1) - Select functionality for GPIO pin 57. For functionality see Table 7.

3: 0 GPIO56 functional select (GP0) - Select functionality for GPIO pin 56. For functionality see Table 7.

Note 1: Reset value is set by bootstrap, see section 2.7.

Table 39. 0x8000D020 - SYS.CFG.PULLUP0 - System GPIO pullup configuration register for GPIO 0 to 31

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
UP
0x00000000
rw

31: 0 Select and configure inputs using internal pullup resistor (PULLUP) - Bit *n* in the bitfield corresponds to GPIO *n*. Set to 1 to enable pullup, 0 to disable pullup. The pullup is only enabled if the pin is configured as an input in the I/O switch matrix. Any pin among GPIO 0-31 in mux mode 0 will not have pullup applied unless GRGPIO0 is enabled in the primary clock gating unit (section 27).

Table 40. 0x8000D024 - SYS.CFG.PULLUP1 - System GPIO pullup configuration register for GPIO 32 to 63

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
UP
0x00000000
rw

31: 0 Select and configure inputs using internal pullup resistor (PULLUP) - Bit *n* in the bitfield corresponds to GPIO 32+*n*. The pullup is only enabled if the pin is configured as a digital input in the I/O switch matrix. Pullup is not applied for pins in analog mode (see section 2.8.3). Pins without analog capability configured in mux mode 0x0 will not have pullup applied unless GRGPIO1 is enabled in the secondary clock gating unit (section 27).

Table 41. 0x8000D028 - SYS.CFG.PULLDOWN0 - System GPIO pulldown configuration register for GPIO 0 to 31

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
DOWN
0x00000000
rw

Table 41. 0x8000D028 - SYS.CFG.PULLDOWN0 - System GPIO pulldown configuration register for GPIO 0 to 31

31: 0 Select and configure inputs using internal pulldown resistor (DOWN) - Bit *n* in the bitfield corresponds to GPIO *n*. Set to 1 to enable pulldown, 0 to disable pulldown. The pulldown is only enabled if the pin is configured as an input in the I/O switch matrix. Any pin among GPIO 0-31 in mux mode 0 will not have pulldown applied unless GRGPIO0 is enabled in the primary clock gating unit (section 27).

Table 42. 0x8000D02C - SYS.CFG.PULLDOWN1 - System GPIO pulldown configuration register for GPIO 32 to 63

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DOWN																															
0x00000000																															
rw																															

31: 0 Select and configure inputs using internal pulldown resistor (DOWN) - Bit *n* in the bitfield corresponds to GPIO 32+*n*. Set to 1 to enable pulldown, 0 to disable pulldown. The pulldown is only enabled if the pin is configured as a digital input in the I/O switch matrix. Pulldown is not applied for pins in analog mode (see section 2.8.3). Pins without analog capability configured in mux mode 0x0 will not have pulldown applied unless GRGPIO1 is enabled in the secondary clock gating unit (section 27).

Table 43. 0x8000D030 - SYS.CFG.LVDS0 - System LVDS configuration register 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RX3				TX3				RX2				RX1				RX0				TX2				TX1				TX0			
* 2)				0x8				* 2)				* 2)				* 2)				* 2)				* 2)				* 2)			
rw				rw				rw				rw				rw				rw				rw				rw			

31: 28 LVDS Reciever 3 (RX3) - Select functionality for LVDS receiver 3

0x0 - LVDS receiver enable [3)]
0x8 - LVDS receiver disable

27: 24 LVDS Transmitter 3 (TX3) - Select functionality for LVDS transmitter 3 [1)]

0x0 - SpaceWire port 1 strobe output (SPW_TXS1) - also available on TX5
0x4 - LVDS GP Out 3 (LVDS_OUT3)
0x8 - LVDS transmitter disable

23: 20 LVDS Reciever 2 (RX2) - Select functionality for LVDS receiver 2

0x0 - LVDS receiver enable [3) 4)]
0x8 - LVDS receiver disable

19: 16 LVDS Reciever 1 (RX1) - Select functionality for LVDS receiver 1

0x0 - LVDS receiver enable [3) 4)]
0x8 - LVDS receiver disable

15: 12 LVDS Reciever 0 (RX0) - Select functionality for LVDS receiver 0

0x0 - LVDS receiver enable [3) 4)]
0x8 - LVDS receiver disable

11: 8 LVDS Transmitter 2 (TX2) - Select functionality for LVDS transmitter 2 [1)]

0x0 - SpaceWire router port 2 data output (SPW_TXD1) - also available on TX4
0x1 - SPI for Space Slave MISO output of nominal interface (SPI4S_MISO_N)
0x2 - SPI controller 0 MOSI output (SPI_MOSI0) - only used in master mode
0x3 - SPI controller 0 MISO output (SPI_MISO0) - only used in slave mode
0x4 - LVDS GP Out 2 (LVDS_OUT2)
0x8 - LVDS transmitter disable

7: 4 LVDS Transmitter 1 (TX1) - Select functionality for LVDS transmitter 1 [1)]

0x0 - SpaceWire router port 1 [5)] strobe output (SPW_TXS0)
0x2 - SPI controller 0 slave select output 0 (SPI_SLV0_0) - only used in master mode
0x4 - LVDS GP Out 1 (LVDS_OUT1)
0x8 - LVDS transmitter disable

Table 43. 0x8000D030 - SYS.CFG.LVDS0 - System LVDS configuration register 0

3: 0 LVDS Transmitter 0 (TX0) - Select functionality for LVDS transmitter 0 [1)]

0x0 - SpaceWire router port 1 [5)] data output (SPW_TXD0)
0x2 - SPI controller 0 SCK output (SPI_SCK0) - only used in master mode
0x4 - LVDS GP Out 0 (LVDS_OUT0)
0x8 - LVDS transmitter disable

Note 0: To use any LVDS receiver or transmitter, the LVDS IBIAS and LVDS VREF for that receiver or transmitter must be enabled. See R0, R1, and R2 fields in the SYS.CFG.LVDS1 register (Table 44).

Note 1: Transmitter configurations less than 0x8 not listed will force the transmitter to output a logic '0'. Transmitter configurations greater than 0x8 will disable the transmitter.

Note 2: The reset values of the TX0, TX1, RX0, RX1, RX2, and RX3 fields depend on the bootstrap configuration, see section 2.7. When configured for SpaceWire remote access boot mode, these fields all reset to 0x0. In all other boot modes they reset to 0x8.

Note 3: The connections listed below are always active. Whether or not a given connection is used depends on the

LVDS Receiver 0 (RX0): SpaceWire router port 2 [5)] data input (SPW_RXD1)
SPI for Space Slave SCK input of nominal interface (SPI4S_SCK_N)
LVDS GP In 0 (LVDS_IN0)

LVDS Receiver 1 (RX1): SpaceWire router port 2 [5)] data input (SPW_RXD0)
SPI for Space Slave MOSI output of nominal interface (SPI4S_MOSI_N)
LVDS GP In 1 (LVDS_IN1)

LVDS Receiver 2 (RX2): SpaceWire router port 1 [5)] strobe input (SPW_RXS0)
SPI for Space Slave Select input of nominal interface (SPI4S_CS_N)
LVDS GP In 2 (LVDS_IN2)

LVDS Receiver 3 (RX3): SpaceWire router port 1 [5)] strobe input (SPW_RXS1)
LVDS GP In 3 (LVDS_IN3)

Note 4: The connections listed below are active only if a GPIO is not employed for the relevant signal (see Table 7):

LVDS Receiver 0 (RX0): SPI controller 0 MISO input (SPI_MISO0) - only used in master mode
SPI controller 0 SCK input (SPI_SCK0) - only used in slave mode

LVDS Receiver 1 (RX1): SPI controller 0 MOSI input (SPI_MOSI0) - only used in slave mode

LVDS Receiver 2 (RX2): SPI controller 0 SEL input (SPI_SEL0) - used in both master and slave mode

Note 5: The *SpaceWire router port* numbering differs from the *SpaceWire interface* numbering as follows:

SpaceWire router port 0 - configuration port (internal, RMAP commands only)
SpaceWire router port 1 - external SpaceWire interface 0 (SPW_TXD0, SPW_TXS0, SPW_RXD0, SPW_RXS0).
SpaceWire router port 2 - external SpaceWire interface 1 (SPW_TXD1, SPW_TXS1, SPW_RXD1, SPW_RXS1).
SpaceWire router port 3 - AMBA port 0 (internal SpaceWire node)

See section 33 for more information about the embedded SpaceWire router.

Table 44. 0x8000D034 - SYS.CFG.LVDS1 - System LVDS configuration register 1

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16	15	14	13	12	11 10 9 8	7 6 5 4	3 2 1 0
RESERVED	R3	R2	R1	R0	FS	TX5	TX4
0	0	1	* 2)	* 2)	0b0000	* 2)	* 2)
r	rw	rw	rw	rw	rw	rw	rw

31: 16 Reserved

15 Unused field (R3) - This bit is readable and writable but has no other effect on the system.

14 LVDS reference control 2 (R2) - LVDS VFREF power disable for TX4-5

13 LVDS reference control 1 (R1) - LVDS VFREF power disable for RX0-3 and TX0-3

12 LVDS reference control 0 (R0) - LVDS IBIAS power enable for RX0-3 and TX0-3

11 LVDS fail-safe control (FS[3]) - Disable Fail-Safe detection for RX3

10 LVDS fail-safe control (FS[2]) - Disable Fail-Safe detection for RX2

9 LVDS fail-safe control (FS[1]) - Disable Fail-Safe detection for RX1

8 LVDS fail-safe control (FS[0]) - Disable Fail-Safe detection for RX0

Table 44. 0x8000D034 - SYS.CFG.LVDS1 - System LVDS configuration register 1

7: 4 LVDS Transmitter 5 (TX5) - Select functionality for LVDS transmitter 5 [1)]

0x0 - SpaceWire router port 2 [3)] strobe output (SPW_TXS1) - also available on TX3
0x4 - LVDS GP Out 4 (LVDS_OUT4)
0x8 - LVDS transmitter disable

3: 0 LVDS Transmitter 4 (TX4) - Select functionality for LVDS transmitter 4 [1)]

0x0 - SpaceWire router port 2 [3)] data output (SPW_TXD1) - also available on TX2
0x1 - SPI for Space Slave MISO output of nominal interface (SPI4S_MISO_N)
0x2 - SPI controller 0 MOSI output (SPI_MOSI0) - only used in master mode
0x3 - SPI controller 0 MISO output (SPI_MISO0) - only used in slave mode
0x4 - LVDS GP Out 3 (LVDS_OUT4)
0x8 - LVDS transmitter disable

Note 0: When using any of RX0-3 or TX0-3 in any non-disabled mode, their LVDS IBIAS and LVDS VREF must be enabled by setting R0=1 and R1=0 (these two fields have opposite polarity).

When using any of TX4 or TX5 in any non-disabled mode, their LVDS VREF must be enabled by setting R2=0.

Note 1: Transmitter configurations less than 0x8 not listed will force the transmitter to output a logic '0'. Transmitter configurations greater than 0x8 will disable the LVDS transmitter.

Whenever TX4 is enabled, the CMOS outputs of GPIO59 and GPIO60 will be forced to 'HiZ'.

Whenever TX5 is enabled, the CMOS outputs of GPIO61 and GPIO62 will be forced to 'HiZ'.

Note 2: The reset values of fields R0, R1, TX4, and TX5 depend on bootstrap settings, see section 2.7:
SpaceWire remote access boot mode: TX4=TX5=0, R0=1, R1=0.
All other boot modes: TX4=TX5=0x8, R0=0, R1=1.

Note 3: The *SpaceWire router port* numbering differs from the *SpaceWire interface* numbering as follows:

SpaceWire router port 0 - configuration port (internal, RMAP commands only)
SpaceWire router port 1 - external SpaceWire interface 0 (SPW_TXD0, SPW_TXS0, SPW_RXD0, SPW_RXS0).
SpaceWire router port 2 - external SpaceWire interface 1 (SPW_TXD1, SPW_TXS1, SPW_RXD1, SPW_RXS1).
SpaceWire router port 3 - AMBA port 0 (internal SpaceWire node)

See section 33 for more information about the embedded SpaceWire router.

Table 45. 0x8000D038 - SYS.CFG.SCHMITT0 - System GPIO schmitt trigger configuration register for GPIO 0 to 31

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENABLE																															
0x00000000																															
rw																															

31: 0 Select and configure inputs using internal schmitt trigger - Bit *n* in the bitfield corresponds to GPIO *n*.

Table 46. 0x8000D03C - SYS.CFG.SCHMITT1 - System GPIO schmitt trigger configuration register for GPIO 32 to 63

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN	RESERVED																										ENABLE	RESERVED			ENABLE
0	0x0																										0x0	0x00			0x00
rw																															

63 Select and configure input using internal schmitt trigger - GPIO 63.

30: 19 Reserved (GPIO 51 to 62)

18: 17 Select and configure inputs using internal schmitt trigger - Bit *n* in the bitfield corresponds to GPIO 49+*n*.

16: 5 Reserved (GPIO 37 to 48)

4: 0 Select and configure inputs using internal schmitt trigger - Bit *n* in the bitfield corresponds to GPIO 32+*n*.

Table 47. 0x8000D070 - SYS.CFG.INT - System IO configuration interrupt register

31	1	0
RESERVED		BP
0x00000000		0
r		rw

31: 1 Reserved

0 Bit parity check error (BP) - Configuration bit error detected. This parity check covers the 16 configuration registers. Becomes '1' when a configuration parity error is detected.

Note: Writable for testing purposes, generates respective interrupts.

Table 48. 0x8000D074 - SYS.CFG.IRQSTAT - System IO configuration interrupt status register

31	1	0
RESERVED		BP
0x00000000		0
r		wc

31: 1 Reserved

0 This register is set to 1 whenever an interrupt has been generated due to bit *0* of SYS.CFG.INT. This bit stays set to 1 until software clears it by writing. Write 1 to clear.

Table 49. 0x8000D078 - SYS.CFG.MASK - System IO configuration interrupt mask register

31	1	0
RESERVED		BP
0x00000000		0
r		rw

31: 1 Reserved

0 This register controls whether the event corresponding to bit *0* of SYS.CFG.INT can generate an interrupt or not. When this bit is 0, the interrupt event will not generate interrupt.

Table 50. 0x8000D07C - SYS.CFG.EDGE - System IO configuration interrupt edge register

31	1	0
RESERVED		BP
0x00000000		1
r		rw

31: 1 Reserved

0 This registers controls the polarity of interrupt generation from bit *0* of the SYS.CFG.INT register.
0 => falling edge: interrupt generated when bit *0* changes from 1 to 0
1 => rising edge: interrupt generated when bit *0* changes from 0 to 1

Table 51. 0x8000D0E0 - SYS.CFG.INFO - System IO configuration Local register interface information

31 24	23 16	15 8	7 0
IRQ	CFG	STAT	IRQ
0x01	0x10	0x00	0x01
r	r	r	r

31: 24 IRQ: Number of Interrupts (same as 7:0)

23: 16 CFG: Number of Configuration registers

Table 51. 0x8000D0E0 - SYS.CFG.INFO - System IO configuration Local register interface information

15:8	STAT: Number of Status registers
7:0	IRQ: Number of Interrupts

7.2 Bootstrap information register

The register shows the current status of the external boot strap configuration used. The register can be modified in order to trigger a reboot and re-configuration of the microcontroller using the internal on-chip boot ROM. The default settings after reset depends on the state of external bootstrap pins. See section 3.1 for details.

Table 52. Boot strap register

AMBA address	Register	Acronym
0x80008000	Internal boot ROM configuration register. Register gets default value from external bootstrap pins after reset.	SYS.CFG.BOOT

Table 53. 0x80008000 - SYS.CFG.BOOT - Internal boot ROM configuration register

31	30	29	28	27	26	25	24	21	20	16	15	14	13	12	10	9	8	5	4	2	1	0
EM	DE	RE	BY	CB		RE	SEL		DIV		NB	NV	R	BN		C	REM		SRC		AS	CS
0	* 1)	* 1)	* 1)	* 1)			* 1)		0x9		1	1	0	* 1)		* 1)	* 1)		* 1)		* 1)	* 1)
r	rw	rw	rw	rw			rw		rw		rw	rw	rw	rw		rw	rw		rw		rw	rw

31 Reserved

30 Disable EDAC for external memory. The default setting is determined by GPIO[0].

29 Redundant memory available. The default setting is determined by GPIO[63].

28 Bypass of internal boot ROM. This will force the microcontroller to boot from external selected source. The default setting is determined by GPIO[17].

27: 26 CAN Bit Rate (CB[1:0]). The default setting is determined by GPIO[63] (CB[0]) and DUART_TXD (CB[1]).

25 Remote boot mode. The default setting is determined by SPIM_MOSI.

24: 21 PLL Divisor startup value

If remote boot is disabled then default value is 0b0010.

If remote boot is enabled then default setting is determined by DUART_TX and GPIO[63].

20: 16 SpaceWire clock divisor

The register field set the reset value of register RTR.IDIV in the SpaceWire router. This register controls the link-rate during initialization (all states up to and including the connecting-state). For more information see 33.2.21. The default value is 0b01001.

15 Enable second memory controller (NVRAM) when bit is set to ’0’. Only available in PBGA/SIP package option. The default setting is determined by MEM1_CONFIG[1]. Pin strapped to ’1’ by default in CQFP package and shall not be modified.

14 Boot from second memory controller (NVRAM) when bit is set to ’0’. Only available in PBGA/SIP package option. The default setting is determined by MEM1_CONFIG[0]. Pin strapped to ’1’ by default in CQFP package and shall not be modified.

13 Not used

12: 10 CAN-FD/I2C node ID (BN[2:0]). The default setting is determined by GPIO[0] (BN[0]), GPIO[62] (BN[1]) and GPIO[15] (BN[2]).

9 CAN remote boot mode. The default setting is determined by SPIM_MOSI and GPIO[18].

Table 53. 0x80008000 - SYS.CFG.BOOT - Internal boot ROM configuration register

8:	5	Enable remote access interface: 0x0 - None 0x1 - SpaceWire 0x2 - SPI2AHB 0x4 - I2C 0x8 - UART 0xA - CAN All other values are reserved for future boot options. The default setting is determined by SPIM_MOSI, SPIM_SCK and SPIM_SEL.
4:	2	Select external memory to boot from 0x0 - External SPI ROM 0x1 - External SRAM/MRAM 0x2 - External ROM/PROM/EEPROM 0x3 - Reserved 0x4 - Reserved for future boot options 0x5 - Reserved for future boot options 0x6 - Reserved for future boot options 0x7 - Reserved for future boot options The default setting is determined by SPIM_MOSI, SPIM_SCK and SPIM_SEL.
1:		Configure boot ROM to check and use ASW container. The default setting is determined by DUART_TX.
0:		CAN Select line, shows if the nominal or the redundant bus is selected. The default setting is determined by GPIO[1].
Note 1:		The default settings after reset depends on the state of external bootstrap pins. For more details about bootstrap pins and the boot modes they configure, see section 3.1.

7.3 Special Configuration Registers

The special registers are used for getting access to special functions in the LEON3FT microcontroller.

Special functions accessible via special configuration registers:

- Make digital control and status signals for on-chip analog functionality available on the external general inputs and outputs.
- Enable and run Production test on individual embedded memories
- Trigger interrupt test
- Enable external voltage reference

7.3.1 On-chip analog functions

Access to digital control and status signals for integrated analog functionality. Access to control and status for individual analog functions can be configured in the register SYS.CFG.ANA1 and SYS.CFG.ANA2.

Note: Registers in this section are available only in debug mode and must not be modified during normal operation. Please contact Frontgrade Gaisler support if more information is needed.

Table 54. Analog access configuration register

AMBA address	Register	Acronym
0x94002000	Analog test bus 1 configuration	SYS.CFG.ANA1
0x94002004	Analog test bus 2 configuration	SYS.CFG.ANA2
0x94002008	Analog test bus 3 configuration	SYS.CFG.ANA3
0x9400200C	Analog test bus 4 configuration	SYS.CFG.ANA4
0x94002010	Analog test bus 5 configuration	SYS.CFG.ANA5
0x94002014	Analog test bus 6 configuration	SYS.CFG.ANA6
0x94002018	Analog test bus 7 configuration	SYS.CFG.ANA7
0x9400201C	Analog test bus 8 configuration	SYS.CFG.ANA8
0x94002020	Test configuration register	SYS.CFG.ANA9
0x94002024	Bypass test buffer configuration	SYS.CFG.ANA10
0x94002028	Test buffer enable register	SYS.CFG.ANA11
0x9400202C	Select internal comparator	SYS.CFG.ANA12

Table 55. 0x940020xx - SYS.CFG.ANAx - Analog test bus configuration registers

31	0
ANA1	
0x0	
rw	

31: 0 Contact support for information

7.3.2 Memory Test

All memory entities have a build-in test structure for automatic testing. The automatic testing is triggered from software and can only be enabled when the external DSU_EN signal is high. The test is destructive and all memory contents will be overwritten.

The memory test algorithm used is a March C- (evolved March C). The advantage of using the March C- test algorithm is that the algorithm covers many faults models without knowing the internal structure or the layout of the memory. The covered fault models includes Stuck-At, Transition, Coupling, Neighborhood Sensitivity and Address decoding fault.

The disadvantage of using the March C- algorithm is that it is very time consuming due to its nature of checking bit by bit multiple times.

March C- algorithm implemented {↑(w0);↑(r0,w1);↑(r1,w0);↓(r0,w1);↓(r1,w0);↓(r0)}

Notation of the algorithm:

↑ : address 0 bit 0 to address n-1 bit m

↓ : address n-1 bit m to address n bit 0

w0 : write 0 to bit (memory cell) location

w1 : write 1 to bit (memory cell) location

r0 : read a bit (memory cell) value should be 0

r1 : read a bit (memory cell) value should 1

The March C- test algorithm is enabled per memory instantiation by writing to the configuration register SYS.CFG:MEMTEST.

Table 56. Memory test configuration register

AMBA address	Register	Acronym
0x94003000	Configuration register for memory test 0	SYS.CFG.MEMTEST0
0x94003004	Configuration register for memory test 1	SYS.CFG.MEMTEST1
0x94003008	Configuration register for memory test 2	SYS.CFG.MEMTEST2
0x94003080	Status register for memory test 0	SYS.STS.MEMTEST0
0x94003084	Status register for memory test 1	SYS.STS.MEMTEST1
0x94003088	Status register for memory test 2	SYS.STS.MEMTEST2

Table 57. 0x94003000 - SYS.CFG.MEMTEST0 - Memory test configuration register0

31 30 29 28	27 26	25 24	23 22	21 20	19 18	17 16	15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
R	AHBT3	AHBT2	AHBT1	AHBT0	DSU3	DSU2	DSU1	DSU0	TBUF3	TBUF2	TBUF1	TBUF0	RW1	RW0
0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

31: 28 Reserved

27: 26 Trace Memory on MAIN AHB bus (AHBT3 - AHBT0):

25: 24
23: 22
21: 20
0x0 - Not used (Memory bit for memory is kept in reset state)
0x1 - Enable March C- test algorithm
0x2 - Write 0x0 to all locations in memory
0x3 - Not used

19: 18 Trace Memory on DSU (DSU3 - DSU0):

17: 16
15: 14
13: 12
0x0 - Not used (Memory bit for memory is kept in reset state)
0x1 - Enable March C- test algorithm
0x2 - Write 0x0 to all locations in memory
0x3 - Not used

11: 10 Instruction Trace Memory (TBUF3 -TBUF0):

9: 8
7: 6
5: 4
0x0 - Not used (Memory bit for memory is kept in reset state)
0x1 - Enable March C- test algorithm
0x2 - Write 0x0 to all locations in memory
0x3 - Not used

3: 2 LEON3FT register Window 1 memory (RW1):

0x0 - Not used (Memory bit for memory is kept in reset state)
0x1 - Enable March C- test algorithm
0x2 - Write 0x0 to all locations in memory
0x3 - Not used

1: 0 LEON3FT register Window 0 memory (RW0):

0x0 - Not used (Memory bit for memory is kept in reset state)
0x1 - Enable March C- test algorithm
0x2 - Write 0x0 to all locations in memory
0x3 - Not used

Table 58. 0x94003004 - SYS.CFG.MEMTEST1 - Memory test configuration register1

31 30 29 28 27 26 25 24	23 22	21 20	19 18	17 16	15 14 13 12 11 10 9 8	7 6	5 4	3 2	1 0
R	DM3	DM2	DM1	DM0	R	IM3	IM2	IM1	IM0
0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0

Table 58. 0x94003004 - SYS.CFG.MEMTEST1 - Memory test configuration register1

r	rw	rw	rw	rw	r	rw	rw	rw	rw

31: 24 Reserved

23: 22 On-chip data memory test control bits (DM3 - DM0):

21: 20 0x0 - Not used (Memory bit for memory is kept in reset state)

19: 18 0x1 - Enable March C- test algorithm

17: 16 0x2 - Write 0x0 to all locations in memory
0x3 - Not used

15: 8 Reserved

7: 6 On-chip Instruction memory test control bits (IM3 - IM0):

5: 4 0x0 - Not used (Memory bit for memory is kept in reset state)

3: 2 0x1 - Enable March C- test algorithm

1: 0 0x2 - Write 0x0 to all locations in memory
0x3 - Not used

Table 59. 0x94003008 - SYS.CFG.MEMTEST2 - Memory test configuration register2

31 30	29 28	27 26	25 24	23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
RTAD1	RTAI1	RTAD0	RTAI0	R
0x0	0x0	0x0	0x0	0x0
rw	rw	rw	rw	r

31: 30 RTA1 On-chip Data memory test control bits (RTAD1):

0x0 - Not used (Memory bit for memory is kept in reset state)
0x1 - Enable March C- test algorithm
0x2 - Write 0x0 to all locations in memory
0x3 - Not used

29: 28 RTA1 On-chip Instruction memory test control bits (RTAI1):

0x0 - Not used (Memory bit for memory is kept in reset state)
0x1 - Enable March C- test algorithm
0x2 - Write 0x0 to all locations in memory
0x3 - Not used

27: 26 RTA0 On-chip Data memory test control bits (RTAD0):

0x0 - Not used (Memory bit for memory is kept in reset state)
0x1 - Enable March C- test algorithm
0x2 - Write 0x0 to all locations in memory
0x3 - Not used

25: 24 RTA0 On-chip Instruction memory test control bits (RTAI0):

0x0 - Not used (Memory bit for memory is kept in reset state)
0x1 - Enable March C- test algorithm
0x2 - Write 0x0 to all locations in memory
0x3 - Not used

23: 0 Reserved

To minimize the power consumption all memory tests should be executed in sequence. It is still possible to execute all tests in parallel to shorten the test time. The run time is depended upon the number of memory cells in the memory entity. The largest memory entity's are the data (64KiB) and instruction memory (64KiB).

The results and current status can be read in the status register SYS.STS:MEMTESTx. The status register indicates if test is running and if any error was detected during the memory test per memory entity in the LEON3FT microcontroller.

Table 60. 0x94003080 - SYS.STS.MEMTEST0 - Memory test status register0

31 30 29 28	27 26	25 24	23 22	21 20	19 18	17 16	15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
R	AHBT3	AHBT2	AHBT1	AHBT0	DSU3	DSU2	DSU1	DSU0	TBUF3	TBUF2	TBUF1	TBUF0	RW1	RW0
0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

31: 28 Reserved

27: 26 Trace Memory on MAIN AHB bus (AHBT3 - AHBT0):

25: 24 0x0 - No error detected during last test (If test has been run)

23: 22 0x1 - Status for ongoing test, goes low after test completed

21: 20 0x2 - Error during last scan
0x3 - Invalid state and test result

19: 18 Trace Memory on DSU (DSU3 - DSU0):

17: 16 0x0 - No error detected during last test (If test has been run)

15: 14 0x1 - Status for ongoing test, goes low after test completed

13: 12 0x2 - Error during last scan
0x3 - Invalid state and test result

11: 10 Instruction Trace Memory (TBUF3 -TBUF0):

9: 8 0x0 - No error detected during last test (If test has been run)

7: 6 0x1 - Status for ongoing test, goes low after test completed

5: 4 0x2 - Error during last scan
0x3 - Invalid state and test result

3: 2 LEON3FT register Window 1 memory (RW1):

0x0 - No error detected during last test (If test has been run)
0x1 - Status for ongoing test, goes low after test completed
0x2 - Error during last scan
0x3 - Invalid state and test result

1: 0 LEON3FT register Window 0 memory (RW0):

0x0 - No error detected during last test (If test has been run)
0x1 - Status for ongoing test, goes low after test completed
0x2 - Error during last scan
0x3 - Invalid state and test result

Table 61. 0x94003084 - SYS.STS.MEMTEST1 - Memory test status register1

31 30 29 28 27 26 25 24	23 22	21 20	19 18	17 16	15 14 13 12 11 10 9 8	7 6	5 4	3 2	1 0
R	DM3	DM2	DM1	DM0	R	IM3	IM2	IM1	IM0
0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
r	rw	rw	rw	rw	r	rw	rw	rw	rw

31: 24 Reserved

23: 22 On-chip data memory test status bits (DM3 - DM0):

21: 20 0x0 - No error detected during last test (If test has been run)

19: 18 0x1 - Status for ongoing test, goes low after test completed

17: 16 0x2 - Error during last scan
0x3 - Invalid state and test result

15: 8 Reserved

7: 6 On-chip Instruction memory test status bits (IM3 - IM0):

5: 4 0x0 - No error detected during last test (If test has been run)

3: 2 0x1 - Status for ongoing test, goes low after test completed

1: 0 0x2 - Error during last scan
0x3 - Invalid state and test result

Table 62. 0x94003088 - SYS.STS.MEMTEST2 - Memory test status register2

31 30	29 28	27 26	25 24	23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
RTAD1	RTAI1	RTAD0	RTAI0	R
0x0	0x0	0x0	0x0	0x0
rw	rw	rw	rw	r

31: 30 RTA1 On-chip Data memory test status bits (RTAD1):

0x0 - No error detected during last test (If test has been run)
0x1 - Status for ongoing test, goes low after test completed
0x2 - Error during last scan
0x3 - Invalid state and test result

29: 28 RTA1 On-chip Instruction memory test status bits (RTAI1):

0x0 - No error detected during last test (If test has been run)
0x1 - Status for ongoing test, goes low after test completed
0x2 - Error during last scan
0x3 - Invalid state and test result

27: 26 RTA0 On-chip Data memory test status bits (RTAD0):

0x0 - No error detected during last test (If test has been run)
0x1 - Status for ongoing test, goes low after test completed
0x2 - Error during last scan
0x3 - Invalid state and test result

25: 24 RTA0 On-chip Instruction memory test status bits (RTAI0):

0x0 - No error detected during last test (If test has been run)
0x1 - Status for ongoing test, goes low after test completed
0x2 - Error during last scan
0x3 - Invalid state and test result

23: 0 Reserved

7.3.3 System configuration register

This register can be used to test system, change system error behavior or enable special system functions e.g. interface loopback functionality.

The interrupt test is accessible to the system in all functional modes. A protection scheme has been added to the interrupt test functionality in order to prevent erroneous accesses to the functionality. The generated interrupt event will be inserted into the interrupt controller and the intention is to test interrupt controller and interrupt software.

The interrupt test control register contains a interrupt number bit field and two protection bits. The two protection bits are used as protection and enable bits for the interrupt test. When the protection bits are toggled an interrupt event is asserted to the interrupt controller.

Table 63. Interrupt test configuration register

AMBA address	Register	Acronym
0x8000E000	System Configuration register	SYS.CFG.SCFG
0x94006000	DSU Break configuration	SYS.CFG.BREAKCFG
0x94007000	DSU Break enable	SYS.CFG.BREAKEN

Table 64. 0x8000E000 - SYS.CFG.SCFG - System Configuration register

31 29	28	27 23	22	21	20 18	17 16	15	14 13	12	11	10 9	8 3	2	1	0
RES	DI	TIOSEL	MD	MO	RES	SPW	LL	LS	LE	FS	PR	IRQ	MR	WE	EE
0x0	0x0	0x0	0x1	*	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x1	0x1	*
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

31: 29 Reserved

28 DAC production test input (DI) - Control for proprietary test function.

27: 23 Analog production test output selection (TIOSEL) - Control for proprietary test function.

22 Internal memory IO direction (MD) - See description of MO field.

21 Override internal memory IO direction (MO) - If set to 1 then the direction of the internal memory IO is determined by the MD field of this register. The reset value is 1 for GR716B in the CQFP package. The reset value for PBGA/SIP package depends on the pin MEM1_CONFIG(1), refer table 202.

20: 18 Reserved

17: 16 SpaceWire Loop-back control (SPW) - Control of SpaceWire internal loop-back

Bit #17 - Enable internal loop-back for SpaceWire PHY 1 (SpaceWire router port 2)
Bit #16 - Enable internal loop-back for SpaceWire PHY 0 (SpaceWire router port 1)

Internal loop-back means that the ports internal data and strobe signals are not mapped to the corresponding external SpaceWire I/O pins. They are instead routed back to the port internally (transmit data to receive data, transmit strobe to receive strobe).

15 LVDS External Loop (LL) - Enable LVDS external loop-back

External loop-back means that the external LVDS I/O pins are not routed to the corresponding port. Instead they are asynchronously routed back out on the external pins (LVDS_RXp/n to LVDS_TXp/n) according to the mapping below:

LVDS_RX[0]p/n - Looped back on LVDS_TX[2]p/n and LVDS_TX[4]p/n
LVDS_RX[1]p/n - Looped back on LVDS_TX[0]p/n
LVDS_RX[2]p/n - Looped back on LVDS_TX[1]p/n
LVDS_RX[3]p/n - Looped back on LVDS_TX[3]p/n and LVDS_TX[5]p/n

Enabling of external loop-back forces all LVDS receivers and transmitters to be enabled. Note that in the Revision0 of CQFP package, LVDS_RX[0]p/n and LVDS_TX[2]p/n are bonded to the same pair of package pins.

14: 13 SpaceWire External Loop (LS) - SpaceWire external loop-back

0x0 - Normal operation
0x1 - External loop-back mode routed back via rising edge clocked flip-flops
0x2 - External loop-back mode routed back via falling edge clocked flip-flops
0x3 - External loop-back mode routed back via rising or falling edge clocked flip-flops

SpaceWire External loop-back means that the input signals from each SpaceWire port is passed through sampling flip-flops in the corresponding SpaceWire-Phy and returned to the output signals of the same SpaceWire port. to the corresponding port (SPW_RXDp/n to SPW_TXDp/n and SPW_RXSp/n to SPW_TXSp/n). See section 2.5 for mapping between LVDS pins and SpaceWire signals.

Test option 0x1 and 0x2 are used for setup and hold measurements for respective clock edge. Test option 0x3 is used for minimum pulse width detection.

12 Locken (LE) - Support Locked transfers for scrubber.

11 Force Scrubber (FS) - Forces the scrubber to use the main AMBA bus.

0: Memory scrubber and FPGA scrubber have dedicated AMBA bus connections to the FTMCTRL0, FTMCTRL1, SPIMCTRL0, and SPIMCTRL1 memory controllers.

1: Memory scrubber and FPGA scrubber are connected through the main AMBA bus.

10: 9 Interrupt test protection bits (PROT) - Protection and generation of interrupt test for specified interrupt source.

An interrupt will be generated on the interrupt line encoded by the IRQ field of this register, if both bits of the PROT field are simultaneously written with the inverse of their previous value. E.g. a test interrupt can be generated by reading this register, inverting the two PROT bits, and writing back the modified value.

8:3 Interrupt source (IRQ) - These 6 bits encode an integer in the range 0-63. And interrupt will be generated on the interrupt line with this number upon a write to this register if the two PROT bits are inverted compared to their previous value.

2 MIL-1553B reset disable (MR) - Set to 1 to disable reset signal from MIL-1553B core. Reset value: 1

Table 64. 0x8000E000 - SYS.CFG.SCFG - System Configuration register

1 Override error mode reset generation (EE) - Set to 1 to disables reset generation from processor error mode. Reset value: 1

0 Override watchdog reset generation (WE) - Set to 1 to disable internal reset from watchdog timeout. Reset value depends on DSU_EN. Resets to 1 when DSU_EN is high, resets to 0 when DSU_EN is low.

Table 65. 0x94006000 - SYS.CFG.BREAKCFG - DSU Break configuration

31 14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	R1R	R0R	R	R	R1R0	R1UC	R	R0R1	R	R0UC	R	UCR1	UCR0	R
0x0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

31: 14 Reserved

13 Unused (R1R)

12 RTA0 and RTA1 Power down after reset (R0R)

11 Reserved

10 Reserved

9 Break RTA1 on RTA0 (R1R0)

8 Break RTA1 on LEON3FT (R1UC)

7 Reserved

6 Break RTA0 on RTA1 (R0R1)

5 Reserved

4 Break RTA0 on LEON3FT (R0UC)

3 Reserved

2 Break LEON3FT on RTA1 (UCR1)

1 Break LEON3FT on RTA0 (UCR0)

0 Reserved

Table 66. 0x94007000 - SYS.CFG.BREAKEN - DSU Break enable

31 6	5	4	3	2	1	0
R	RC1	RC0	UC	RB1	RB0	UB
0x0	0	0	0	0	0	0
r	rw	rw	rw	rw	rw	rw

31: 6 Reserved

5 RTA1 Microcontroller Clear Now (RC1)

4 RTA0 Microcontroller Clear Now (RC0)

3 LEON3FT Microcontroller Clear Now (UC)

2 RTA1 Microcontroller Break Now (RB1)

1 RTA0 Microcontroller Break Now (RB0)

0 LEON3FT Microcontroller Break Now (UB)

8 Reset generation, brownout detection and analog system configuration registers

8.1 Overview

The Reset Generation and Brownout detection provides the system with a reset signal, deterministic startup behavior during power-on of the system and detection of power supply failure on board level.

The generated reset is output on a 3.3V IO to be used in the system.

The brownout monitors are enabled and configured via a register interface for analog system configuration. The same register interface allows trimming the LDO voltage, enabling or disabling the VREFBUF precision voltage output, trimming of ADC linearity (added in silicon revision 1), and readout of LVDS failsafe and brownout monitor state.

8.2 Operation

8.2.1 Overview of POR and brownout

The Reset generation and Brownout detection consists of two analog functions: The Power On Reset (POR) and the Brownout detection (BO).

The Brownout detectors will monitor the 1.8V and all 3.3V supplies. At the event of crossing a Brownout threshold, an interrupt will be generated. When such an interrupt is detected, the software needs to take action, typically shutting down critical parts of the system in a well controlled way. The time from detection of supply brownout to activation of system reset is determined by the external power supplies capability to maintain the supply voltages (the amount of decoupling capacitance on PCB).

8.2.2 Reset generation

An internal reset signal is generated from level detection with hysteresis of the core supply voltage, V_{DD_CORE}. When this supply is below the low threshold, the internal reset signal is low. When the supply goes above the high threshold, the internal reset is still kept low until the reset release time has passed, then it goes high. The release time is configurable by external capacitor C_RST. See datasheet section 2.11 [DS] for voltage thresholds and delay parameters.

An internal reset can also be generated if there is a watchdog timeout, if the main CPU enters error mode, and if a reset command is received via MIL-STD-1553B. These three sources of internal reset can be individually enabled or disabled by the user though the SYS.CFG.SCFG register described in section 7.3.3. When an internal reset is triggered by one of these three sources, internal reset will remain asserted for 2^{17}=131072 system clock cycles before deasserting.

Watchdog timeout reset: The watchdog timer unit drives a signal to request restart of the system. Watchdog functionality is described in section 35. This source of reset is enabled at reset in flight configuration (DSU_EN low), but disabled at reset in debug configuration (DSU_EN high). Software may enable or disable this source of reset using the SYS.CFG.SCFG control register described in section 7.3.3.

Main CPU error mode reset: In general, error mode is a state entered by a SPARC V8 processor if a trap occurs while traps are disabled. A LEON3 processor halts when it enters error mode, but also signals this state via an internal signal. See [SPARC] and sections 17.2.10 and 20.2. In the GR716B the error mode status signal of the main CPU is a source for internal reset. This source of reset is disabled at reset, but may be enabled from software during or after boot as described in section 7.3.3. An RTA can also enter error mode, but RTA error state is not used for reset generation. Error mode can be forced via a write to the error mode status register in the interrupt controller, see section 40.2.7.

MIL-STD-1553B triggered reset: When the GR716B is acting as Remote Terminal (RT), it can be configured to generate an internal reset upon reception of the mode code “Reset Remote Terminal”. To enable this functionality refer to chapter 24 which describes the MIL-STD-1553B interface, sec-

tion 24.5.3 describes the respective mode code. This source of reset is disabled at reset, but may be enabled from software after boot through the control register described in section 7.3.3.

Besides internal reset generation, there is an external reset input signal, RESET_IN_N which is combined with internal reset sources to generate the internal reset signal. This input has hysteresis and rejects pulses shorter than a minimum width. See datasheet section 2.11 [DS] for electrical characteristics.

The internal reset sources and the external reset input signal RESET_IN_N input are asynchronous to all clocks. External pins (with the exception of LVDS transmitters) and analog functions will be in a well-defined state when either of the reset signals is low. However, most logic internal to the GR716B resets synchronously to the system clock and requires 1 to 10 positive edges of the system clock to reach a well-defined. And after reset goes high, an additional 30 clock cycles are needed to complete the reset. Also note that some internal logic is not reset at all. In particular, this is the case for all internal RAM.

The pin RESET_OUT_N output is a buffered copy of the internal reset signal, see datasheet section 3.2 [DS] for electrical characteristics. It is inactive high, active tri-state with internal pull-down. RESET_OUT_N will also briefly be actively driven low prior to reset deassertion.

The Brownout detector on the supplies, V_{DD_CORE}, V_{DD_IO}, V_{DDA_PLL}, V_{DDA_ADC}, V_{DDA_DAC}, V_{DD_LVDS}, V_{DDA_REF}, have individually programmable threshold levels. The threshold selected for each supply must, in worst case, be set below the guaranteed minimum supply voltage instant peak level provided on the package pins for each supply, respectively. Otherwise, undesired Brownout detections giving inadvertent system shutdowns can result, see datasheet section 2.11 [DS] for electrical characteristics.

Furthermore, for any supply voltage in the system that is equipped with a reset threshold detection, such as the on-chip V_{DD_CORE} detector (or any arbitrary supply detector on PCB), the Brownout level must be set with a certain margin higher than the reset level, such that there is enough time to ensure that the Brownout interrupt routine can be executed before the reset is activated. Therefore, the selection of the Brownout threshold levels should be extra carefully co-designed with the power-supply and reset designs on PCB, in Microcontroller applications that will utilize the Brownout detectors on supply voltages that are also reset detected (which V_{DD_CORE} always is).

The Brownout detector status can be read from status registers. It is also forwarded to the Alarm Matrix where it can be configured to generate an interrupt for the processor for software handling, and/or configured to generate an external alarm (on a GPIO pin) or shutdown signal for immediate response for applications where software interrupt handling is too slow or unreliable.

After power-on reset, the Microcontroller starts with all Brownout interrupt mask bits set to enable, but with all programmable-level brownout detector blocks disabled. (TBC)

8.2.3 Reset IO control

The 64 General purpose IO described in chapter 2.4 and 2.5 is forced to high impedance mode when core voltage supply goes below the reset ramp down threshold, see datasheet section 2.11 [DS] for electrical characteristics.

8.2.4 LDO trim control

The output voltage of the LDO described in chapter 13 can be trimmed to 8 discrete levels via a register in this block. At reset, the register configures the maximum voltage trim level.

8.2.5 ADC0-2 linearity trimming

GR716B revision 1 includes a trim register, ADCTRM, for improving the linearity of ADC0-2. This register must be written with the value TBD before starting sampling to reach the linearity specification in datasheet section 2.8 [DS]. The register resets to 0. ADC3 is not affected by this register setting.

8.2.6 LDOANA power down

The internal LDO that provides power to the V_{DDA_PLL} supply rail can be disabled via a register in this block. Note that this is a distinct internal LDO than the one mentioned in section 8.2.4. See the nodes for V_{DDA_PLL} in datasheet section 2.1, 2.2 and 3.2 [DS] for usage information.

8.2.7 VREFBUF enable

The voltage reference output buffer VREFBUF described in chapter 11 can be enabled and disabled (powered down) using a register in this block. The VREFBUF is always disabled at reset. To use the VREFBUF, it must first be enabled by writing to this register.

8.2.8 Access control

The reset release time is programmable by an external capacitor, C_RST. Capacitor value and release time is specified in datasheet section 2.11 [DS]

Brown Out detection level and interrupt generation can be controlled via registers.

8.3 Registers

The Reset Generation and Brownout Detection is programmed through registers mapped into APB address space.

Table 67. Reset Generation and Brownout Detection status and control registers

AMBA address	Register	Acronym
0x8010C000	Brownout detection configuration register	BOCFG
0x8010C004	LDO trim register	LDOTRM
0x8010C008	RESERVED (TBC)	N/A
0x8010C00C	ADC0-2 linearity trim register	ADCTRM
0x8010C010	RESERVED (BC)	N/A
0x8010C014	LDOANA power-down control register	LDOANACFG
0x8010C018	VREFBUF power-down control register	VREFBUFCFG
0x8010C01C-0x8010C06C	RESERVED	N/A
0x8010C070	Interrupt status register	
0x8010C074	Interrupt flag register	
0x8010C078	Interrupt mask register	
0x8010C07C	Interrupt edge register	
0x8010C080	Brownout detection status register	BOSTS
0x8010C084	LVDS receiver fail-safe detection status register	FSSTS
0x8010C088-0x8010C0FC	RESERVED	N/A

Table 68. 0x8010C000 - BOCFG - Brownout detection configuration register

31 30 29 28	27	26	25	24	23	22	21	20 18	17 15	14 12	11 9	8 6	5 3	2 2 0
RES	DI	DC	DA	DD	R	DL	DP	BLI	BLC	BLA	BLD	BLB	BLL	BLP
0	1	1	1	1	1	1	1	0b000	0b000	0b000	0b000	0b000	0b000	0b000
r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

31: 28 Reserved

27 Disable brownout detector for VDD_IO (DI)

26 Disable brownout detector for VDD_CORE (DC)

25 Disable brownout detector for VDDA_ADC and VDDA_REF (DA)

24 Disable brownout detector for VDDA_DAC (DD)

23 Reserved

22 Disable brownout detector for VDD_LVDS (DL)

21 Disable brownout detector for VDDA_PLL (DI)

0 to enable brownout detector, 1 to disable brownout detection (powers down the detector).

20: 18 Brown Out Level for VDD_IO power supply (BLI)

17: 15 Brown Out Level for VDD_CORE power supply (BLC)

14: 12 Brown Out Level for VDDA_ADC supply (BLA)

11: 9 Brown Out Level for VDDA_DAC supply (BLD)

8: 6 Brown Out Level for VDDA_REF supply (BLB)

5: 3 Brown Out Level for VDD_LVDS supply (BLL)

2: 0 Brown Out Level for VDDA_PLL supply (BLP)

Level 0b000 sets the minimum value for the threshold, and 0b111 the maximum. See datasheet section 2.11 [DS] for electrical characteristics of the brownout levels.

Table 69. 0x8010C004 - LDOTRM - LDO Trim register

31 3	2 0
RESERVED	TRM
0	0b011
r	rw

31: 3 RESERVED

2 :0 LDO trimmer value (TRM). Refer to datasheet section 2.12 [DS] for the mapping between trimmer value and output voltage.

Table 70. 0x8010C00C - ADCTRM - ADC0-2 linearity trim register

31 3	2	1	0
RESERVED	R	TP	TN
0	0	0	0
r	rw	rw	rw

31: 3 RESERVED

2 This bit is writable and readable. The value written has no effect.

1 P-side ADC linearity trim value (TP) - This bit must be set to TBD to achieve the linearity of ADC0-2 specified in datasheet section 2.8 [DS].

0 N-side ADC linearity trim value (TN) - This bit must be set to TBD to achieve the linearity of ADC0-2 specified in datasheet section 2.8 [DS].

Table 71. 0x8010C014 - LDOANACFG -LDOANA power-down register

31 … 14	13 … 12	11 … 10	9 … 8	7 … 6	5 … 4	3 … 2	1	0
RESERVED								DIS
0x00000000								0
r								rw

31: 1 RESERVED

0 Disable LDOANA (DIS) - Writing 1 to this bit disables the internal LDO that provides power to the VDDA_PLL supply rail.

Table 72. 0x8010C018 - VREFBUFCFG - VREFBUF control register

31 … 3	2	1	0
RESERVED			EN
0x00000000			0
r			rw

31: 1 RESERVED

0 Enable VREFBUF (EN) - 1 to enable the VREFBUF output, 0 to disable (power down)

Table 73. 0x8010C080 - BOSTS - Brownout Detection status register

31 … 6	5	4	3	2	1	0
RESERVED	BP	BL	BB	BD	BA	BC	BI
0x00000000	0	0	0	0	0	0	0
r	r	r	r	r	r	r	r

31: 7 RESERVED

6 Brown Out Detected (BP) - V_{DDA_PLL} brownout detector output

5 Brown Out Detected (BL) - V_{DD_LVDS} brownout detector output

4 Brown Out Detected (BB) - V_{DDA_REF} brownout detector output

3 Brown Out Detected (BD) - V_{DDA_DAC} brownout detector output

2 Brown Out Detected (BA) - V_{DDA_ADC} brownout detector output

1 Brown Out Detected (BC) - V_{DD_CORE} brownout detector output

0 Brown Out Detected (BI) - V_{DD_IO} brownout detector output

1 when the brownout detector is enabled and has detected a supply voltage below the programmed threshold. 0 otherwise. Note that the ordering of supplies is different in this register compared to the BOCFG register.

Table 74. 0x8010C084 - FSSTS - LVDS receiver fail-safe detection status register (TBC)

31 … 4	3	2	1	0
RESERVED	F3	F2	F1	F0
0x00000000	0	0	0	0
r	r	r	r	r

31: 4 RESERVED

3 LVDS receiver fail-safe condition detected on RX3 (F3)

2 LVDS receiver fail-safe condition detected on RX2 (F2)

1 LVDS receiver fail-safe condition detected on RX1 (F1)

0 LVDS receiver fail-safe condition detected on RX0 (F0)

1 when the fail-safe detection is enabled and has detected a sub-threshold input signal, 0 otherwise. See datasheet section 2.5 [DS] for electrical characteristics of the fail-safe detection function.

9 Crystal Oscillator (XO)

9.1 Overview

The internal crystal oscillator (XO) contains all *active* oscillator parts, and a crystal (XTAL) is added on PCB on XO_X1 and XO_X2 pins. The clock output pin, XO_OUT, provides a CMOS square-wave output signal.

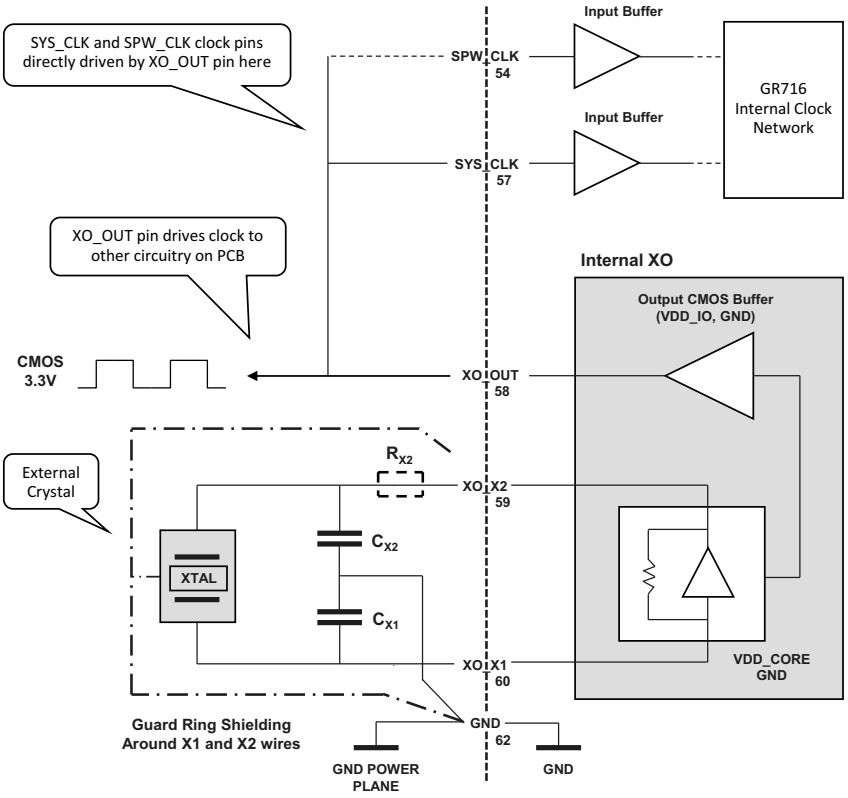


Figure 17. Connection diagram how to use the internal crystal oscillator (XO) as GR716B clock source.

9.2 Operation

9.2.1 System overview

The internal XO implementation supports a general class of crystals, see the following sections. Based on the selected crystal, an accurate and stable frequency is provided on the oscillator output, XO_OUT, which can be directly connected to the GR716B clock inputs. This output clock signal can also be used to other circuitry on PCB, and in case of driving an impedance matched clock net on PCB, e.g. 50Ω, it is recommended to add a clock driver buffer on PCB with adequate drive capability.

9.2.2 Detailed description

The internal XO is supplied by the microcontroller internal core supply, V_{DD_CORE}, and the oscillator output is available on an external pin, XO_OUT (CMOS), supplied by V_{DD_IO}. The external crystal, to be connected to the XO pins (XO_X1 and XO_X2), should be a parallel-resonant fundamental-tone AT-cut crystal. Also a load capacitor to ground on each of these two pins are required on PCB. See the following sections for functional crystal configurations and PCB detailed implementation of the XO interface.

The range of supported frequencies is 5-25 MHz, where down to 5MHz would be recommended in applications where the XO power consumption itself is required to be low (<1mA). Special applica-

tion cases, which need a low-noise clock directly connected from XO_OUT to the application circuitry on PCB, may need the XO to run at a specific frequency for the application, e.g., the analog precision PWM application, APWM_DAC, in chapter 55. In general, any XO frequency within 5-25 MHz can be implemented with the internal XO, and the internal PLL supports five discrete input frequencies across this range, and generates higher internal clock frequencies than 25MHz to be used in GR716B clock configurations internally.

Typical start-up times for XO_OUT, after V_{DD_CORE} is within recommended operating conditions, is in the order of 1ms, and can spread from about 0.2-5 ms for the different crystal configuration examples in table 75. All of these crystal configurations start within 15ms under all operational conditions.

Current consumption for the internal XO is typically 0.5-3 mA, for these different crystal configuration examples. Temporary current during XO start-up is typically 10mA (max 20mA), and does not depend strongly on crystal configuration. In cases where PCB applications require low noise clock directly from XO_OUT, it is recommended to design and decouple the XO supply voltage, V_{DD_CORE}, to have maximum ripple voltage of about $20mV_{p-p}$.

In special applications where the internal XO needs to be replaced by an external, extremely good oscillator such as TCXO or even OCXO, a 3.3V CMOS compatible oscillator signal needs to be provided to the clock input(s) of GR716B. To minimize the internal XO current consumption when it is not used, a detector is build-in to shut it down when both XO pins (XO_X1 and XO_X2) are pulled to ground via 10kΩ each.

9.2.3 Crystal recommendations and examples

This section presents general properties for a crystal (XTAL) to function properly with the internal XO, and a number of crystal configuration examples are listed in the following table.

The frequency of the crystal can be any value from 5-25 MHz. To work as input frequency to the internal PLL, it needs to be 5, 10, 12.5, 20 or 25 MHz, see chapter 10. Furthermore, it always needs to be of the type *parallel-resonant fundamental-tone AT-cut* crystal.

General recommendations for ESR in the crystal, in combination with C_{X1} and C_{X2} (ceramic NP0 or equivalent), at 25MHz, to guarantee XO oscillation:

- Minimum C_{X1} and C_{X2} is always 10 pF_{nom}
- Maximum C_{X1} and C_{X2} is 22 pF_{nom}, for worst-case maximum ESR of 100 Ω
- Maximum C_{X1} and C_{X2} is 33 pF_{nom}, for worst-case maximum ESR of 50 Ω
- Maximum C_{X1} and C_{X2} is 47 pF_{nom}, for worst-case maximum ESR of 30 Ω

A tolerance of 20% on the above nominal capacitance values, and up to 10pF PCB parasitic capacitance per wire to ground, is taken into account for guaranteed oscillation. If the PCB parasitic capacitance is significantly larger than 10pF, the C_{X1} and C_{X2} values are to be decreased accordingly. Lower frequency than 25MHz allows for higher crystal ESR_{Max} limit, where half the frequency will allow double the ESR_{Max} limit.

Table 75. GR716B crystal configuration examples.

Crystal (XTAL)	Frequency [MHz]	Max ESR [Ω]	Load cap (diff.) 1) [pF]	Nominal C_{X1} & C_{X2} 1) [pF]
HC49US / U-Sxxx (Citizen)	5 25	150 , at 25°C 50 , at 25°C	20 15	33 22
ABxxx (Abracon)	10 25	50 , at 25°C 50 , at 25°C	18 18	33 22
JXS32-WA (Jauch)	20	45 , at 25°C	10	15
T1507 ESCC 3501/019 (Rakon)	5 2) 10	25 , full temp. 25 , full temp.	30 30	47 47
T807 ESCC 3501/018 (Rakon)	20 25 3)	25 , full temp. 25 , full temp.	30 30	47 47

Note 1: The load capacitance for a crystal is defined to be the differential capacitance that shall be connected between the two crystal terminals on PCB, for the crystal to oscillate correctly at its specified frequency. The total load capacitance is formed by C_{X1} & C_{X2}, the PCB stray capacitance, and the input capacitance of the XO_X1 & XO_X2 pins. Each XO pin has typically 4pF to ground. E.g., a crystal that needs a load capacitance of 20pF, where the PCB stray capacitance is 3pF per wire, will give C_{X1}=C_{X2} = 2x$C_{Load,diff}$ - 4pF - 3pF = 33pF_{typ}.

Note 2: Example of crystal specification: Rakon STC5856, ESCC3501/019 C0164, for -40 to 95 °C

Note 3: Example of crystal specification: Rakon STC6131, ESCC3501/018 C0318, for -40 to 95 °C

In some crystal configuration cases, where the ESR is order(s) of magnitude lower than the guaranteed maximum limit for oscillation presented above, the oscillation amplitude can become too large from voltage stress point of view on the XO pins (XO_X1 and XO_X2). If the oscillation voltage peaks go outside any of the supply rails in worst-case conditions, the device long-term reliability cannot be guaranteed. Therefore, when the oscillation amplitude is higher than about $0.9V_{pk\text{-}to\text{-}pk}$ on XO_X1 in room temperature, such a crystal configuration needs a resistor, R_{X2}, inserted in-between XO_X2 and C_{X2}, with a value commonly in the order of 0.1-1 kΩ. Either a trim-potentiometer can be inserted temporarily, or a fixed resistor with first trial value of 100Ω can be inserted in the R_{X2} position. This value is increased, e.g., in steps of about 100Ω, until the AC voltage on XO_X1 pin decreases below $0.9V_{pk\text{-}to\text{-}pk}$ typically in room temperature.

For the oscilloscope measurement to not load the XO_X1 node noticeably, which could affect the measured amplitude, this measurement needs to be done with a low-capacitance active probe (ca 2pF). Alternatively, it can be done by a temporary capacitive divider, e.g., by inserting a large capacitance, C_{Large} (ca 100-300 pF), in series with the ground connection of C_{X1}. Then, a standard oscilloscope probe (ca 20pF) can be used to measure across C_{Large}, and the measured amplitude can be recalculated according to the division factor resulting from C_{Large} and C_{X1}.

9.2.4 PCB design considerations

This section lists PCB layout considerations how to connect the external components, see figure 18:

- Place C_{X1} *very close* to XO_X1 pin, and connect ground by a wire to GND pin 62 (connect by individual via to pin 62), see figure 17. In addition to acting crystal load capacitance, C_{X1} will also act as low-pass filter for this XO input for any external disturbances onto the X1 wire (which may have to be rather long in some layout implementations depending on where the crystal needs to be placed).
- Place C_{X2} close to XO_X2 pin, and connect ground by a wire to GND pin 62 (use same via to pin 62 as for C_{X1}), or connect the wire to C_{X1} ground terminal. If series resistor, R_{X2}, is to be used it is placed close to XO_X2 pin, and C_{X2} is placed close to R_{X2}.
- Place the crystal as close to XO_X1 and XO_X2 pins as feasible. However, the crystal may have placement requirements from mechanics point of view, which may have to take priority due to vibration requirements, etc. Anyhow, the crystal must be placed over the same PCB ground plane as used for GR716B GND, to be able to fulfill the guard ring requirements below.
- Route no longer signal traces than necessary from crystal to XO_X1 and XO_X2 pins. Use GND shielding all the way around these two wires (GND guard ring on both sides, above and under each wire), to suppress crosstalk in-between XO_X1 and XO_X2. Crosstalk to XO_OUT and other PCB nets shall also be suppressed. In general, parasitic capacitance from XO_X1 or XO_X2 to any PCB net other than GND must be avoided in XO layout implementations. Since parasitic capacitance to GND can be compensated by deceasing the value of C_{X1} and C_{X2}, it is not harmful at all, as long as this segment of the GND net is *not electrically disturbed*. However, it should be noted that the oscillation frequency may be slightly shifted from the nominal frequency specified for the crystal, due to the significant inductance in the long wires.
- To avoid problems with electrically disturbed GND net, e.g., high supply currents flowing through the GND plane causing AC magnetic-field disturbance onto the X1 and X2 wires, the whole guard ring layout implementation needs to be GND partial planes/wires that are cut out from the general GND plane(s) in the PCB. Moreover, this whole guard ring implementation should be grounded in one single layout point, which should be GND pin 62 (use same via to pin 62 as for C_{X1}), or connect to C_{X1} or C_{X2} ground terminal.

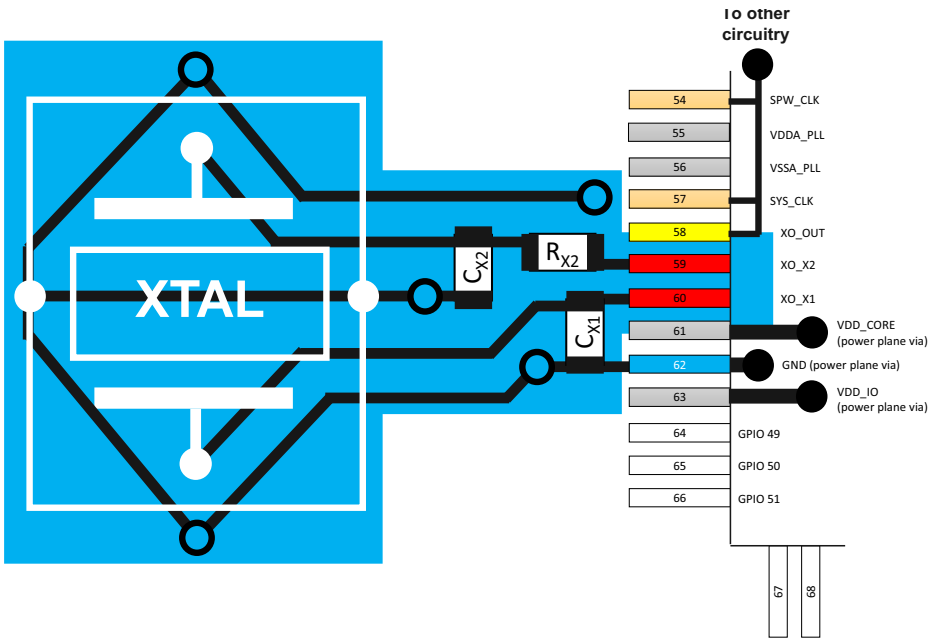


Figure 18. Example of recommended PCB layout

When the crystal is placed as close to the microcontroller as in figure 18, the black wires to the crystal (XTAL) can be routed all the way in the PCB *top* layer. Then, the blue plane, only grounded through pin 62 (also blue), is a cut-out ground plane in the layer directly under the black wire layer. When the crystal needs to be placed further away, the black wires should be routed in a PCB *inner* layer (vias placed directly to the left of C_{X1} and C_{X2}). Then, the blue ground plane shape should be put in two layers, directly under and above the black wire layer. But keep in mind that the nominal oscillation frequency may be slightly shifted when these wires become long.

9.2.5 Access control

N/A

10 PLL

10.1 Overview

The Phase-Lock-Loop (PLL) is capable of generating an phase locked output clock of 400MHz to the system. The input reference clock is multiplied by 2, 4, 8, 16, 20, 32, 40 or 80.

10.2 Operation

10.2.1 System overview

The PLL provides a 400MHz internal clock, typically used as SpaceWire clock, etc. The PLL reference-clock input is a 3.3V CMOS input, to which the XO-oscillator clock output can be directly connected, or any other clock signal generated on PCB fulfilling the electrical specification of this input. The PLL reference-clock input is allowed to be asynchronous to any other clocks in the GR716B microcontroller.

10.2.2 Detailed description

For more information about using the PLL in the system see section 4.

10.2.3 Access control

PLL status and configuration can be accessed via registers

10.3 Registers

Table 76. PLL control and status registers

APB address offset	Register
0x8010D000	PLL Power Down configuration
0x8010D004	Reserved
0x8010D008	Select SpaceWire clock source and divisor
0x8010D00C	Select 1553B clock source and divisor
0x8010D010	Select FPGA Scrubber clock divisor
0x8010D014	Reserved
0x8010D018	Test clock enable
0x8010D01C	Reserved
0x8010D020	PLL Ring oscillator enable
0x8010D070	PLL Interrupt register
0x8010D074	PLL Interrupt status register
0x8010D078	PLL Interrupt mask
0x8010D07C	PLL Interrupt edge
0x8010D080	PLL Status
0x8010D0E0	PLL Local register interface information
0x8010B000	Select system clock source
0x8010B004	Enable system clock error detector

Table 77. 0x8010D000 - CFG - PLL configuration registers

31	5	4	3	2	1	0
RESERVED		AL	PD	CFG		

Table 77. 0x8010D000 - CFG - PLL configuration registers

0x00000000	0	*	*
r	rw	rw	rw

31: 4 Reserved

4 Enable PLL alarm - Enable bit to include PLL in alarm matrix. Refer section 53.7 for description of alarm matrix.

3 PLL power down (PD) - If this bit is written to 1, the PLL is powerdown. The PLL should always be in power down mode when not used, i.e.,when the PLL is bypassed.
* This register can be changed after reset value set by bootstrap pins

2: 0 PLL configuration (CFG) - Internal PLL multiplier depended upon the input frequency of the PLL, set this register to
000b - when input frequency 200 MHz
001b - when input frequency 100 MHz
010b - when input frequency 50 MHz
011b - when input frequency 25 MHz
100b - when input frequency 12.5MHz
101b - when input frequency 20 MHz
110b - when input frequency 10 MHz
111b - when input frequency 5 MHz
* This register can be changed after reset value set by bootstrap pins

Table 78. 0x8010D008 - SPWREF - Select reference for SpaceWire clock

31 24	23 16	15 10	9	8	7 0
RESERVED	DUTY	RESERVED	R	EN	DIV
0x0	0x0	0x0	0x0	0x0	0
r	rw	r	rw	rw	rw

31: 24 Reserved

23: 16 DUTY cycle for generated clock frequency - The duty cycle bitfield specifies how many of the total clock cycles specified in the DIV bitfield the generated clock shall be high. IF this register is set to 0x0 the duty cycle will be set to clock cycles defined in DIV and the clock period to 2xDIV.

15: 10 Reserved

9: Reserved

Table 78. 0x8010D008 - SPWREF - Select reference for SpaceWire clock

8: Enable (EN): When enabled the PLL and divider block output drives the SpaceWire clock. To have SpaceWire clock driven directly from the SPWCLK external pin disable this register and set the SPWREF.DIV register Zero.

7: 0 SpaceWire Reference Clock Divisor (DIV) - Set the divisor for input reference clock. Zero (default) bypass the divisor.

When bitfield DUTY period is set to 0x0 or 0x1. The input clock frequency (PLL output, 400 MHz) will be divided by 2xDIV clock cycles with the duty cycle set to 50%. Valid configurations when DUTY period is set to 0 or 1:

0x00 - Bypass i.e. input frequency is divided by 1
0x02 - Divide input frequency by 4
0x04 - Divide input frequency by 8
0x06 - Divide input frequency by 12
0x08 - Divide input frequency by 16
0x0A - Divide input frequency by 20
0x0C - Divide input frequency by 24
0x0E - Divide input frequency by 28
0x10 - Divide input frequency by 32
0x14 - Divide input frequency by 40
0x16 - Divide input frequency by 44
0x18 - Divide input frequency by 48
0x1A - Divide input frequency by 52
0x1C - Divide input frequency by 56
0x1E - Divide input frequency by 60

All other combinations are not valid.

When bitfield DUTY period is equal or greater then 0x2.The DIV bifield will divide the input frequency by DIV clock cycles and with the duty cycle defined in the DUTY bitfield.

0x04 - Divide input frequency by 4
0x06 - Divide input frequency by 6
0x08 - Divide input frequency by 8
0x0A - Divide input frequency by 10
0x0C - Divide input frequency by 12
0x0E - Divide input frequency by 14
0x10 - Divide input frequency by 16
0x14 - Divide input frequency by 20
0x16 - Divide input frequency by 22
0x18 - Divide input frequency by 24
0x1A - Divide input frequency by 26
0x1C - Divide input frequency by 28
0x1E - Divide input frequency by 30

All other combinations are not valid

Table 79. 0x8010D00C - MILREF - Select reference for 1553B clock

31 24	23 16	15 10	9 8	7 0
RESERVED	DUTY	RESERVED	SEL	DIV
0x0	0x0	0x0	0	0
r	rw	r	rw	rw

31: 24 Reserved

23: 16 DUTY cycle for generated clock frequency - The duty cycle bitfield specifies how many of the total clock cycles specified in the DIV bitfield the generated clock shall be high. IF this register is set to 0x0 the duty cycle will be set to clock cycles defined in DIV and the clock period to 2xDIV.

15: 10 Reserved

Table 79. 0x8010D00C - MILREF - Select reference for 1553B clock

9: 8 1553B Reference Clock (SEL) - Select 1553B reference clock and source

0x0 - Clock source from external signal SYS_CLK. See Note 1

0x1 - Clock source from external signal GPIO10

0x2 - Clock source from external signal GPIO47 See Note 1

0x3 - Clock source from external signal GPIO61 See Note 1

Note 1: When MILREF.DIV is not equal to Zero then Clock generated from PLL is used for 1553B Reference clock

7: 0 1553B reference Clock Divisor (DIV) - Set the divisor for input reference clock. Zero (default) bypass the divisor.

When bitfield DUTY period is set to 0x0. The input clock frequency will be divided by 2xDIV clock cycles with the duty cycle set to 50%
When bitfield DUTY period is larger then 0x1.The DIV bitfield will divide the input frequency by DIV clock cycles and with the duty cycle defined in the DUTY bitfield

Table 80. 0x8010D010 - SCLKREF - Select reference for FPGA Scrubber clock

31 24	23 16	15 10	9 8	7 0
RESERVED	DUTY	RESERVED	R	DIV
0x0	0x0	0x0	0x0	0
r	rw	r	r	rw

31: 24 Reserved

23: 16 DUTY cycle for generated clock frequency - The duty cycle bitfield specifies how many of the total clock cycles specified in the DIV bitfield the generated clock shall be high. IF this register is set to 0x0 the duty cycle will be set to clock cycles defined in DIV and the clock period to 2xDIV.

15: 10 Reserved

Table 80. 0x8010D010 - SCLKREF - Select reference for FPGA Scrubber clock

9:	8	RESERVED
7:	0	FPGA Scrubber Reference Clock Divisor (DIV) - Set the divisor for input reference clock. Zero (default) bypass the divisor. (TBD) When bitfield DUTY period is set to 0x0 or 0x1. The input clock (SYS_CLK) frequency will be divided by DIV clock cycles (if DIV > 0) with the duty cycle set to 50%. Valid configurations when DUTY period is set to 0 or 1: 0x00 - Divide input frequency by TBC 0x02 - Divide input frequency by 2 0x04 - Divide input frequency by 4 0x06 - Divide input frequency by 6 0x08 - Divide input frequency by 8 0x0A - Divide input frequency by 10 0x0C - Divide input frequency by 12 0x0E - Divide input frequency by 14 0x10 - Divide input frequency by 16 0x14 - Divide input frequency by 20 0x16 - Divide input frequency by 22 0x18 - Divide input frequency by 24 0x1A - Divide input frequency by 26 0x1C - Divide input frequency by 28 0x1E - Divide input frequency by 30 All other combinations are not valid. When bitfield DUTY period is equal or greater then 0x2.The DIV bitfield will divide the input frequency by DIV clock cycles and with the duty cycle defined in the DUTY bitfield. 0x04 - Divide input frequency by 4 0x06 - Divide input frequency by 6 0x08 - Divide input frequency by 8 0x0A - Divide input frequency by 10 0x0C - Divide input frequency by 12 0x0E - Divide input frequency by 14 0x10 - Divide input frequency by 16 0x14 - Divide input frequency by 20 0x16 - Divide input frequency by 22 0x18 - Divide input frequency by 24 0x1A - Divide input frequency by 26 0x1C - Divide input frequency by 28 0x1E - Divide input frequency by 30 All other combinations is not valid

Table 81. 0x8010D018 - TCTRL - Test Clock Enable

31	6	5	4	3	2	1	0
		PL	FS	MC	SP	SC	EN
0x0		0	0	0	0	0	0
r		rw	rw	rw	rw	rw	rw

31: 6	Reserved
5	PL - Enable of PLL LOCK - when enabled the internal PLL lock signal is available on GPIO 19
4	FS - Enable of FPGA Scrubber clock - when enabled the internal FPGA Scrubber clock is available on GPIO 20
3	MC- Enable of MIL-1553 clock - when enabled the internal MIL-1553 clock is available on GPIO 21
2	SP- Enable of SpaceWire clock- when enabled the internal SpaceWire clock is available on GPIO 22
1	SC - Enable of System clock - when enabled the internal system clock is available on GPIO 23
0	EN- Enable of output test clocks and PLL lock signal

Table 82. 0x8010D020 - RINGEN- Ring oscillator enable register

31	3	2	1	0
RESERVED		EN		
0x0		0		
r		rw		

31: 3 Reserved

2: 0 Enable
0x0 - Disable Ring oscillator
0x1 - Enable Ring oscillator
--
0x7 - Enable Ring oscillator

Table 83. 0x8010D070 - IRQ - PLL interrupt register

31	4	3	2	1	0
RESERVED		BP	SC	XO	CL
0x00000000		0	0	0*	0
r		rw	rw	rw	rw

31: 4 Reserved

3 Bit parity check error (BP) - Configuration bit error detected. When parity error occurs the System clock detector consider it as clock error and sets the System Clock Error (SC) bit to zero.

2 System Clock Error (SC) - System Clock Error detected. This bit is set to zero by the System clock detector when an error occurs.

1 XO Connection (XO) - XO no crystal detected, when no crystal is detected this bit is set to '1' by the detection mechanism.

Note*: To minimize the internal XO current consumption when it is not used, a detector is build-in to shut it down when both XO pins (XO_X1 and XO_X2) are pulled to ground via 10kΩ each, Refer section 9.

0 PLL clock lock (CL) - Shows the current value of the PLL lock output

Note: Writable for testing purposes, generates respective interrupts.

Table 84. 0x8010D074 - STS - PLL interrupt status register

31	4	3	2	1	0
RESERVED		BP	SC	XO	CL
0x00000000		0	0	0	0
r		wc	wc	wc	wc

31: 4 Reserved

3 Bit parity check error (BP) - Configuration bit error detected

2 System Clock Error (SC) - System Clock Error detected.

1 XO Connection (XO) - XO no crystal detected.

0 PLL clock lock (CL) - PLL lock

Table 85. 0x8010D078 - MASK - PLL interrupt mask register

31	4	3	2	1	0
RESERVED		BP	SC	XO	CL
0x00000000		0	0	0	0
r		rw	rw	rw	rw

31: 4 Reserved

Frontgrade Gaisler AB
Kungsgatan | SE-411 19 | Goteborg | Sweden
+46 31 7758650 | frontgrade.com/gaisler

Table 85. 0x8010D078 - MASK - PLL interrupt mask register

3	Bit parity check error (BP) - Interrupt mask for Configuration bit error detected
2	System Clock Error (SC) - Interrupt mask for System Clock Error detected
1	XO Connection (XO) - Interrupt mask for XO no crystal detected
0	PLL clock lock (CL) - Interrupt mask for PLL lock

Table 86. 0x8010D07C - EDGE - PLL interrupt edge register

31	4	3	2	1	0
RESERVED		BP	SC	XO	CL
0x00000000		1	1	1	1
r		rw	rw	rw	rw

31: 4	Reserved
3	Bit parity check error (BP) - Configuration bit error detected
2	System Clock Error (SC) - System Clock Error detected
1	XO Connection (XO) - XO no crystal detected
0	PLL clock lock (CL) - PLL lock

Table 87. 0x8010D080 - PLLSTS - PLL status register

31	3	2	1	0
RESERVED		XP	XO	CL
0x00000000		0*	0*	0
r		r	r	r

31: 2	Reserved
2	XO Power Down Status (XP) - XO power down status. Based on the XO no crystal detected signal, power down (active high) is driven to the XO after TBD delay, status of the power down signal is provided in this register.
1	XO Connection (XO) - XO no crystal detected, when no crystal is detected this bit is set to '1' by the detection mechanism. Note*: To minimize the internal XO current consumption when it is not used, a detector is build-in to shut it down when both XO pins (XO_X1 and XO_X2) are pulled to ground via 10kΩ each, Refer section 9.
0	PLL clock lock (CL) - Shows the current value of the PLL lock output.

10.3.1 PLL Local register interface information

Table 88. 0xE0 - INFO - PLL Local register interface information

31 24	23 16	15 8	7 0
IRQ	CFG	STAT	IRQ
0x04	0x9	0x01	0x04
r	r	r	r

31: 24	IRQ: Number of Interrupts (same as 7:0)
23: 16	CFG: Number of Configuration registers
15:8	STAT: Number of Status registers
7:0	IRQ: Number of Interrupts

Table 89. 0x8010B000 - SYSEL - System clock source configuration register

31	3	2	1	0
RESERVED			S	
0x0			0	
r			rw	

Table 89. 0x8010B000 - SYSEL - System clock source configuration register

31: 3 Reserved

2: 0 Select system input clock source
0x0 - Internal System clock from external SYS_CLK clock

0x6 - Internal System clock from external SYS_CLK clock
0x7 - Internal System clock from internal SpaceWire clock. Refer figure 16.

Note: In order to write to these registers the Ring oscillators must be enabled using PLL Ring oscillator enable register (RINGEN).

Table 90. 0x8010B004 - DETEN - System clock error detector enable register

31 3	2 1 0
RESERVED	EN
0x0	0
r	rw

31: 3 Reserved

2: 0 Enable clock error detector
0x0 - Disable detector
0x1 - Enable detector
--
0x7 - Enable detector

Note: In order to write to these registers the Ring oscillators must be enabled using PLL Ring oscillator enable register (RINGEN).

11 Voltage and Current References

11.1 Overview

The internal voltage and current reference block provides accurate reference voltage and currents in the system.

11.2 Operation

11.2.1 System overview

The internal voltage and current references consist of a bandgap reference providing a high-impedance unbuffered voltage of nominal 1 V and a bias block generating accurate bias currents. Also an internal temperature sensor is available which is read by ADC0, see Figure 10 and 19.

11.2.2 Detailed description

The reference blocks, internal voltage reference and current reference generator, are supplied by VDDA_REF and VSSA_REF. It is essential that there is good PCB decoupling on this supply, especially at high frequencies, since the on-chip disturbance suppression commonly is poor at high frequencies, which would result in high-frequency disturbance transferred directly onto the references used by analog blocks such as ADC and DACs.

Another decoupling capacitor, which is the most critical (sensitive) one for the whole Microcontroller, is on the internal voltage reference output pin, VREF. This decoupling capacitance should be 4.7 nF located very close to the VREF pin, and grounded (very close) to the VSSA_REF pin. There should be no other components on PCB connected to the VREF pin, and its PCB layout connection should not extend beyond the decoupling capacitor, to avoid disturbance on this pin. Preferably, a VSSA_REF local ground plane and guard ring around this pin should be implemented in the PCB layout.

The reference buffer providing VREFBUF is a buffer amplifier with gain 1.9 of VREF. The maximum load current on VREFBUF, with full voltage performance maintained, is 20 mA. It can be used, for example, to perform accurate bridge measurements with the ADC, such as thermistor measurements, or wherever a reference voltage (referred to VSSA_REF) is needed in application circuits on PCB. It is, however, critical that no fast current load steps are present on the VREFBUF output, since that can cause erroneous voltage transients.

The reference resistor, RREF, sets all the reference currents for internal bias currents and the fullscale current for the four DACs. Therefore, it is critical that RREF always is within 4.9-5.3 kΩ over worst-case conditions. The DAC fullscale current is proportional to the current through RREF, where 5.11 kΩ gives a nominal fullscale current of 4.0 mA.

For further details refer datasheet section 2.10 [DS].

11.2.3 VREFBUF control

During and after reset, the VREFBUF buffer amplifier is powered down and the VREFBUF pin is in a high impedance state. To power up the VREFBUF amplifier requires software to write 1 to the VREFBUFCFG register, see sections 8.2.7 and 8.3.

12 ADC, Preamplifier and Analog MUX

12.1 Overview

The GR716B contains 4 Analog-to-Digital Converters (ADCs). Each ADC includes an analog MUX connecting it to 8 GPIO pins and one on-chip sensor. In total there are 16 GPIO pins that can be used as ADC inputs. The ADC can sample GPIO pins in single-ended mode (0-2.5 V nominal measurement range) or as differential pairs (±2V nominal measurement range) at a configurable resolution of 11-16 bits with a maximum sample rate of between 500 and 80 kS/s depending on resolution. See datasheet section 2.8 [DS] for performance specifications. The inputs are measured relative to on-chip reference voltages derived from VREF (see chapter 11). Each ADC additionally has a preamplifer with configurable gain (bypass/x1/x2/x4) which may be used for differential inputs. Section 2.3.5 shows a block diagram of analog connections, including comparators that share the same GPIO inputs.

Each ADC has a dedicated digital control unit with features such as synchronization of ADC start to triggers, autonomous sequencing, level detection, and oversampling to off-load the processors. Possible trigger sources include general purpose timers, APWM timers and GPIO inputs. The control units are interfaced via registers on an APB bus and are mapped to the AMBA address range 0x80400000 to 0x80403FFF. The registers are described in section 12.4. The APB bus is simultaneously accessible from the main CPU, DMA controller, and both real-time accelerators (RTA0 and RTA1) without timing interference. Figure 19 shows an architectural block diagram of the ADCs and supporting peripherals. The system can be configured to protect and restrict access to individual ADC units in the **MEMPROT0** and **APBPROT** units. See section 12.3.19 and chapter 47 and for more information.

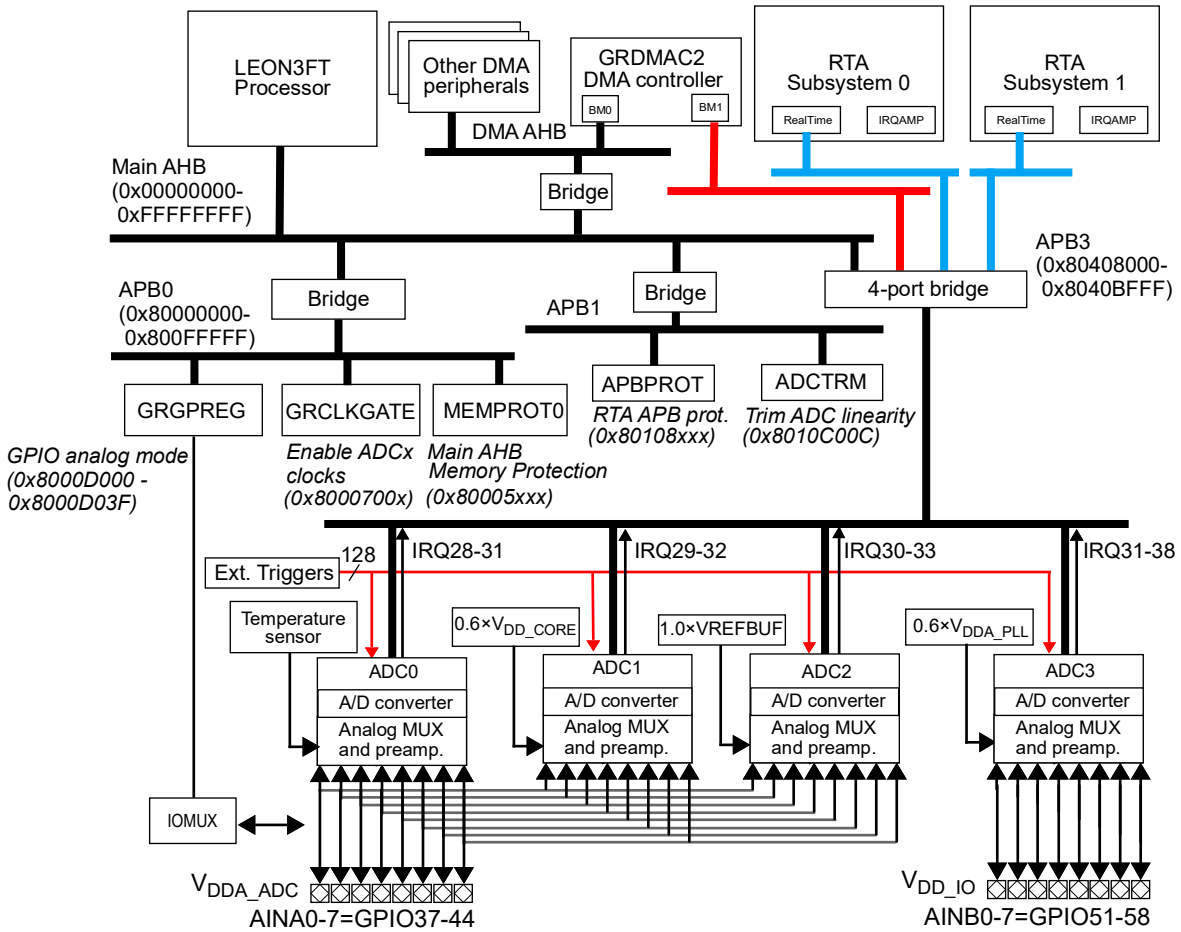


Figure 19. Partial bus diagram showing how the GR716B ADCs and related peripherals are connected to the system bus.

The primary clock gating unit **GRCLKGATE** described in chapter 27 is used to enable/disable individual ADC units. The unit **GRCLKGATE** can also be used to perform reset of individual ADC units. Software must enable clock and release reset described in chapter 27 before ADC configuration and sampling can start. When an ADC is held in reset by the clock gating units, the analog parts of the ADC will be powered down.

The GPIO pins with analog input capability also have digital input and output functionality. When non-digital signals are present on the pins, the digital input and output cells must be disabled by placing the GPIO in analog mode (0x8) in the system IO configuration register (**GRGPREG**) in the address range from 0x8000D000 to 0x8000D03F. See sections 12.3.1, 2.8.3, and 7.1 for more information. For GPIO41-44 and GPIO51-58, the LEB and TEB modes (0x9 or 0xA) are also compatible with the presence of analog signals.

For the linearity of ADC0-2 to reach electrical specifications, the ADCTRM register must be written before starting sampling. ADC3 is not affected by the ADCTRM register. See section 8.2.5.

ADC0, ADC1 and ADC2 are supplied by VDDA_ADC and VSSA_ADC, while ADC3 is supplied by VDD_IO and GND (CQFP package option) or GND_ADC3 (PBGA package option). When an ADC is to be used, its supply needs to be well decoupled at high frequencies (>1 MHz) to not degrade ADC performance. VSSA_ADC and GND (CQFP) or VSSA_ADC and GND_ADC3 (PBGA) must be locally connected to same PCB ground as VSSA_REF to guarantee ADC full functionality and performance.

12.2 Minimal steps to record a sample

1. Place all GPIOs on which non-digital signal levels will be present in analog mode using the system IO configuration registers (section 7.1). The ADC will operate correctly even if this setting is incorrect, but the digital circuitry may be stressed. See section 12.3.1 for more information.

2. Configure the ADC0-2 trim register (ADCTRM). See section 8.2.5.

3. Enable the ADC unit in the clock gating unit (chapter 27). When not enabled in the clock gating unit, the analog components of the ADC will be powered down and the registers are not writable.

4. Write to the idle configuration register (ADC.ICFG). See sections 12.3.17 and 12.4.4. NOTE: The default value of ADC.ICFG after reset will connect the on-chip sensor to ground, which prevents that sensor from being sampled correctly unless ADC.ICFG is first correctly configured from software.

5. Power up the analog parts of the ADC by setting PE and AE fields to 1 in the ADC configuration register (ADC.CFG). Other parameters, such as the ADC clock frequency can be set at the same time. The ADC clock frequency must be within the appropriate range for conversion results to be accurate. See sections 12.3.8 and 12.4.1 for details. NOTE: There are several fields in the ADC.CFG register whose default values at reset are different from the recommended and required values for correct operation. All fields of this register should be written by software.

6. In one or more sampling buffers, select input channel, differential/single-ended sampling, conversion mode (resolution), whether to use or bypass the preamplifier, the number of oversamples to accumulate, if interrupts should be generated and which trigger source should be used to start the sampling. See section 12.4.5 for register descriptions.

7. Start sampling by setting the AS field of the ADC.CFG register to 1. Depending on sequencing mode (see section 12.3.11) this either starts sampling immediately or waits until a trigger condition occurs.

8. Wait until the sampling operation has completed as indicated by the appropriate ADC.SB.STS register or in the shared ADC.STS register. Then read the result of the operation from ADC.SB.STS of the correct sampling buffer, or from ADC.STS.ACC, and clear any status flags of interest. When the sampling rate is not too high, waiting for an interrupt and handling the readout in an interrupt handler is possible. But at high sampling rates, interrupt latency becomes prohibitive and a real-time schedule synchronized to APWM timers may be more appropriate. See section 12.3.8.

9. For sequencing modes with ADC.CFG.SC=1 (sequence continuously), the ADC will automatically start new sampling operations (see section 12.3.11) and one may return to step 7. In the other sequencing modes, the control unit automatically clears ADC.CFG.AS and one may return to step 6 to starts sampling again with the same parameters, or an earlier step to change one or more parameters.

12.3 Detailed description

The analog parts of the ADC unit consists of an analog input MUX connected to GPIO pins, a preamplifier, an on-chip sensor input, a cross-over switch and an analog to digital converter with sample and hold circuit. The reference voltages for all conversions derives from the on-chip bandgap reference (VREF, see chapter 11). A schematic diagram is given in Figure 20.

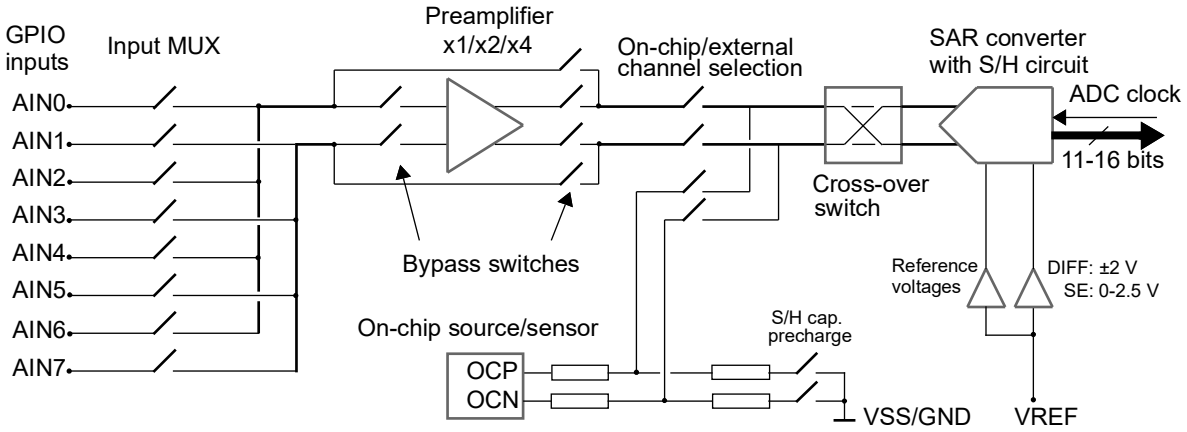


Figure 20. Functional block diagram of the analog stages of one ADC unit. All reference voltages are generated on-chip and are proportional to VREF. The on-chip sensor input can optionally be connected to internal ground when the ADC is idle to discharge the S/H circuit to 0 V in between every sample. The switches are controlled by a digital state machine.

The input MUX connects 8 GPIO pins to the ADC. When making a single-ended conversion, only one input pin is connected to the S/H circuit by the MUX during the ADC **Track** state while the others remain disconnected. For differential conversions, the pins are grouped in differential pairs AIN0-AIN1, AIN2-AIN3, AIN4-AIN5, AIN6-AIN7 and both pins of a pair are connected to the S/H circuit during the **Track** state. See section 12.3.1 for more details about analog GPIO inputs, section 12.3.3 for simplified input circuits of selected channels, and section 12.3.15 for how the MUX and ADC as a whole disturbs the voltage of the pins it measures.

The inside of the input MUX is connected to a differential preamplifier that can provide gain of x1, x2, or x4, or be bypassed. The preamplifier can only be used in differential mode and must be bypassed to sample single-ended inputs. When not bypassed, the preamplifier has an input impedance of approximately 30-80 kΩ depending on gain. See section 12.3.2 for more details on the preamplifier and 12.3.3 for a simplified input circuit.

Besides the GPIO inputs, each ADC has an on-chip sensor input. In the GR716B, the on-chip sensor inputs of ADC0, ADC1, ADC2, and ADC3 are connected to a temperature sensor, V_{DD_CORE}, $V_{REF-BUF}$ and V_{DDA_PLL} respectively. The preamplifier must be bypassed when sampling the on-chip sensor. The internal channels can optionally be used to precharge or discharge the capacitors of the S/H circuit in between conversions. A crossover switch allows the precharge polarity to be controlled. See section 12.3.4 for a detailed description of the on-chip sensors and section 12.3.17 for S/H capacitor precharge.

The main analog component of the ADC is the actual analog to digital converter with sample hold circuit. It can operate in single-ended mode with fullscale range 0-2.5 V or in differential mode with fullscale range ±2 V, with some restrictions on the common mode voltage. When not bypassed, the differential preamplifer can extend the common mode voltage range. The converter can be operated in four conversion modes (0-3) that allow different tradeoffs between sample rate and resolution. Conversion mode 0 provides 11-bit samples at up to 500 kS/s and mode 3 provides 16-bit samples at up to

80 kS/s. Conversion modes 2 and 3 additionally provide a SET detection status flag. See datasheet section 2.8 [DS] for electrical specifications, section 12.3.5 for details about the conversion modes, section 12.3.8 for a discussion of sampling rate and section 12.3.14 for how input filters affect the track time.

All ADC input pins are also analog comparator (ACOMP) inputs. If an ADC samples a pin that is simultaneously used as comparator input, the ADC can emit a voltage disturbance large enough to cause false switching of the comparator. See section 12.3.15 for the disturbance mechanisms and sections 12.3.16 and 12.3.17 for mitigation methods. Section 12.3.13 details the states of the ADC during conversions, including the state of the inputs and precharge state of the S/H capacitors.

To interface the analog circuitry, each ADC unit includes a digital control unit. The control unit is operated via a memory mapped register interface on an APB bus and can generate interrupts on the same bus. It also has a trigger input to allow synchronization to timers or external events. The control unit is composed of 4 (ADC0-2) or 8 (ADC3) sampling buffers each with its own registers, some control registers shared between all sampling buffers, an accumulation register for oversampling, a clock divider to generate the ADC clock for the A/D conversion circuit, and a state machine that performs low-level control of the analog circuit and arbitrates access to the conversion circuit between the sampling buffers. A block diagram of a control unit is given in Figure 21. The registers of the control unit are described in detail in 12.4.

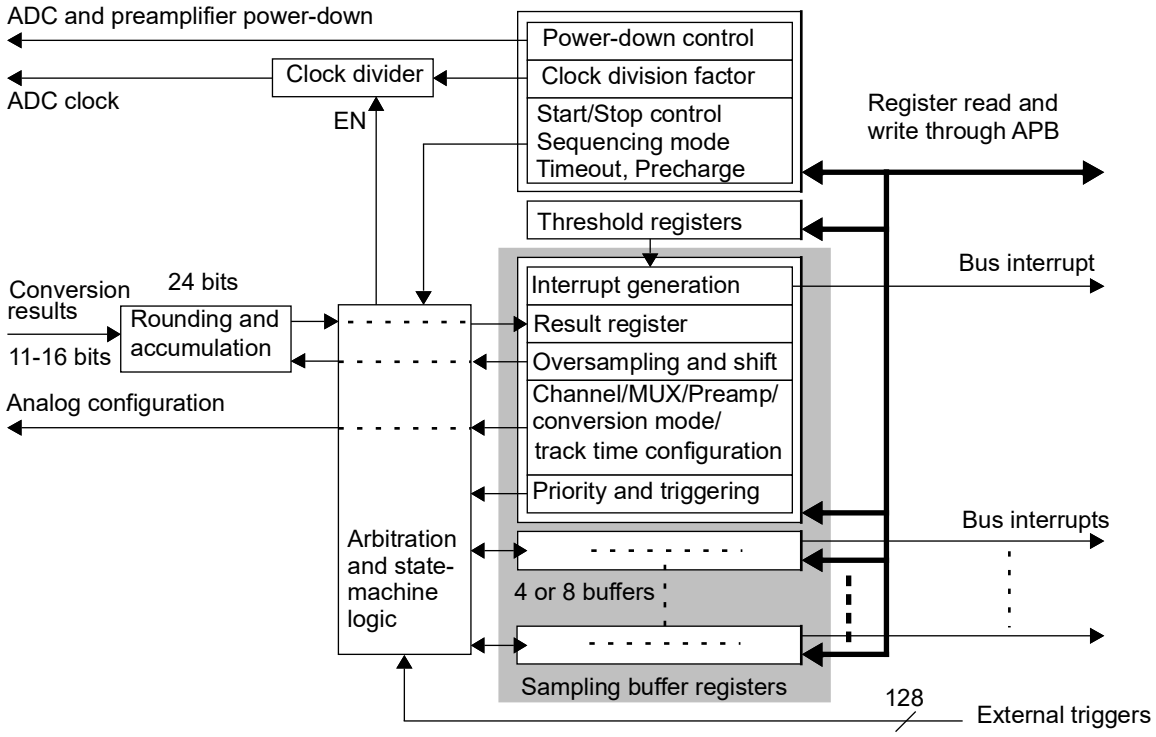


Figure 21. Block diagram of an ADC control unit. The interface to analog circuits is on the left. The system bus interface is to the right.

The control unit implements four sequencing modes that give flexibility in how start of sampling is managed. Sampling can be done immediately in response to a register write, or delayed until a synchronization trigger event occurs. Independently, sampling can be performed as a one-shot operations (where the ADC does not sample again until re-armed by the processor), or with continuous operation (automatic re-arming). The sequencing modes are described in section 12.3.11, and trigger sources in section 12.3.12.

The control unit has multiple sampling buffers to allow more than one sampling operation to be configured in advance per ADC. Each sampling buffer encodes the information needed to perform one sampling operation (channel selection, preamplifier use, track time, resolution/conversion mode, etc) and selects the trigger condition at which the sampling operation will start. The result of a sampling operation is recorded in the status register of the sampling buffer that started it. When multiple sam-

pling buffers are used, access to the A/D converter is arbitrated, with two programmable priority levels. See section 12.3.10, for more details. The data output format is described in section 12.3.6.

The control unit implements configurable **oversampling and shift** as a means to extend the number of effective bits by performing multiple conversions in sequence and averaging the results. Any number of samples from 1 to 256 can be accumulated into a 24-bit accumulation register. The ADC can right shift the accumulated result by 0 to 15 steps, with optional rounding of the result. See section 12.3.7. A note on terminology: Here *sampling operation* is used to denote a complete oversampling sequence consisting of 1-256 individual *ADC conversions*. In case oversampling is not used, then one *sampling operation* consists of a single *ADC conversion*.

Triggers are used for starting conversions in the triggered sequencing modes. When a trigger event occurs for a sampling buffer, a sampling operation will be started with a deterministic number of system clock cycles of delay. The trigger sources include digital GPIO inputs, timer units, APWM ticks and more, with a total of 128 sources. The trigger type is configurable to edge (rising, falling or both) or level (high or low). See section 12.3.12 for a full list of trigger sources and section 12.3.11 for sequencing modes. The trigger bus is shared between all ADC units and also for all DAC units (chapter 15).

The ADC clock is internally generated within the control unit from the system clock frequency using a scaler. The available division factors are any integer from 2 to 256, see section 12.3.8. Refer to datasheet section 2.8 [DS] for the allowed range of ADC clock frequencies. The ADC clock divisor is shared between all sampling buffers within one ADC unit, but can differ between ADC units. The **track time** is however programmable per sampling buffer in units of system clock cycles, allowing it to be optimized for different inputs and analog signal sources. See sections 12.3.13 and 12.3.14 for details.

The sampling buffers have the capability of generating system interrupts. There are three available interrupt conditions: completion of sampling operation, high threshold detection, and low threshold detection. Interrupts are masked/unmasked individually per sampling buffer. See section 12.3.9 for details.

12.3.1 Analog input from GPIO

The ADC analog MUX inputs of the GR716B are connected to the same physical pins as digital GPIO functions. Prior to applying an analog voltage on a GPIO pin, the pin must be placed in analog mode using the I/O switch matrix. Long-term exposure of a GPIO pin not in analog mode to non-digital voltage level will stress the digital input circuit, so software should perform this action early during startup. In analog mode, the digital output driver of the GPIO in a high-impedance state and the digital input disabled, preventing internal current consumption and stress from non-digital signal levels. Analog-capable GPIO pins can be placed in analog mode by setting their I/O switch matrix configurations to 0x8 (see table Table 7 and section 7.1). This applies also to GPIO45-48 when used as DAC outputs, and to any GPIO used as ACOMP input. For ACOMP inputs used with leading/trailing edge blanking function (LEB/TEB, configurations 0x9 and 0xA) in the APWM_AB or APWM_A blocks, selecting the LEB or TEB function in the I/O switch matrix will also place the pin in analog mode except during the blanking period. See sections 53.2 and 53.3 for details. The LEB/TEB modes are therefore also compatible with analog signal levels.

ADC0, ADC1, and ADC2 all share the same 8 input pins GPIO37-44 (AINA0-AINA7). The digital I/O supply of these pins is VDDA_ADC and VSSA_ADC, which also supplies these three ADC units. ADC3 is instead connected to GPIO51-58 (AINB0-7) which are not shared with any other ADC. ADC3 and GPIO51-58 are all supplied by VDD_IO and GND (CQFP package option) or GND_ADC3 (PBGA package option). This means that ADC3 is in principle more susceptible to disturbances due to supply and ground current variations from unrelated digital I/Os switching during conversion. Similarly, ADC0-2 can in principle be indirectly disturbed by activity on any of the GPIO37-44 configured as digital I/O.

When the ADC is not in the **Track** state (see section 12.3.13 for a description of ADC states), the GPIO inputs to the analog MUX are by default disconnected, which ensures that two ADC inputs will not disturb one another. And in the **Track** state only a single GPIO pin or differential pin pair is connected by the MUX. Hence two ADC units can sample different pins simultaneously without any direct disturbance of one measurement on the other. But note that if two different ADC units sample the same pin simultaneously they will disturb one another if the track-times overlap. A use-case for this type of interleaved sampling of one pin by multiple ADCs can be to increase sample rate beyond the capability of one ADC, see section 12.3.8. For the purpose of precharging the S/H circuit in preparation for an upcoming measurement, it is possible to configure a GPIO pin to be selected also outside the **Track** state. See section 12.3.17 for details.

All ADC input pins (GPIO37-44 and GPIO51-58) are also analog comparator (ACOMP) inputs. The pins can be used as ADC inputs and ACOMP inputs simultaneously. However, during an ADC measurement the input pin(s) will be electrically disturbed and can have voltage transients large enough to falsely trigger the ACOMP unless mitigated on PCB and by software configuration. See sections 12.3.3, 12.3.15, 12.3.16, and 12.3.17 for details.

12.3.2 Preamplifier

The ADC includes a fully differential preamplifier for preconditioning of the input signal. The preamplifier can only be used for differential input and must be bypassed for single ended sampling. For differential inputs, it can be bypassed or enabled in one of three gain settings (1, 2, or 4). The gain of the preamplifier allows increased resolution of small differential signals, at the cost of reducing the nominal full-scale range from ±2.0 V (bypassed and gain 1) to ±1.0 V (gain 2) or ±0.5 V (gain 4). Enabling the preamplifier also increases the common mode range compared to when the preamplifier is bypassed. The allowable common mode range depends on the gain setting, see datasheet section 2.8 [DS] for electrical parameters. Another usage of the preamplifier can be to reduce the track time, see section 12.3.14.

When the preamplifier is bypassed, the input pin(s) become directly connected to the internal S/H capacitor(s) when the ADC is in the **Track** state (section 12.3.13). When the preamplifier is enabled, the input pins instead see the preamplifier input impedance during **Track**. See section 12.3.3 for a circuit diagram. The differential input impedance of the preamplifier is approximately 80/50/30 kΩ for gain x1/x2/x4, and it additionally has a common mode impedance of 80 kΩ (independent of gain) to a ~1.5 V internally generated voltage. For source impedances of 1 kΩ and larger, the voltage division over the input impedance results in a gain error of more than 1%. And an imbalance in source impedance between the positive and negative inputs will induce a differential voltage due to the common mode current to/from the internal 1.5 V. I.e.this would introduce a significant dependence of the differential preamplifier output voltage on the common mode source voltage.

For detailed electrical characteristics of the preamplifer, refer to datasheet section 2.8.

12.3.3 Simplified input circuit

Simplified input circuit diagrams for an ADC input channel are given in Figure 22. When the preamplifier is bypassed, the inputs are directly connected to the sample and hold capacitors ($C_{S/H}$). The switches are open by default. The switches for the selected channel are closed during the **Track** state and may optionally be closed at other times as a means to precharge the sample and hold capacitors, see sections 12.3.13 and 12.3.15. The diagrams do not include leakage currents and only cover the ADC input itself. The AIN/AINP/AINN ADC inputs are however connected to GPIO pins with multiple internal functions. For a diagram including all functions see datasheet section 2.4 [DS].

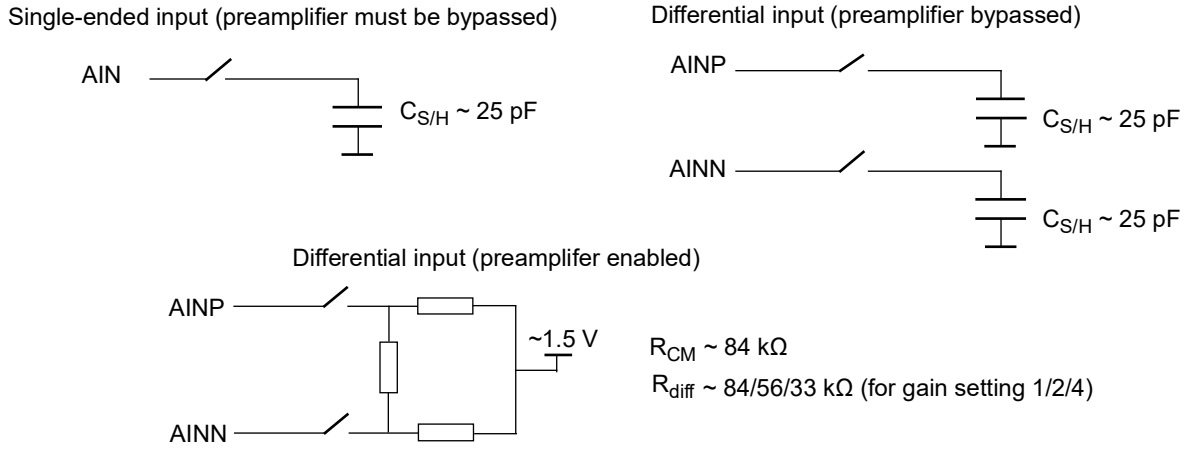


Figure 22. Simplified circuits of an ADC channel, with preamplifier bypassed or enabled.

In the electrical specification, the following definitions are used of preamplifier common mode and differential input resistances R_{CM} and R_{diff}, common mode voltage V_{CM}, differential voltage V_{diff}, common mode current I_{CM} and differential current I_{diff}:

$$V_{CM} = (V_{INP} + V_{INN})/2$$
$$V_{diff} = V_{INP} - V_{INN}$$
$$I_{CM} = (I_{INP} + I_{INN})/2 = (V_{CM} - V_{AGND})/R_{CM}$$
$$I_{diff} = (I_{INP} - I_{INN})/2 = V_{diff}/R_{diff}$$

Here I_{INP} and I_{INN} are the currents *into* the AINP and AINN ADC inputs and V_{INP} and V_{INN} are the corresponding voltages. V_{AGND}~1.5 V is an internal virtual ground generated by the ADC. Notice that the definition of I_{diff} is slightly different from V_{diff} in the sense that $I_{INP}=I_{diff}+I_{CM}$ and $I_{INN}=-I_{diff}+I_{CM}$, while $V_{INP}=V_{diff}/2+V_{CM}$ and $V_{INN}=-V_{diff}/2+V_{CM}$.

12.3.4 On-chip sources/sensors

As shown in Figure 19 and Figure 20, besides the 8 GPIO input sources, each ADC also has an on-chip source/sensor input. Each of the four ADC units is connected to a different on-chip source/sensor. The on-chip source/sensor input of ADC0 connects to an on-chip temperature sensor (chapter 14), for ADC1 it connects to V_{DD_CORE} with a gain factor of 0.6, for ADC2 it connects directly to V_{REFBUF} (gain 1.0), and for ADC3 it connects to V_{DDA_PLL} with a gain of 0.6.

For V_{DD_CORE} and V_{DDA_PLL}, the gain factors are smaller than 1 to ensure that abnormally high supply voltages will not saturate the ADC. For V_{REFBUF} this is not needed because the full-scale range of the ADC and V_{REFBUF} both derive from the same internally generated reference voltage (VREF).

To measure the on-chip source/sensor, the ADC must be in differential mode and bypass the preamplifier. The required minimum track time for the internal channels is generally different than for external channels and is given in Table 91. Additionally, software must set the ADC.ICFG.OCD register bit to 1 to prevent the on-chip source/sensor from being connected to internal ground. Note that this register bit defaults to 0 after reset.

An additional use for the on-chip sources/sensors is as precharge source for the S/H capacitor. This source can be used both directly as the source voltage, and indirectly to discharge the S/H capacitor to internal ground. See section 12.3.15 for details

Table 91. Track settling time (t_{track}) for measurement of internal ADC channels, only available in by-pass differential mode. No controlled precharge of the internal S/H capacitors is assumed. For a functional description of conversion modes 0 to 3 see section 12.3.7. For electrical specification of the conversion modes see datasheet section 2.8 [DS].

ADC unit	On-chip source/sensor	Gain factor	t_{track} [μs_{min}]	
			Conversion mode 0	Conversion modes 1 - 3
ADC0	Temperature sensor)	See chapter 14	6	8
ADC1	V_{DD_CORE} supply	0.6 V/V	0.8	1.0
ADC2	V_{REFBUF} output	1.0 V/V	1.6	2.0
ADC3	V_{DDA_PLL} supply	0.6 V/V (*)	0.8	1.0
(*) The internal channel of ADC3 has an offset of order 100 mV with temperature dependence of around -0.8 mV/°C.				

12.3.5 Conversion modes and resolution

The ADC has four conversion modes (0, 1, 2, and 3) that provide different trade-offs between sample rate, resolution, and SET rejection capability. Conversion mode 0 is the fastest and conversion mode 3 has the highest ENOB. The digital output resolution in conversion modes 0, 1, 2, and 3 are 11, 14, 15, and 16 bits, respectively. The digital output resolution is the sense of the formulas of in section 12.3.6, but the effective analog resolution (ENOB) is smaller. The increased resolution comes at the cost of requiring more ADC clock cycles per conversion and requiring a longer track time. A summary is given in Table 92. See Table 91 and Table 98 for required track time depending on conversion mode and source impedance.

Conversion modes 2 and 3 offer SET warning flags. This allows some classes of SET occurring in the ADC during conversion to be detected and flagged by the ADC.

Note that the *continuous triggered conversion* sequencing mode (see section 12.3.11) can only be used with conversion mode 0 and is not compatible with conversion modes 1, 2, or 3.

Table 92. Summary of GR716B ADC conversion mode properties. The digital resolution is in the sense of section 12.3.6. In this table, the “ENOB” is a theoretical upper bound of the number of statistically significant bits of resolution in the absence of non-linearity in the converter. The integral and differential non-linearity are measured in production, refer to datasheet section 2.8 [DS]. The maximum sample rate is reported under the same single-ended inputs conditions as reported in Table 93.

Conversion mode	SET warning	Digital resolution	ENOB	Non-linearity (INL/DNL)		ADC clocks per sample	Maximum sample rate
				integral	differential		
0	No	11 bits	<11 bits	TBC LSb	TBC 1 LSb	13	526 kS/s
1	No	14 bits	<14 bits	TBC LSb	TBC 2 LSb	23	322 kS/s
2	Yes	15 bits	<14.5 bits	TBC LSb	TBC 4 LSb	69	126 kS/s
3	Yes	16 bits	<15 bits	TBC LSb	TBC 8 LSb	115	80 kS/s

The integral non-linearity in the GR716B ADCs mainly comes from a discontinuity at the midpoint. It occurs at output code $2^{10}/2^{13}/2^{14}/2^{15}$ in conversion mode 0/1/2/3 respectively, which corresponds to an input voltage of ~2.5 V/2 = 1.25 V for single-ended inputs and 0 V for differential inputs. The discontinuity is a sharp step in conversion mode 0. Conversion modes 1-3 increasingly smooth out this discontinuity, which decreases the differential non-linearity, but does not significantly reduce the integral non-linearity on large scales. The net effect can be thought of as an offset that varies with input

voltage: V(c)=k*c+b(c), where b(c) is a step-function for conversion mode 0 and more complicated smoothed step functions for conversion modes 1-3. This means that signals that pass the midpoint will be sampled with larger systematic error than signals that remain on one side of the midpoint. Close to the midpoint, conversion modes 1-3 will have non-constant gain.

At common mode voltage extremes (for differential inputs that bypass the preamplifier) and at low ADC supply voltage, the size of the discontinuity increases. The non-linearity is not strongly time dependent so it can in principle be reduced by calibration. The compensation is a function of the digital output code rather than the voltage itself, so a compensation can remain valid even if there is variation in offset voltage or full-scale range due to change in temperature.

12.3.6 Data formats

All conversion results are encoded as unsigned integers. For differential inputs it represents a signed input voltage with the differential zero at the midpoint of the output range. The following equations give the ideal linearized input voltage V_{IN} (single-ended input) and differential input voltage V_{INP}-V_{INN} (differential input) respectively corresponding to a single conversion result *R*:

$$V_{IN} = FS_{single} \times \frac{R}{2^n} + V_{INOFFS}$$

$$V_{INP} - V_{INN} = \frac{FS_{diff}}{G} \times \frac{R - 2^{n-1}}{2^{n-1}} + V_{INOFFS}$$

The parameters are described below:

- *n* : The number of bits of resolution of the conversion mode. In conversion modes 0, 1, 2, and 3, *n* has value 11, 14, 15, and 16 respectively. See section 12.3.7.
- *R* : The conversion result, an integer in the range 0 to 2^n-1.
- FS : The full-scale range. Nominally FS_{single}=2.5 V and FS_{diff}=2.0 V (±), but see datasheet section 2.8 [DS] for electrical characteristics. The full-scale range is proportional to the on-chip reference voltage VREF (chapter 11). Note that the difference between 2^n and 2^n-1 in the denominator is insignificant compared to typical variation of the full-scale range (FS).
- *G* : The gain of the differential preamplifier. When the preamplifier is bypassed, *G*=1 by definition. When the preamplifier is enabled, the nominal values of *G* are 1.0, 2.0, and 4.0 depending on gain setting. See datasheet section 2.8 [DS] for electrical characteristics.
- V_{INOFFS} : Combined input offset voltage. Ideally 0 mV, but see datasheet section 2.8 [DS] for electrical characteristics. When the preamplifier is enabled, the offset of the preamplifier and that of the ADC both contribute and the resulting offset voltage depends on the gain setting.

Note that the effective full-scale range and input offset voltage may also be modified by source properties and may be influenced by ADC track time and sample rate, since the ADC in general disturbs the channel it samples. See sections 12.3.15.

If read from the ADC.STS.ACC register field, *R* is a 24-bit integer. If read from the ADC.SB.STS.DATA, then *R* is instead a 20-bit integer. In both cases bit *n* and upwards are 0. However, note that bits 31:25 and 31:20 respectively of the ADC.STS and ADC.SB.STS registers contain status bits that must be discarded before any arithmetic is performed on the data.

When oversampling or shifting is enabled (section 12.3.7), the output format of a single conversion remains the same. But the accumulated and shifted value saved in the active sampling buffer status register will have a different range. This is summarised in the two equations below:

$$V_{IN} = FS_{single} \times \frac{A}{(NO+1) \times 2^{n-SH}} + V_{INOFFS}$$

$$V_{INP} - V_{INN} = \frac{FS_{diff}}{G} \times \frac{A - (NO+1) \times 2^{n-1-SH}}{(NO+1) \times 2^{n-1-SH}} + V_{INOFFS}$$

- *A* : An accumulated, shifted and rounded sample as stored in the ADC.SB.STS.DATA register field (the ADC.STS.ACC field is only accumulated, not shifted or rounded). Since *A* is a sum of

NO+1 independent conversions, the relation to R can be thought of as A=(NO+1)*average(R)/2^{SH}. When overflow cannot occur, A has a range of 0 to (NO+1)*(2^n-1). If overflow can occur then A has a range of 0 to 2^{20}-1 (the control unit detects overflow, see section 12.3.7).

- NO : The value of the ADC.SB.CFG3.NO register field. This is the number of oversamples, the total number of accumulated samples per sampling operation is NO+1.
- SH : The value of the ADC.SB.CFG3.SH register field. The number of bits to right shift the accumulated result before storing it in ADC.SB.STS.DATA.

12.3.7 Oversampling

Applications that require a fine resolution at relatively low sample rate may benefit from sampling the same input a large number of times within a short time window and accumulate the results into a single averaged sample of higher resolution and/or less noise than an individual ADC conversion. To increase the effective number of bits (ENOB) by 1 with this method, the signal must be oversampled by a factor of four. Hence increasing ENOB by n bits requires 4^n samples. And the maximum sampling rate decreases by a factor as 4^n well. For example, to increase the resolution by 3 bits requires an oversampling factor of 64. About an order of magnitude (3-4 ENOB) resolution improvement can be expected to be possible with this technique. Beyond that systematic biases such as ADC non-linearity limit the resolution, but noise reduction is still possible.

For oversampling to increase ENOB at all, some minimum amount of noise or dither needs to be present in individual conversions. Otherwise the quantization causes significant bias towards integers in the averaged result. At least 1-2 LSb noise (RMS) is needed for an order of magnitude (3-4 ENOB) resolution improvement to be possible. If the analog source does not have sufficient noise, then oversampling will not be effective in conversion mode 0 (11-bit conversion results) because the noise added by the GR716B ADC itself is less than one 11-bit LSb. In conversion modes 1-3, the ADC noise has a relatively larger effect and may allow some resolution improvement through oversampling. But in general, low-noise sources may need to have noise added on PCB (dithering) for oversampling to increase ENOB.

The ADC control unit implements oversampling in hardware with up to 256=4^4 accumulated samples per conversion allow ENOB increase of up to 4. Further oversampling can be done in software. In the ADC control unit, oversampling is configured separately per sampling buffer using the ADC.SB.CFG3.NO register field, which has a range of 0 to 255. When NO=0, the sampling buffer will not oversample and will only perform a single conversion when triggered. But when NO>0, the sampling buffer will instead track and convert its input channel NO+1 times consecutively. The sum of the NO+1 conversion results is accumulated in the 24-bit register field ADC.STS.ACC. After the last conversion, this 24-bit sum is, after optional right-shifting and rounding, copied to the 20-bit register field ADC.SB.STS.DATA of the sampling buffer.

For each of the NO+1 conversions in a sampling operation with oversampling, the ADC traverses through all states (**Pretrack**, **Precharge**, **Track**, **Hold/Convert**), skipping only the **Idle** state (see section 12.3.13). In particular, it re-tracks the input for every conversion.

Oversampling by a factor 4^n gives $2n$ extra bits in the accumulated result, but at most n of those bits are significant. The ADC control unit allows extraneous bits to be removed by right-shifting and rounding the accumulated result. Right shifting is configured per sampling buffer using the ADC.SB.CFG3.SH register field (range 0 to 15) while rounding is configured per ADC in ADC.CFG.RD. In the following, let ACC denote the 24-bit accumulated value (ADC.STS.ACC) and SH be a shorthand for ADC.SB.CFG3.SH. If rounding is disabled (ADC.CFG.RD=0, or SH=0), then the 20-bit shifted result will be

$$\texttt{shifted\_result} = \texttt{floor}(\texttt{ACC}/2^{\texttt{SH}})$$

If rounding is enabled (ADC.CFG.RD=1, and SH>0), then the 20-bit shifted result is instead

$$\texttt{rounded\_result} = \texttt{floor}((\texttt{ACC}+2^{\texttt{SH}-1})/2^{\texttt{SH}})$$

As can be seen from the formula, ties are rounded up.

Since the sampling buffer data register has fewer bits than the accumulation register, overflow can occur when writing the shifted and rounded result to the sampling buffer. This cannot happen in conversion mode 0, but it can happen in conversion mode 1 if NO>63, in conversion mode 2 if NO>31 and in conversion mode 3 if NO>15. Such overflow can be prevented by using a non-zero shift, SH, and cannot occur if SH=4 or larger. If SH<4 and overflow does occur then this is detected automatically and the 20-bit result will be set to its maximum value (2^{20}-1=0xFFFFF).

With most combinations of settings, overflow cannot occur in the 24-bit accumulation register. However, if rounding is enabled, and NO=255, and conversion mode 3 is used, and SH>8, then there exists a corner case where overflow can occur. This type of overflow is not detected by the hardware and will result in both the ADC.STS.ACC and ADC.SB.STS.DATA fields wrapping around to 0.

Note: Conversion modes 1, 2, and 3 have an overflow flag for each single conversion. This is unrelated to the oversampling discussed above. If overflow or underflow flags are set for any single conversion within an oversampling sequence, the accumulated result will be inaccurate due to saturation of the ADC. An accumulated result will also be inaccurate if an SET warning flag was set for any single conversion.

12.3.8 ADC clock generation and sampling rate

The ADC control units are clocked by the system clock, but the analog portion of the ADC has its own ADC clock of lower frequency. This clock is called “ADC clock” in this section but is referred to as “ADC clock input” in the datasheet [DS].

The ADC clock is generated by the ADC control unit by dividing the system clock by an integer factor. The division factor is individually configurable per ADC control unit in the AC field of the ADC.CFG register (section 12.4.1). The division factor is 2+AC and the allowed range of AC is 0 to 254, hence any integer division factor between 2 and 256 is possible. However, setting AC=255 will stop the ADC clock. The ADC clock only runs during the **Hold/Convert** state (see section 12.3.13) and is automatically stopped in all other states.

The ADC clock must not be too slow or too fast. At low frequency the accuracy degrades because of the cumulative effect of leakage current over time. At high frequency the limiting factor is instead settling time of circuits internal to the ADC. Single-ended inputs can use a somewhat higher ADC clock frequency (~10 MHz, TBC) than differential inputs (~5.0 MHz, TBC) allowing correspondingly higher sample rates. For details, see electrical parameters $f_{adc,clk}$ (max) and t_{conv} (min/max) in datasheet section 2.8 [DS]. The ADC retains functionality outside of these frequency limits, but accuracy and linearity are degraded.

Performing one sampling operation first requires a minimum time t_{track} in the **Track** state to transfer the external voltage to the sample/hold capacitor (see section 12.3.5). Next it takes a fixed number of ADC clock cycles (13/23/69/115 in conversion modes 0/1/2/3, respectively) in the **Hold/Convert** state to convert the sampled voltage into a digital code which is latched in a status register. After the **Hold/Convert** state completes, the **Pretrack** state of the next conversion can start immediately without passing through **Idle**. Readout of the previous conversion result can be done in parallel with the next conversion and hence does not cause any delay. The maximum sampling rate is therefore a function of only the track time and the ADC clock frequency, see Table 93 below. Sustaining these sampling rates is possible in the sequencing modes *Direct continuous conversion* and *Continuous triggered conversion* (section 12.3.11), or with any conversion mode within a sampling operation with oversampling enabled. In other sequencing modes, software or DMA must trigger the start of every conversion with a register write, which adds latency and reduces the maximum sampling rate. There is also a limit on how quickly samples can be read by the system, see the last paragraph in section 12.3.9 for a discussion of this aspect.

Table 93. Maximum sample rate for each conversion mode as limited by minimum t_{track} and maximum $f_{adc,clk}$. Depending on source impedance and PCB implementation, longer track time may be needed (section 12.3.14). Depending on sequencing mode and precharge settings, additional delays of some number of system clock cycles will be present (section 12.3.11).

Conversion mode	Minimum track time	Number of ADC clock cycles	Maximum sampling rate per ADC	
			Single-ended, $f_{adc,clk}$=10 MHz	Differential, $f_{adc,clk}$=5.0 MHz
0 (11-bit)	0.6 µs	13	526 kS/s	312 kS/s
1 (14-bit)	0.8 µs	23	322 kS/s	185 kS/s
2 (15-bit)	1.0 µs	69	126 kS/s	67 kS/s
3 (16-bit)	1.0 µs	115	80 kS/s	41 kS/s

The sample rates in Table 93 are for one ADC. Since the GR716B has three ADCs sharing the same input pins, it is possible to reach higher sampling rates by sequentially sampling the same pin with two or even three ADCs in parallel, as illustrated in Figure 23 below. In such an application, it is important that the **Track** states do not overlap, because charge is ejected at the start of tracking which would disturb the other ADC (section 12.3.15). Note that different ADCs on the same chip may have systematically different offset and gain/full-scale range and software may need to apply calibration factors to compensate for this difference. It is also important that the ADCs retain their relative timing delay over arbitrarily long time and that software correctly keeps track of the time ordering of samples during readout. Note that the sampling buffer status register can only retain the most recent sample. A robust (to both software and radiation-induced errors) method is to use the *Continuous triggered conversion* sequencing mode (section 12.3.11) with triggers selected from an APWM timer tick line (sections 12.3.12 and chapter 53). In case of SEU, the APWM timers have parity check with configurable interrupt generation and shutdown in the alarm matrix, unlike for example the GPTIMERs (chapter 35). See section 12.3.9 for some software aspects in managing a high sample rate.

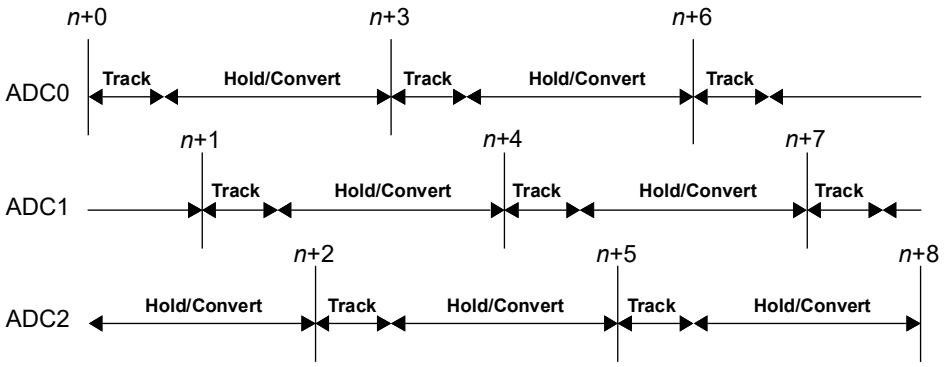


Figure 23. Timing diagram of an interleaved sampling schedule with three ADCs alternately sampling the same pin.

When using the *continuous triggered conversion* sequencing mode, it is possible to configure multiple sampling buffers to sample the same channel, provided they are given different trigger sources. This can allow more samples to be recorded before software/DMA readout which may enable some reduction of latency requirements for readout at high sampling rates.

12.3.9 Interrupt and tick generation and threshold detection

The ADC sampling buffers can generate system interrupts and RTA ticks. Three kinds of events can generate interrupts/ticks: Completion of a sampling operation, high threshold detection, and low threshold detection. The three events can be masked individually per sampling buffer.

Each sampling buffer within an ADC unit is connected to a different system interrupt line. Specifically, sampling buffer *m* of ADC unit *n* drives interrupt line 28+*n*+*m*. Note that this means that there is

overlap between the system interrupt numbers of different ADC units. Additionally, interrupts 0-3 of ADC0-2 are connected to the TickLines2 inputs of the two RTAs. Specifically, the interrupt output of sampling buffer *m* of ADC *n* drives TickLines2(4+4*n*+*m*). But ADC3 is not connected to any RTA tick lines. A complete list is given in Table 94. See also section 52.5.

Table 94. Interrupt line assignment for ADC sampling buffers in GR716B. Note that ADC0-2 only have 4 sampling buffers. And that ADC3 has not interrupt outputs connected to RTA tick lines.

Sampling buffer number	System interrupt line number				RTA TickLines2 connection			
	ADC0	ADC1	ADC2	ADC3	ADC0	ADC1	ADC2	ADC3
0	28	29	30	31	4	8	12	-
1	29	30	31	32	5	9	13	-
2	30	31	32	33	6	10	14	-
3	31	32	33	34	7	11	15	-
4	-	-	-	35	-	-	-	-
5	-	-	-	36	-	-	-	-
6	-	-	-	37	-	-	-	-
7	-	-	-	38	-	-	-	-

High and low threshold interrupts occur if the result of a sampling operation exceeds ADC.AHT or ADC.ALT. There is a single AHT and ALT register per ADC shared between all sampling buffers. Enabling threshold detection can be done independently per buffer, but the thresholds are the same for all sampling buffers within an ADC.

Each interrupt can be enabled or disabled per sampling buffer using the MH, ML, and ME mask bits of the ADC.SB.CFG2 register. When an interrupt has occurred, this is recorded in the IH, IL, or IE flag fields of the ADC.SB.STS register.

Interrupts are only generated when the condition for the interrupt occurs and the corresponding interrupt flag (ADC.SB.STS.IH, ADC.SB.STS.IL, or ADC.SB.STS.IE) is not set. If an interrupt of a given type has already occurred for a given sampling buffer, then the flag must be cleared by writing to the corresponding ADC.SB.STS register before the interrupt can occur again. For the completion-of-sampling-operation interrupt, the sampling buffer can be configured to self-clear the ADC.SB.STS.IE which may simplify readout using the GRDMAC2 DMA controller.

Note that in applications with high sampling rate, readout and management of ADCs through interrupt handlers on the main processor is not always possible. The reason is that high-level interrupt handlers can, depending on the operating system and software runtime, have prohibitively long latency. To give an example, general-purpose interrupt handlers in high-level programming language typically require 100-200 system clock cycles to be reached in the GR716B when executing from on-chip RAM. At 20 MHz system clock, this corresponds to 5-10 µs. But a single ADC sampling at 500 kS/s will deliver one new sample every 2 µs. A simple polling loop, even if written in a high-level language might require 10-20 system clock cycles (1-2 µs at 20 MHz) per iteration and would more easily cope with high sampling rates. At 50-100 MHz there is more margin, but high-level interrupt handlers are still prohibitively slow. Low-level interrupt handlers written at the assembler level may reduce the latency to practical levels comparable with polling loops.

If the main processor has other tasks to attend besides ADC readout, some offloading of ADC readout is possible with the GRDMAC2 DMA engine and this core can be driven by interrupts from the ADC. But overhead from descriptor fetches makes this slower than a dedicated software based polling loop. To manage multiple ADCs at high sample rate, a real-time schedule based on timer ticks from the APWM and readout using one or two RTAs is recommended.

12.3.10 Sampling buffers

To enable scheduling of more than one channel at a time, each ADC control unit contains multiple sub-units termed “sampling buffers”. Only the *triggered* sequencing modes make use of more than one sampling buffer. The *direct* sequencing modes only use sampling buffer 0. Each sampling buffer has its own set of configuration registers for one upcoming sampling operation and a status register to hold the result of one completed sampling operation. The configurable parameters are:

- Which event should trigger the sampling operation specified by the sampling buffer to start (section 12.3.12) and the priority of the conversion. This is only applicable in the *Triggered single conversion* and *Continuous triggered conversion* sequencing modes (section 12.3.11).
- The analog configuration of the ADC for the conversion. This includes whether to perform a single-ended or differential conversion, the preamplifier setting (bypassed, or enabled with gain 1, 2, or 4), which input pin/internal channel to sample, which conversion mode to use (0-3, see section 12.3.5), and the track time of the conversion (see section 12.3.14).
- The oversampling configuration. I.e. the number of ADC conversions to accumulate into one sample during the sampling operation and by how many bits to right-shift the final result. See section 12.3.7.
- Whether interrupts should be generated on sample completion or if the sample result is outside limits set by threshold registers (see section 12.3.9).

The sampling-buffer specific registers have names starting with “ADC.SB” and are described in section 12.4.5. Settings other than the above are the same for all sampling buffers. These shared settings include:

- Power enable for preamplifier and ADC analog blocks (ADC.CFG.AE and ADC.CFG.PE)
- ADC clock frequency (see section 12.3.8)
- Sequencing mode (see section 12.3.11)
- S/H capacitor precharge, and more generally the analog configuration for the ADC when it is *not* in the **Tracking** or **Hold/Convert** states. See ADC.ICFG and sections 12.3.13 and 12.3.17.
- Conversion timeout (see ADC.CFG.TSET in section 12.4.1)
- High and low thresholds for level detection interrupts (ADC.ALT and ADC.AHT)
- Rounding mode to use for shifted oversamples (ADC.CFG.RD)

12.3.11 Sequencing modes

An ADC conversion is started in different ways depending on sequencing mode. There are four sequencing modes, one for each combination of values of the SQ and SC fields of the ADC.CFG register. See Table 95 for details. For modes that make use of trigger conditions, refer to section 12.3.12 for available trigger sources. The four *sequencing modes* are distinct from and should not be confused with the four conversion modes (section 12.3.5).

Table 95. ADC sequencing modes configurable in ADC.CFG. The SQ, SC and AS columns indicate the value of the field of ADC.CFG of the same name.

ADC.CFG configuration bit			
SQ	**SC**	**AS**	**Description**
X	X	0	*Idle* - No new ADC conversion will be started, but an already started conversion will run to completion, unless forced to stop by setting ADC.CFG.ST=1.
0	0	1	*Direct single conversion* - Immediately starts a sampling operation for sampling buffer 0. ADC.CFG.AS is automatically cleared when the conversion completes.
0	1	1	*Direct continuous conversion* - Immediately starts a sampling operation for sampling buffer 0. When the operation completes, ADC.CFG.AS remains set and a new sampling operation is started immediately.
1	0	1	*Triggered single conversion* - A sampling operation starts when a trigger condition occurs for one of the sampling buffers. The sampling operation parameters are taken from the sampling buffer whose trigger occurred. ADC.CFG.AS is cleared after the sampling operation is complete.
1	1	1	*Continuous triggered conversion* - A sampling operation starts when a trigger condition occurs for one of the sampling buffers. The configuration is taken from the sampling buffer whose trigger occurred. ADC.CFG.AS remains set after the sampling operation completes. Note: Only supported with conversion mode 0.

Direct single conversion: The simplest mode which requires minimal software complexity. To perform a sampling operation, software configures the parameters of the conversion in the registers of sampling buffer 0, starts the conversion by writing to the ADC.CFG register, waits for the conversion to complete (as determined by polling the status register or by occurrence of an interrupt), and finally reads out the result from the status register. After the conversion completes, the ADC returns to the *Idle* sequencing mode. Some drawbacks of this sequencing mode are difficulty in synchronizing the start of sampling, reduced sampling rate due to software latency in between sample readout and register write to start the next conversion, and additional latency needed when changing channel from one conversion to the next.

Direct continuous conversion: Preparing and starting sampling is the same as in *single conversion*, but the sample rate is maximized since the latency between readout and start of next conversion is eliminated. Controlling the precise sample rate is possible indirectly through the number of system clock cycles in the **Pretrack**, **Precharge** and/or **Track** states. Synchronizing the start of sampling to an external event or timer is difficult. And in case of SEU in the ADC, the sampling phase may drift relative to timers or PWM periods.

Triggered single conversion: In this mode sampling only starts when a trigger condition occurs. When a triggered sample completes the ADC returns to *Idle*. This allows precise synchronization of samples with external events, such as a specific point within an APWM duty cycle. If the order of events cannot be predicted in advance, or to reduce the amount of reconfiguration needed of the ADC in between samples, multiple sample buffers can be enabled. Sampling rate is limited by the need for software to write to ADC.CFG after every conversion before a new one can start.

Continuous triggered conversion: This mode is the same as *triggered single conversion*, with the sole difference that ADC.CFG.AS remains 1 after the conversion. Hence there is no software latency requirements for the start of the next conversion. A drawback is that this sequencing mode does not function correctly for conversion modes 1-3, but only for conversion mode 0.

In the *Direct single conversion* and *Direct continuous conversion* sequencing modes, only sampling buffer 0 is used and the trigger settings in ADC.SB.CFG1 and priority setting in ADC.SB.CFG2.HP of all buffers are ignored. But in *Triggered single conversion* and *Continuous triggered conversion*, the fields of all 4 or 8 sampling buffers are used and an external trigger (section 12.3.12) is necessary for a conversion to start.

In case the conditions for more than one sampling buffer are fulfilled simultaneously in the *Triggered single conversion* or *Continuous triggered conversion* sequencing modes and all sampling buffers have the same value of ADC.SB.CFG2.HP, then the highest numbered sampling buffer gets priority.

An ongoing sampling operation can be stopped before completion in several different ways. Firstly, software can at any time force an ongoing conversion to stop immediately by writing 1 to the ADC.CFG.ST register bit. This forces the ADC to *Idle* (ADC.ACG.AS=0) mode and to the **Idle** state (section 12.3.13). If enabled (via ADC.CFG.TE and ADC.CFG.TSET), a timeout in the **Hold/Convert** state has the same effect. Disabling power to the ADC via ADC.CFG.AE or resetting the ADC control unit via the clock gating also immediately stop sampling and place the ADC in power-down mode. Setting ADC.CFG.AS to 0 will allow any currently ongoing sampling operation to complete before returning the ADC to *Idle*.

In the *Triggered single conversion* and *Continuous triggered conversion* sequencing modes a sampling buffer can be configured to have high priority (ADC.SB.CFG2.HP=1). If the trigger for a sampling buffer with HP=1 occurs, while a buffer with HP=0 is performing a sampling operation, then the ongoing operation is stopped. And a sampling operation for the high priority buffer is started. Buffers of equal priority never interrupt one another. In case of simultaneous triggers for two sampling buffers of the same priority, the higher numbered buffer will start first. In the *Continuous triggered conversion* sequencing mode, the triggered status (ADC.SB.STS.TRG) for an interrupted sampling buffer remains set. Hence it will restart its sampling operation when the high-priority sampling buffer has completed its sampling operation, without any new external trigger. The interrupted sampling buffer will indicate that it was interrupted by setting its trigger lost flag (ADC.SB.STS.TLO).

12.3.12 External trigger sources

To synchronize ADC conversions to external events or a real-time schedule, 128 trigger input sources are provided. They are listed in table 96. Triggers are an optional feature and are only used in the *Triggered single conversion* and *Continuous triggered conversion* sequencing modes, see table 95. The same set of trigger sources are also available in DACs, see chapter 15.

Trigger sources are individually selectable per sampling buffer. And the sampling buffer can be selected to trigger on rising/falling edges or low/high level of the trigger source. In total up to four (ADC0-2) or 8 (ADC3) different triggers can be enabled simultaneously per ADC. Additional delay relative to the trigger occurrence can be programmed by extending the **Pretrack** state or by enabling the **Precharge** state, see section 12.3.13 for details. Using the same trigger source in two different ADCs is a simple way to make them sample simultaneously or with a relative delay that is a precisely known number of system clock cycles. Using the same trigger source for two sampling buffers within the same ADC also has potential applications. Doing so will result in the higher numbered sampling buffer executing first with lower-numbered buffers proceeding in turn, provided no additional triggers

occur until the sampling operation for the lowest numbered sampling buffer in the set has started.

Table 96. ADC trigger input sources.

Trigger source number (ADC.SB.CFG1)	Description	Documented in
0	Timer unit 0 scaler tick	Section 35
1 to 7	Timer unit 0 counter 1-7 underflow. ADC.SB.CFG1=*n* selects counter *n*.	
8	Timer unit 1 scaler tick	Section 36
9 to 15	Timer unit 1 counter 1-7 underflow. ADC.SB.CFG1=8+*n* selects counter *n*.	
16 to 31	GPIO0-15 via Schmitt trigger and sampled into one flip-flop clocked by the system clock prior to reaching the ADC. These GPIO Schmitt trigger input paths are always active regardless of GPIO MUX configuration. For example, this means that it is possible to use a GPIO configured as an output to generate a trigger. ADC.SB.CFG1=16+*n* selects GPIO*n*.	This table.
32 to 87	APWM TickLines1(0 to 55) ADC.SB.CFG1=32+*n* selects TickLines1(*n*).	Table 701
88 to 117	APWM TickLines0(34 to 63) ADC.SB.CFG1=88+*n* selects TickLines0(34+*n*)).	Table 700
118	RTA1 tick output 0 (local GPTIMER subtimer 1 tick)	Section 52.4
119	RTA1 tick output 1 (local GPTIMER subtimer 2 tick)	
120	RTA0 tick output 0 (local GPTIMER subtimer 1 tick)	
121	RTA0 tick output 1 (local GPTIMER subtimer 2 tick)	
122 to 125	DAC0-3 synchronization output (ADC.SB.CFG1=122+*n* selects DAC*n* synchronization output).	Section 15.4.3
126 to 127	These two trigger inputs have the constant value 0. Can be used as continuous trigger by selecting low level trigger in the ADC.SB.CFG1.SEQC register field.	This table.

12.3.13 ADC states during conversion

At any given time, the ADC is in one of five states: **Idle**, **Pretrack**, **Precharge**, **Track** or **Hold/Convert**. When no sampling operation is in progress, the ADC is in the **Idle** state. When a sampling operation is started, the ADC transitions, in sequence, through **Pretrack**, **Precharge** (optional), **Track** and **Hold/Convert**. At the conclusion of the **Hold/Convert** state, the conversion result is written (or accumulated) to the ADC.STS.ACC register field. After the **Hold/Convert** state, the ADC will either go to the **Idle** state (if the sampling operation is complete) or immediately start a new conversion by going directly to **Pretrack**. The latter happens within an oversampling sequence, and can also happen if using one of the *continuous* sequencing modes (ADC.CFG.SC=1, see section 12.3.11). A summary of these states and their properties is given in Table 97. In particular the table gives the electrical connections of the GPIO pin(s) and internal S/H capacitor which is important when using the precharge function described in section 12.3.17.

During **Track**, the internal S/H circuit is connected to and charged by the selected channel, see section 12.3.14 for details. During **Hold/Convert** the voltage of the internal S/H circuit is compared against the internal reference voltage and converted to a digital value with 11, 14, 15, or 16 bits of resolution depending on the ADC conversion mode, see section 12.3.5 for details. The **Idle** and **Precharge** states may be used for precharging the S/H circuit since **Idle override** can be configured to be in effect in these states. See section 12.3.17 for details. In the **Pretrack** state, the input is deselected and the S/H circuit floats. This is a delay state through which the ADC always passes and stays in for a programmable number of system clock cycles (at minimum one).

In applications that rely on single-cycle accuracy for start or end of **Track**, it is recommended to verify that the settings used are correct by electrical measurement on a breadboard. For GPIO inputs with sufficiently low capacitance, the start of **Track** is detectable with standard oscilloscope probes from the transient caused by the sudden connection to the on-chip S/H capacitor to the GPIO (section 12.3.15), provided the S/H capacitor is precharged to a voltage different to that of the GPIO. The end of the **Track** state occurs exactly ADC.SB.CFG2.TT+1 system clock cycles after the start of **Track**.

Table 97. Summary of ADC states, their duration and the analog connection of GPIO pins and the internal S/H capacitor in each state. Note that the "GPIO connection" row only applies to the GPIO pin selected by the active sampling buffer during **Track**, or by ADC.ICFG when **Idle override** is in effect. Other analog GPIO inputs remain High-Z. Note that in differential mode a differential pin pair is connected. And each GPIO pin of the pair is connected to an independent S/H capacitor if the preamplifier is bypassed.

State	**Idle**	**Pretrack**	**Precharge**[1)]	**Track**	**Hold/Convert**
Next state	**Pretrack**	**Track** or **Precharge**[1)]	**Track**	**Hold/Convert**	**Idle** or **Pretrack**, depending on triggers, sequencing and oversampling settings.
Description	ADC is stopped and waiting for a new conversion.	A trigger or start condition has occurred and conversion is ready to start.	This state can be used for controlled precharge of S/H capacitor if configured in ADC.ICFG.	The analog voltage is sampled into the S/H capacitor in this state.	The voltage on the S/H capacitor is converted into a digital output code by comparison to the internal voltage reference.
Duration	Until trigger or manual ADC start. Enters **Pretrack** state 3 system clock cycles after an edge trigger.	1+PTE*(PT+1) system clock cycles. Note: PTE and PT are fields in the ADC.ICFG register.	PCH+1 system clock cycles. This state can only be reached if PCHE=1. Note: PCH and PCHE are fields in the ADC.ICFG register.	TT+1 system clock cycles. Note: TT means ADC.SB.CFG2.TT of the active sampling buffer.	Fixed number of ADC + system clock cycles depending on conversion mode. [4)] Mode 0: 13+0 cycles Mode 1: 23+0 cycles Mode 2: 69+2 cycles Mode 3: 115+4 cycles
[2)] **Idle override**	In effect if ADC.ICFG.IO=1	Never in effect	In effect if ADC.ICFG.IOC=1	Never in effect	Never in effect
GPIO connection	Without **Idle override**: High-Z	High-Z	Without **Idle override**: High-Z	Connected directly to S/H capacitor or to preamplifier input[3)].	High-Z
	With **Idle override**: Configurable via ADC.ICFG.		With **Idle override**: Configurable via ADC.ICFG.		
S/H capacitor connection	Without **Idle override**: Floating	Floating	Without **Idle override**: Floating	Connected directly to GPIO if the preamplifier is bypassed. Otherwise to preamplifer output.	Internal connection to SAR converter. The charge of the capacitor remains unchanged, except for the slow drift caused by leakage current.
	With **Idle override**: Configured by ADC.ICFG.TRACK.		With **Idle override**: Configured by ADC.ICFG.TRACK.		

[1)] After the **Pretrack** state, the next state is **Precharge** if ADC.ICFG.PCHE=1. Otherwise the next state will be **Track**.

[2)] "**Idle override**" is a condition that is programmable in ADC.ICFG to be in effect in the **Idle** and/or **Precharge** states. One use is for controlled precharge of the S/H capacitor, see section 12.3.17.

[3)] When not bypassed, the preamplifier has nominal differential input impedance 33/56/84 kΩ (depending on amplifier gain) and common mode impedance of ~84 kΩ to ~1.5 V, see section 12.3.3. For single-ended inputs, the preamplifier must be bypassed.

[4)] The duration of the **Hold/Convert** state is given in the from "*x*+*y*", where *x* indicates the number of ADC clock cycles and *y* indicates the number of system clock cycles. One ADC clock cycle is exactly ADC.CFG.AC+2 system clock cycles long and the ADC clock is started on the same system clock cycle that the **Hold/Convert** state is entered. Hence "*x*+*y*" corresponds to precisely (ADC.CFG.AC+2)**x*+*y* system clock cycles.

12.3.14 Input filter and track-time configuration

The internal sample and hold (S/H) circuit in the ADC is at any time either tracking the input voltage, or holding a previously sampled voltage. While tracking, the internal S/H capacitor is connected to the analog signal source to be measured. When the internal preamplifier is used, see Figure 20, the S/H capacitor is connected to the preamplifier output. When the internal preamplifier is bypassed, the S/H capacitor is connected directly to the GPIO pin(s). Bypass mode can be used either in single-ended or differential input mode. In differential mode, the S/H circuit consists of two independent S/H capacitors to ADC ground.

The end time point of track state, when going into hold state, is when the analog voltage in the S/H capacitor goes from following the input source to becoming frozen. Therefore, this is the time point to be regarded the sample time point of the analog input signal. It is critical that the analog signal source gets enough time with the ADC in the **Track** state for the S/H capacitor voltage to settle accurately before entering **Hold/Convert**. This settling should go on at least until the residual settling voltage error is 1 LSb or smaller. Here, 1 LSb refers to the ENOB resolution for the A/D conversion mode (0-3) to be executed.

The track time is individually selectable per sampling buffer, separately from selection of conversion mode 0 to 3. The minimum track time, which always must be fulfilled, is 0.6 µs for conversion mode 0 and 0.8 µs for conversion modes 1 to 3, or up to 1.0 µs to fully benefit from conversion modes 2 and 3 ADC resolution.

When the preamplifier is enabled, the S/H capacitor will always settle within these minimum track times. However, note that the preamplifier has DC input impedance in the 30-80 kΩ range (depending on gain setting) as described in sections 12.3.2 and 12.3.3. This impedance is only connected to the two selected GPIO pins *during* the **Track** state, so some resettling time may be needed if the PCB signal source is sufficiently disturbed. See also section 12.3.15. The preamplifier also has a maximum track time beyond which offset is introduced, see datasheet section 2.8 [DS].

When the preamplifier is bypassed, as must be done for single ended signals and the on-chip sources/sensors, the settling time depends on source impedance and this must be taken into account when calculating the minimum track time. For the on-chip sources/sensors, minimum track times are given in Table 91. For GPIO inputs consider, as an application example, an RC low-pass filter for a single ended input as in Figure 24. Here R_{LP} and C_{LP} form a low-pass filter, R_{prot} is a current limiting resistor and an additional capacitor $C_{LP,ACOMP}$ may be needed when the ADC input is simultaneously used as a comparator input (see section 12.3.16). When C_{LP}>10 nF, the protection resistor R_{prot} of at least 330 Ω protects against transient overcurrent (max 10 mA). When C_{LP}<10 nF a protection resistor is not needed in this position. C_{LP} (and R_{prot} and $C_{LP,ACOMP}$ if present) should be placed close to the package pins.

Table 98 lists minimum required track time for ADC single ended inputs in bypass mode for this filter topology. The track times also apply for differential inputs, if the topology is modified such that C_{LP} is inserted differentially between the GPIO pin pair with one resistor of size $R_{LP}/2$ inserted in each differential line. When C_{LP} is significantly smaller than 100 nF, the capacitive disturbance from switching in the S/H capacitor can become larger than 1 LSb (conversion mode 0) and the full RC filter must resettle during **Track** so the main contributor of t_{track} is the RC time constant. When C_{LP} is sufficiently large, the S/H capacitor will disturb the voltage on the capacitor by less than 1 LSb so the minimum track time is again sufficient. However, for large C_{LP}, the RC filter must still settle in between conversions. If the S/H capacitor is charged to a different voltage in between conversions (for example if it alternates between two channels) then this RC settling will limit the maximum sample rate.

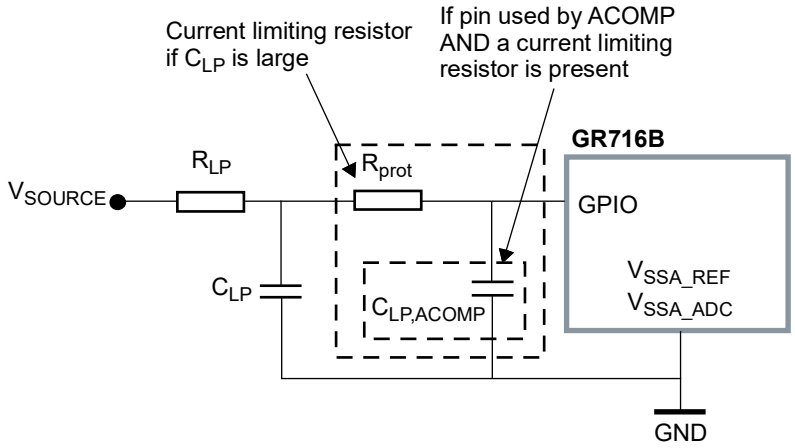


Figure 24. Low-pass RC filter on analog GPIO single ended ADC input pin. If R_{prot} is not present, then C_{LP} can perform the function of $C_{LP,ACOMP}$.

Table 98. Track settling time (t_{track}) that gives 1 LSb accuracy for ADC by-pass mode, including C_{LP} tolerance of ±20% and maximum internal S/H capacitance for the filter topology of Figure 24. No controlled precharge of the S/H capacitor is assumed (see section 12.3.15). For brevity "Conversion mode" is abbreviated to "Mode". For electrical characteristics of conversion modes 0 to 3, see datasheet section 2.8 [DS].

C_{LP} [pF_{nom}]	R_{LP} [Ω_{max}]	R_{prot} [Ω_{max}]	t_{track} [μs_{min}]			Comment
			Mode 0	Mode 1	Modes 2-3	
0	600	0	0.6	0.8	TBD	PCB source is continuously on, or turns on fast (<<t_{track}) at track start. PCB parasitics of up to 25 pF taken into account.
0	1000	0	0.8	1.0	TBD	
0	10000	0	6	8	TBD	
33	500	0	0.6	0.8	TBD	
33	1000	0	0.9	1.2	TBD	No current limiting resistor is needed for C_{LP}<10 nF, hence R_{prot}=0.
33	10000	0	7	9	TBD	
100	400	0	0.6	0.8	TBD	PCB source is continuously on, and capacitively disturbed by track turn on. PCB parasitics of up to 25 pF taken into account.
100	1000	0	1.2	1.6	TBD	
100	10000	0	11	15	TBD	
330	200	0	0.6	0.8	TBD	
330	400	0	1.1	1.5	TBD	No current limiting resistor is needed for C_{LP}<10 nF, hence R_{prot}=0.
330	1000	0	2.5	3.5	TBD	
330	10000	0	25	35	TBD	
1000	80	0	0.6	0.8	TBD	
1000	400	0	2.3	3.4	TBD	
1000	1000	0	6	9	TBD	
1000	10000	0	60	90	TBD	
100 nF_{nom}	N/A [1]	600	0.6	--- [2]	--- [2]	PCB source is continuously on, and C_{LP} voltage disturbed <1LSb by track turn on. For C_{LP}>10 nF it is necessary to have a current limiting resistor R_{prot} of minimum 330 Ω.
1 uF_{nom}	N/A [1]	600	0.6	0.8	TBD	

Note 1: With sufficiently large C_{LP}, the charge ejection by the ADC disturbs the voltage on the capacitor by less than 1 LSb. Hence there is no need to settle the input filter during **Track** and the minimum track time is sufficient for any value of R_{LP}. Instead, charge ejection by the ADC sets a limit on the minimum time between two conversions since the source plus C_{LP} must still settle in between two conversions.

Note 2: Charge ejection in ADC bypass mode causes more than 1 LSb voltage change in 100 nF capacitance for ADC conversion mode 1 to 3, and it takes very long time to resettle the voltage in this large capacitance. Therefore, the order of 100 nF is very unfavorable for these three ADC modes. Instead, at least 1 µF should be used here which gives less than 1 LSb change caused by track turn on in bypass mode.

12.3.15 ADC disturbance on input pins during sampling

When sampling a pin or differential pin pair, the selected pin(s) will generally be disturbed during the **Track** state while the S/H circuit in the ADC is sampling the voltage. This can contribute to measurement error and potentially interfere with other circuitry connected to the pin(s). In particular the GR716B has fast analog comparators (ACOMP, see chapter 16) connected to the same pins as the ADCs, and the transient disturbances emitted by the ADC at the start of **Track** can be large enough to falsely trigger the comparators. There are three types of disturbance in this state:

1. For any ADC measurement, the analog input MUX will inject a small charge when switching to select a pin. This can be suppressed with a filter capacitor on the PCB, see section 12.3.16. Note that the analog input MUX by design will not disturb a non-selected pin. Only pins selected by an active sampling buffer or from the ADC.ICFG register will be disturbed in this way.

2. For ADC measurements in bypass mode (which is always the case in single-ended mode), there will be a transient due to the charging/discharging of the S/H capacitor from/to the selected pin. See section 12.3.3 for a simplified circuit diagram. The size and direction of the transient depends on the previous voltage on the S/H capacitor relative to the capacitance and voltage of the PCB node connected to the pin to be sampled. Some control of this disturbance is possible by precharging the S/H capacitor, see section 12.3.17. For differential measurements there are two independent S/H capacitors, one for each pin of the differential pair to be sampled. See also section 12.3.14 for required track time for the S/H capacitor to settle.

3. For ADC measurements that use the preamplifier (only applicable to differential mode), the input impedance of the preamplifier will be connected to the pin pair during **Track**. The input impedance is composed of both a differential resistance between the pins, and a common mode resistance between both pins and an internal 1.5 V virtual ground. See section 12.3.3 for an equivalent circuit and section 12.3.2 for general information about the preamplifier. High impedance sources measured with the preamplifier will have a static gain error due to voltage division over the preamplifier input impedance, and at the start of **Track** there can be transients due to abrupt change of impedance.

12.3.16 Filter capacitor to reject short MUX disturbance transients

Always when ACOMP is used at the same time as *any* ADC measurement is executed on the same GPIO pin, false ACOMP switchings can occur. This is caused by analog switches in the internal ADC channel-selection MUX, which eject small amounts of charge back out on the GPIO pin. To guarantee that no such false ACOMP switchings occur, these charge transients need to be suppressed by a small low-pass filter capacitor to ground on PCB, $C_{LP,ACOMP}$, placed close to the GPIO pin. See Table 99 maximum amplitudes of this kind of transient for some example values of $C_{LP,ACOMP}$.

Table 99. Required low-pass filter capacitor, $C_{LP,ACOMP}$, on GPIO pin for guaranteed maximum GPIO transient, caused by ADC MUX switchings. Capacitance tolerance of ±20% taken into account for $C_{LP,ACOMP}$.

$C_{LP,ACOMP}$ [pF_{nom}]	Transient [$mV_{peak,max}$]
33	150
100	50
330	15
1000	5

Note that these switching transients are not related to the S/H capacitor(s), but they come from the internal ADC MUX connecting the GPIO pin to the ADC. They contain much lower charge levels than the S/H capacitor transients, and can therefore be filtered by relatively small capacitance values, see table 99 for recommended values. To prevent disturbance from the S/H capacitor(s) from triggering an ACOMP requires controlled precharge as described in section 12.3.17.

When no false ACOMP switching is allowed, and the GPIO pin is driven by a series resistor on PCB such as a current-limiting resistor of $330\Omega_{min}$ or the 100 nF/1 uF case in table 98, then $C_{LP,ACOMP}$ according to table 99 always is required. And consequently, when adding such a capacitor directly on the GPIO pin, the required ADC track time, t_{track}, needs to be adapted to the selected $C_{LP,ACOMP}$ and series resistor values, see C_{LP} resp R_{LP} in table 98.

There is one special ACOMP configuration case that may not require any filter capacitor, $C_{LP,ACOMP}$, on the GPIO pin. When ACOMP is configured to disturbance-tolerant/SET-hard mode, it has a pulse rejection time, t_{rej} (about 30 ns, see datasheet section 2.9 [DS]), which can be utilized. The RC time constant on the GPIO pin must be short enough to give re-settling back to the right differential voltage polarity, with a margin of $V_{IN,SW}$, within the ACOMP minimum pulse rejection time, $t_{rej,min}$, see datasheet section 2.9 [DS]. To achieve such fast re-settling on the GPIO pin, e.g., the total PCB-node and GPIO-pin capacitance may need to be in the order of 10 pF and the signal source resistance may need to be in the range 100-1000 Ω. And keep in mind that the re-settling time also depends on how many time constants of settling that are needed to achieve the right polarity plus a margin of $V_{IN,SW}$. The GPIO-pin conditions for this ACOMP pulse-rejection solution might be more or less difficult to fulfill in different applications, anyhow, the solution might become useful in some application cases.

12.3.17 Controlled precharge of S/H capacitor

For an ADC measurement in bypass mode, there will be a transient disturbance on the GPIO at the start of the **Track** state due to the previous voltage on the S/H capacitor (see Figure 22 for a circuit diagram). To allow some control over this transient, the ADC allows the S/H capacitor to be precharged to a user configurable deterministic voltage prior to entering the **Track** state. The available precharge sources are 0 V, the on-chip source/sensor channel, and external GPIO pins. When precharging is not enabled (which is the default configuration after reset), the S/H capacitor will be disconnected/floating when the ADC is not in the **Track** state. Then, the latest sampled voltage will be kept until internal leakage causes arbitrary drifts, which can be in the order of 10 mV/ms at high temperature.

There are three foreseen application use-cases for S/H capacitor precharge. The main use-case is in applications that sample GPIOs that are simultaneously used as analog comparator (ACOMP) inputs. By controlling the sign of the transient, false switching of the ACOMP due to the transient ejected by the ADC can be prevented. But note that a much smaller amount of charge is also ejected by the input MUX itself, see section 12.3.16, and this must also be accounted for in this application.

A second use-case for precharge can be to start the upcoming ADC track settling on a well controlled voltage in the S/H capacitor(s), known to be close to the final track-settling voltage, to minimize the required track time in the upcoming ADC measurement.

Thirdly, precharge to 0 V (or another accurate DC level) can allow reducing the track time for bypass measurement to below the settling time of the input circuit without affecting ADC offset or linearity, even though the track settling is incomplete. Track time and settling time constant (τ) must then be the same for all samples, and ADC precharge must be performed to an accurate DC level in between each conversion. If the above conditions are fulfilled, the incomplete settling will only induce a gain error (decreased gain), since constant settling time gives constant relative settling residual in exponential settlings. If the precharge is not completed to 0 V, but at least accurately to the same voltage level for all samples, this can be compensated by ADC offset calibration.

Precharge is performed in the **Idle** and/or **Precharge** states (see section 12.3.13) if the **Idle override** condition is enabled for those states. The discharge/precharge has an RC time-constant settling (τ with exponential settling) that depends on the internal S/H capacitance and DC source resistance. Table 100 lists all available precharge and discharge sources for each ADC together with typical and maximum time constant, τ. The actual time within these states needed for settling depends on how accurately settling is needed.

For example 4τ, giving about 2 % rest voltage, would mean a settling time of maximum 4*50 ns=200 ns, based on “Internal ground” in table 100. Continuing the example, to settle the S/H

capacitor voltage to <0.1 V from a starting voltage anywhere from 0 to 3.6 V would require discharge time up to 3.6 τ (note that exp(-3.6τ/τ)*3.6V=0.098V). Using discharge by the pulled down on-chip source/sensor input then means that up to 3.6*50 ns=180 ns is required. Precharge from external GPIO pin, driven by 0 Ω on PCB, gives the same required time, i.e., up to 3.6*50 ns=180 ns. But here, it will ensure settling to the GPIO DC voltage level instead, with accuracy better than 0.1 V.

With S/H capacitor discharged to <0.1 V, consequently, the next GPIO to be ADC bypass measured cannot be transiently disturbed by the S/H capacitor to any higher voltage than 0.1 V. If this GPIO is driven by a PCB signal with range of 0 V to 0.1 V, e.g. a current-sense resistor in ground line, the GPIO will never go above 0.1 V, not even transiently. Thus, on the same GPIO pin, also an ACOMP trig level of, e.g., 125 mV could be configured as an overcurrent limit protection. More generally, as long as the S/H capacitor precharge level is guaranteed to be lower than the maximum level of PCB normal signal range, an ACOMP trig level down to the normal-range maximum level can be configured on the same GPIO pin. Some tolerance and disturbance margin would still be recommended though, e.g., see the analog MUX switching transients below.

Note that with the S/H capacitor fully discharged to 0 V during ADC idle state, a GPIO transient could go all the way down to 0 V at the next ADC by-pass measurement, but it is strongly dependent on the signal source impedance on PCB how deep the transient will be. After the transient, the GPIO voltage re-settles to the PCB source voltage level with a time constant that depends on the PCB source impedance. As usual, this re-settling needs to be completed accurately within the selected track time, as described earlier. Hence, keep in mind that the purpose of precharge is not to eliminate GPIO transients, caused by ADC S/H capacitor in by-pass measurements. Again, the purpose is mainly to control the sign of the transient, e.g., to avoid false triggering of ACOMP comparators, and possibly other transient problems on PCB if this GPIO-pin net is used elsewhere on PCB.

Table 100. Time constant (τ) and final settling voltage for S/H capacitor precharge in ADC **Idle** and **Precharge** states when **Idle override** is in effect. When running consecutive conversions, the ADC controller can skip the **Idle** state. But if the **Precharge** state is enabled using ADC.ICFG.PCH and ADC.ICFG.PCHE, then the ADC always passes through this state. Additionally, note that in between **Idle** and **Precharge**, the ADC is always within the **Pretrack** state for at least one system clock cycle and **Idle override** is never in effect in the **Pretrack** state. See section 12.3.13 for details.

ADC unit	Typ τ [ns]	Max τ [ns]	Settling [V]	Analog MUX channel	ADC.ICFG Configuration
Any	N/A	N/A	N/A	Disconnected/floating S/H capacitor.	TRACK=0 or whenever **Idle override** not in effect.
Any	25	50	Pos: 0.0 Neg: 0.0	Internal ground via pull-down of on-chip source/sensor channel.	TRACK=1, SELOC=1 OCDI=0
ADC0	300	800	Pos: 0.9 - 1.7 Neg: 1.0	On-chip source/sensor channel. Voltage depends on Temp Sensor.	TRACK=1, SELOC=1 OCDI=1 Non-inverted[1]: CROSS=0 Inverted[1]: CROSS=1:
ADC1	50	100	Pos: 1.2 - 1.6 Neg: 0.3 - 0.4	On-chip source/sensor channel. Voltage depends on V_{DD_CORE}.	
ADC2	70	200	Pos: 1.9 Neg: 0	On-chip source/sensor channel. Voltage is V_{REFBUF}.	
ADC3	50	100	Pos: 1.36 - 1.60 Neg: 0.34 - 0.40	On-chip source/sensor channel. Voltage depends on V_{DDA_PLL}.	
Any	25	50	V_{GPIO}	External GPIO pin driven by **0 Ω**	TRACK=1, SELOC=0 See Table 101.
	50	100		External GPIO pin driven by **1 kΩ**	

Note 1: In 'Non-inverted' configuration, the positive ADC S/H capacitor is precharged by the 'Pos:' voltage, and the negative capacitor is precharged by the 'Neg:' voltage. In 'Inverted' configuration, the precharge is vice versa. Note that any upcoming *single-ended* ADC measurement will connect the *positive* ADC S/H capacitor to GPIO pin during **Track** (and negative capacitor floating), so 'Non-inverted' is used for 'Pos:' precharge and 'Inverted' for 'Neg:' precharge. Any precharge voltage present on the negative capacitor can only cause a disturbance on an external pin in a *differential* ADC measurement that bypasses the preamplifier.

Precharge is configured using the idle configuration register (ADC.ICFG). This register provides low-level control of the input MUX applied in some, but not all ADC states. For full details on available settings, refer to the register description in section 12.4.4. But for the purpose of precharge, the main feature configured by ADC.ICFG is the **Idle override** condition. **Idle override** can be configured to be in effect in the **Idle** and/or **Precharge** states (see section 12.3.13). Register values for selecting between all available precharge channels are listed in Table 101.

More generally, when **Idle override** is in effect, this has the following effects:

- The S/H capacitor is connected to an input channel if ADC.ICFG.TRACK=1. Otherwise it floats.
- The input MUX either selects a GPIO input pin/pin pair (MUX input enabled) or disconnects all inputs (MUX input disabled). If ADC.CFG.IE=0 and ADC.ICFG.IEO=1, then ADC.ICFG.IEI controls the MUX input enable setting during **Idle override** and the state machine controls it in all other states. For other possibilities, see the descriptions of register fields IE, IEI, and IEO in the ADC.CFG and ADC.ICFG registers in sections 12.4.1 and 12.4.4.
- Whether the input MUX selects a single GPIO pin or a differential pair is controlled by ADC.ICFG.SE. This also affects whether or not the N-side S/H capacitor is floating or connected.
- The GPIO input pin or pin pair selected by the input MUX is determined by ADC.ICFG.SELINP and ADC.ICFG.CROSS.
- Whether the S/H capacitor is connected to the pin selected by the input MUX, or to the on-chip source/sensor channel is determined by ADC.ICFG.SELOC. If the on-chip source/sensor channel is selected, then ADC.ICFG.CROSS determines the polarity of the precharge (see Table 100).
- Whether the on-chip source/sensor input is internally pulled down or connected to the source/sensor is determined by ADC.ICFG.OCDI and ADC.ICFG.OCDF.
- If ADC.ICFG.BYPE=1, then the preamplifier bypass setting is controlled by ADC.ICFG.BYP. Otherwise it remains set to the value configured by the most recently triggered sampling buffer. Note that the gain setting of the preamplifier is always taken from the most recently triggered sampling buffer and cannot be directly controlled during **Idle override**. Hence use of the preamplifier during precharge is not recommended.
- The unused mode of the preamplifier and the A/D converter is controlled by ADC.ICFG.PU and ADC.ICFG.CU.

Table 101. S/H capacitor precharge input connections when **Idle override** is in effect. The acronyms in the first row refers to a field in the ADC.ICFG register. For example, SELINP is short for ADC.ICFG.SELINP. For this table to apply, the following register field values are assumed: TRACK=1, BYPE=1, BYP=1, OCDF=0, IEF=1, IEI=1, and ADC.CFG.IE=0. See individual register field descriptions in section 12.4.4 for details.

N-side cap. [1)]	P-side cap.	SE	SELOC	IEO	OCDI	SELINP	CROSS
On-chip source, negative side	On-chip source, positive side	0	1	0	0	X	0
On-chip source, positive side	On-chip source, negative side	0	1	0	0	X	1
Internal ground	Internal ground	0	1	0	1	X	X
AIN1	AIN0	0	0	1	X	0b00	0
AIN0	AIN1	0	0	1	X	0b00	1
AIN3	AIN2	0	0	1	X	0b01	0
AIN2	AIN3	0	0	1	X	0b01	1
AIN5	AIN4	0	0	1	X	0b10	0
AIN4	AIN5	0	0	1	X	0b10	1
AIN7	AIN6	0	0	1	X	0b11	0
AIN6	AIN7	0	0	1	X	0b11	1
Floating	AIN0	1	0	1	X	0b00	0
Floating	AIN1	1	0	1	X	0b00	1
Floating	AIN2	1	0	1	X	0b01	0
Floating	AIN3	1	0	1	X	0b01	1
Floating	AIN4	1	0	1	X	0b10	0
Floating	AIN5	1	0	1	X	0b10	1
Floating	AIN6	1	0	1	X	0b11	0
Floating	AIN7	1	0	1	X	0b11	1

[1)] In single-ended mode only the P-side capacitor is used. Hence, when precharging prior to a single-ended measurement, only the precharge voltage of the P-side capacitor matters. See also Table 100.

12.3.18 Chip low-noise condition during ADC sampling and conversion

The LEON3FT microcontroller supports ADC operation in low-noise microcontroller mode, i.e. the LEON3FT microcontroller can disable the LEON3FT processor and peripherals not needed by the system, to minimize ADC measurement noise introduced internally by the LEON3FT microcontroller itself.

To be able to operate or be able to sample and wake-up the application shall:

- Set the PSR.PIL register low enough for processor to wake-up from expected interrupt
- Keep clocks enabled for peripherals generating the expected interrupt

Failure to keep expected interrupt source enabled will most likely result in the watchdog timer resetting the system.

For low noise sampling the user application shall:

- Enable ADC clock for channel to use
- Disable all other interfaces or peripherals not needed in the clock gating unit
- Enable interrupt generation in ADC or if the DMA is used the DMA controller can be set to generate the interrupt to wake-up the processor

Start sampling and set the LEON3FT microcontroller in power down mode. See 17.1.6.

12.3.19 Access control

ADC control unit status and configuration is done entirely through a memory mapped register interface, documented in section 12.4. The registers are directly accessible from the main processor, from the two RTAs and from GRDMAC2 port BM1. DMA capable peripherals in the system have indirect access via a bridge from the DMA bus. Registers in distinct ADCs, and in other peripherals on the same APB bus, can be accessed simultaneously without timing interference. But if registers in the same ADC are accessed simultaneously by two distinct AHB masters then one of the accesses will be delayed until the other completes, causing timing interference.

Write protection can be configured for accesses from the main CPU and from the DMA bus via address ranges in GRMEMPROT. See section 47. For the RTAs, write protection can be configured per ADC unit, or other APB peripheral. If a write access is attempted to a write protected area, an AHB ERROR response results for the master that attempted the access, without timing interference on other masters. Write protection from the BM1 interface of the GRDMAC2 is not implemented. Read access protection is not implemented.

Reading of registers can be done at any time without side effects on the ADC operation. But register reads may have a timing impact on the bus.

Since registers control some functions directly without shadow registers in between, writing to registers during a sampling operation can have undefined results.

An ADC can be powered down and made inactive by disabling it in the clock gating unit.

12.4 Registers

The set of configuration and status registers available for each ADC controller is documented in table 102. Detailed descriptions of each register follows in subsections. Note that the number of registers and address offsets in ADC3 are different than those of ADC0, ADC1, and ADC2.

Table 102. ADC, Preamplifier and Analog MUX status and control registers

APB address offset [1]	**Register name**	**Acronym** [5]
0x00	ADC control register	ADC.CFG
0x04	ADC status register	ADC.STS
0x08	ADC high threshold detection register	ADC.HT
0x0C	ADC Low threshold detection register	ADC.LT
0x10+0x10*m [2]	ADC Sampling Buffer m Control Register 1	ADC.SB.CFG1
0x14+0x10*m [2]	ADC Sampling Buffer m Control Register 2	ADC.SB.CFG2
0x18+0x10*m [2]	ADC Sampling Buffer m Control Register 3	ADC.SB.CFG3
0x1C+0x10*m [2]	ADC Sampling Buffer m Status and Result	ADC.SB.STS
0x50-0x5C (ADC0-2), 0x90-0x9C (ADC3) [3]	PROPRIETARY [4]	N/A
0x60 (ADC0-2), 0xA0 (ADC3) [3]	ADC Idle Configuration Register	ADC.ICFG
0x8010C00C [6]	ADC0-2 trim register, see section 8.3	ADCTRL

1) Each ADC has its own registers. The offset is relative to the base address of each ADC which is 0x80400000 for ADC0, 0x80401000 for ADC1, 0x80402000 for ADC2 and 0x80403000 for ADC3. See also section 2.10.

2) Each sampling buffer has an interface consisting of four consecutive registers CFG1, CFG2, CFG3, and STS. Here m denotes the sample buffer number. ADC0-2 each has 4 sampling buffers so for them m ranges from 0-3. ADC3 has 8 sampling buffers, so m ranges from 0 to 7 in this case.

3) Due to the different number of sampling buffers in ADC0-2 (4) compared to ADC3 (8), the ADC.ICFG and PROPRIETARY registers have a different address offset for ADC0-2 than they have in ADC3.

4) The proprietary registers can be read without side effects, but the data should be considered undefined. Writes to the proprietary registers will cause undefined behavior and should be avoided for correct operation.

5) Since each register exists in multiple copies, adding numbers can be used to distinguish them. For example, the ADC sampling buffer 3 status register of ADC2 can be referred to as ADC2.SB3.STS. But in the text in previous sections which specific instance of a register is being reference is not important and the numbers left out, e.g. the text might refer to ADC.SB.STS instead of ADC2.SB3.STS.

6) The ADC0-2 trim register (ADCTRM) at address 0x8010C00C is described in section 8.3. The trimming fields in this register must be set to 1 to reach the linearity specification in datasheet section 2.8 [DS].

12.4.1 ADC control register

Table 103. 0x00 - ADC.CFG - ADC, Preamplifier and Analog MUX Control register

31 24	23 16	15	14	13	12	11	10	9	8	7 6	5	4	3	2	1	0
AC	TSET		S1	S2	ST	IE	TE	MD	RD	PB	AE	PE	S3	SQ	SC	AS
0xFF	0xFF		0	0	0	0	0	0	1	0b00	0	0	0	0	0	0
rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

31: 24 ADC clock divider (AC) - The ADC clock frequency is (system frequency)/(ADC.CFG.AC +2). The maximum useful value is 0xFE. A value of 0xFF will cause the ADC clock to be stopped. See section 12.3.8 for permitted frequency range and effect on sampling rate.

23: 15 ADC Timeout (TSET) - Maximum number of system clock cycles that the control unit will wait for an end-of-conversion signal from the analog converter while in the **Hold/Convert** state if timeout is enabled by ADC.CFG.TE=1. If this timeout count is reached during a conversion, then the conversion is stopped, the ADC is forced into the **Idle** state, and ADC.CFG.AS is cleared (forcing the sequencing mode to *Idle*). The value of this field has no effect if ADC.CFG.TE=0.

Note: This field does *not* set the maximum time of the **Hold/Convert** state, just of a sub-operation that occurs within this state. If TSET>=14*(ADC.CFG.AC+2) (TBC), timeout will in the absence of SEE not occur in any conversion mode.

14 ADC sign compensation 1 (S1) - Must be set to 0 for correct operation.

13 ADC sign compensation 2 (S2) - Must be set to 1 for correct operation.

12 ADC Stop (ST) - ADC Stop. Writing 1 to this bit immediately stops any ongoing conversion, places the ADC in the **Idle** state and clears ADC.CFG.AS, forcing the sequencing mode to *Idle*. This bit self-clears after 1 system clock cycle.

11 ADC Input enable override (IE) - If IE=1, then the internal input-enable signal of the input MUX is controlled directly by ADC.ICFG.IE in all ADC states. If IE=0, then MUX input enable depends on ADC.ICFG and the sampling state. See sections 12.3.13 and 12.3.17. Recommended value: 0

10 ADC Timeout enable (TE) - Timeout Enable. When TE=1, timeout detection for the **Hold/Convert** state is enabled. See description of TSET above.

9 ADC Mode Gate (MD) - If MD=1, then the ADC will be set to differential mode during the **Idle** and **Pretrack** states. Otherwise the ADC differential/single ended setting during **Idle** and **Pretrack** is set by the most recently triggered sampling buffer. The **Precharge** state is not affected by this setting. If **Idle override** is configured to be enabled in the **Idle** state, then the setting in ADC.ICFG.SE takes priority over ADC.CFG.MD. Recommended value: 0

8 ADC Round enable (RD) - Enable rounding of ADC result prior to shift. This only affects sampling buffers that have ADC.SB.CFG3.SH>0. See section 12.3.7 for more details.

7: 6 ADC prebias (PB) - Analog prebias applied in conversion modes 1-3. Has no effect in conversion mode 0. 0b00 = none, 0b10 = positive, 0b01 = negative, 0b11 = none. Recommended setting: 0b00.

5 ADC Enable (AE) - When 0, the analog to digital converter is powered down. After setting AE=1, a minimum of 10 µs delay is needed to before starting a sampling operation.

4 ADC Preamplifier Enable (PE) - When 0, the preamplifier will be powered down. Must be set to 1 for correct operation during sampling. After setting PE=1, a minimum of 10 µs delay is needed to before starting a sampling operation.

3 ADC sign compensation 3 (S3) - Must be set to 1 for correct operation.

2 Enable synchronization triggers (SQ) - When 1, conversion will be delayed until ADC.CFG.AS=1 and a synchronization trigger is detected by at least one sampling buffer. When 0, only sampling buffer 0 will be used. This is the difference between the *direct* and the *triggered* sequencing modes of section 12.3.11.

1 Enable continuous sequencing (SC) - When 0, ADC.CFG.AS will be automatically cleared after a sampling buffer completes when conversion. When 1, ADC.CFG.AS will remain set after completion of conversions and the ADC is ready to start a new conversion immediately. If any sampling buffer is configured to use conversion mode 1, 2, or 3 then this bit must be set to 0 for correct operation. This is the difference between *single* and the *continuous* sequencing modes of section 12.3.11.

0 ADC Start (AS) - If AS=0, no new conversions will be started and all triggers are ignored. To start a sampling operation, set AS=1. If ADC.CFG.SQ=0 then doing so will immediately start a sampling operation for sampling buffer 0. If ADC.CFG.SQ=1, this will instead enable trigger inputs. If ADC.CFG.SC=0, then AS will be automatically cleared after completion of any sampling operation. See also section 12.3.11.

12.4.2 ADC status register

Table 104. 0x04 - ADC.STS - ADC status register

31	30	29	28	27	26	25	24	23 … 0
IDLE	IE	WSEE	WOF	TLO	DLO	CLO	DV	ACC
0	0	0	0	0	0	0	0	0x000000
r	r	r	r	r	r	r	r	r

31 ADC Idle State (IDLE) - This bit is 1 if the ADC is in the **Idle** state (see section 12.3.13).

30 End of sampling operation interrupt flag (IE) - Indicates that at least one sampling buffer has ADC.SB.STS.IE=1.

29 SEE warning flag (WSEE) - Indicates that at least one sampling buffer has ADC.SB.STS.WSEE=1.

28 Overflow warning flag (WOF) - Indicates that at least one sampling buffer has ADC.SB.STS.WOF=1.

27 Trigger lost flag (TLO) - Indicates that at least one sampling buffer has ADC.SB.STS.TLO=1.

26 Data Lost flag (DLO) - Indicates that at least one sampling buffer has ADC.SB.STS.DLO=1.

25 Conversion lost flag (CLO) - Indicates that at least one sampling buffer has ADC.SB.STS.CLO=1.

24 Data valid flag (DV) - Indicates that at least one sampling buffer has ADC.SB.STS.DV=1.

23: 0 ADC digital output accumulation register (ACC) - Accumulation register shared between all sampling buffers. Updates after every conversion within a sampling operation. Upon completion of the complete sampling operation, the value in ADC.STS.ACC is shifted and rounded before stored in ADC.SB.STS.DATA for the sampling buffer that started the sampling operation. See section 12.3.6 for data format.

Note: ADC.STS is a read-only global register that is common to all sampling buffers. The flags IE, WOF, TLO, LO, CLO and DV in this registers can only be cleared indirectly by clearing the corresponding flags in one or more ADC.SB.STS registers. See description of the ADC.SB.STS register in Table 111 for a detailed description of these fields.

12.4.3 ADC level detection threshold registers

Table 105. 0x08 - ADC.AHT -ADC high level detection threshold register

31 … 20	19 … 0
Reserved	AHT
0	0x00000
r	rw

31: 20 RESERVED

19: 0 ADC High Level detection threshold (AHT) - An interrupt will be generated if, at the end of a sampling operation in a sampling buffer with ADC.SB.CFG2.MH=1, the result (ADC.SB.STS.DATA) is strictly larger (>) than ADC.AHT. This threshold is shared between all sampling buffers.

Table 106. 0x0C - ADC.ALT -ADC low level detection threshold register

31 … 20	19 … 0
Reserved	ALT
0	0x00000
r	rw

31: 20 RESERVED

19: 0 ADC Low Level detection threshold (ALT) - An interrupt will be generated if, at the end of a sampling operation in a sampling buffer with ADC.SB.CFG2.ML=1, the result (ADC.SB.STS.DATA) is strictly smaller (<) than ADC.ALT. This threshold is shared between all sampling buffers.

12.4.4 ADC idle configuration register

Table 107. 0x60 (ADC0-2), 0xA0 (ADC3)-ADC.ICFG - ADC Idle Configuration Register

31	30	29	28	27	26	25	24 23	22	21	20	19 16
TRACK	BYP	CROSS	SE	SELOC	PU	AU	SELINP	OCDI	OCD	OCDF	PT
1	0	1	1	1	1	1	0x0	0	0	0	0x7
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14 9	8	7	6	5	4	3	2	1	0
PTE	PCH	PCHE	RES	IE	IEO	IED	RESERVED	BYPE	IO	IOC
0	0x05	0	0	0	0	1	0	0	0	0
rw	rw	rw	r	rw	rw	rw	r	rw	rw	rw

31 Track during **Idle override** (TRACK) - If set, then the S/H capacitor will be connected to a precharge source when **Idle override** is in effect. See Table 97 and Table 101.

30 Preamplifier Bypass during **Idle override** (BYP) - Controls preamplifier bypass during **Idle override** when BYPE=1. If BYPE=0, preamplifier bypass setting during **Idle override** is instead taken from the of the most recently triggered sampling buffer. Recommended value: 1

Note: The preamplifier gain setting is always taken from the most recently triggered sampling buffer. And is difficult to control. Therefore it is recommended to bypass the preamplifier for precharge operations.

29 Crossover control during **Idle override** (CROSS) - Together with SELINP and SELOC, controls input MUX channel selection for precharge of S/H capacitor during **Idle override**. See Table 101.

28 Single-ended during **Idle override** (SE) - Controls whether the S/H circuit and input MUX are configured in differential (0) or single-ended (1) mode when **Idle override** is in effect. See Table 101. During **Idle override**, this setting takes precedence over ADC.CFG.MD.

27 Select on-chip channel during **Idle override** (SELOC) - Together with CROSS and SELINP, controls input channel selection for precharge of S/H capacitor during **Idle override**. See Table 101.

26 Preamplifier unused during **Idle override** (PU) - Must be set to 1 for correct operation.

25 A/D converter unused during **Idle override** (AU) - Must be set to 1 for correct operation.

24:23 Input selection during **Idle Override** (SELINP) - Together with CROSS and SELOC controls input MUX channel selection during **Idle override**. See Table 101.

22 On-chip channel pull-down disable during **Idle override** (OCDI) - When **Idle override** *is* in effect, controls whether the on-chip channel is connected to internal ground (0) or to the on-chip source/sensor (1). See field OCD below for when **Idle override** is *not* in effect. See Table 97 and Table 101.

21 On-chip channel pull-down disable (OCD) - When **Idle override** is *not* in effect and OCDF=0, controls whether the on-chip ADC channel is connected to internal ground (OCD=0) or to the on-chip source/sensor (OCD=1). Recommended value: 1.

NOTE: This bit resets to 0, but must be set to 1 before the on-chip channel can be sampled.

20 On-chip channel disable filter (OCDF). Must be set to 0 for correct operation.

19:16 **Pretrack** extension cycles (PT) - The duration of the **Pretrack** state will be extended by PT+1 system clock cycles if PTE=1. If PTE=0, then the value of PT has no effect. See section 12.3.13 and Table 97. See also the description of PCH below.

15 **Pretrack** extension enable (PTE) - See description of PT field above.

14:9 **Precharge** extension cycles (PCH) - The duration of the **Precharge** state will be PCH+1 system clock cycles if PCHE=1. If PCHE=0 then the value of PCH has no effect. See section 12.3.13 and Table 97. See also the description of PT above.

8 **Precharge** extension enable (PCHE) - See description of PCH field above.

7 RESERVED

6 Input MUX enable during **Idle override** (IEI) - If ADC.CFG.IEO=0 and ADC.ICFG.IEO=1, then this bit sets the analog MUX input-enable during **Idle override**. Otherwise has no effect.

5 Input MUX enable override (IEO) - If this bit is 1, then the analog input MUX state is controlled by ADC.ICFG.IE during **Idle override**.

4 Input MUX enable (IE) - Analog input MUX enable in all states when IEO=1. Has no effect when IEO=0.

3 RESERVED

2 Bypass control enable during **Idle override** (BYPE) - See BYP above. Recommended value: 1

1 **Idle override** in **Precharge** state (IOC) - If 1, then **Idle override** will be in effect during the **Precharge** state. See section 12.3.13 and Table 97. See also IO field below.

0 **Idle override** in **Idle** state (IO) - If 1, then **Idle override** is will be in effect during the **Idle** state.

12.4.5 ADC sampling buffer registers

Note: In *direct* sequencing modes, only sampling buffer 0 is used. Other sampling buffers are only used in *triggered* sequencing modes. See section 12.3.11 for details.

Table 108. 0x10+*m**0x10 - ADC.SB.CFG1 - ADC Sampling Buffer *m* control register 1.
Note: *m*=0-3 for ADC0-2, *m*=0-7 for ADC3.

31 20	19 16	15 13	12 9	8	7 0
RESERVED	ACS	TSYNC	RESERVED	ASYNCM	ASYNC
0x0	0x0	0x3	0x0	0	0x0
r	rw	rw	r	rw	rw

31: 20 Reserved.

19: 16 ADC Channel select (ACS). Channel to sample for this sampling buffer, encoded as below:

ADC.SB.CFG1.ACS	Single ended mode [3] (ADC.SB.CFG2.AM=1)	Differential mode [4] (ADC.SB.CFG2.AM=0)
b0000	AIN0 [1]	AIN0 - AIN1 [1]
b0001	AIN1 [1]	AIN1 - AIN0 [1]
b0010	AIN2 [1]	AIN2 - AIN3 [1]
b0011	AIN3 [1]	AIN3 - AIN2 [1]
b0100	AIN4 [1]	AIN4 - AIN5 [1]
b0101	AIN5 [1]	AIN5 - AIN4 [1]
b0110	AIN6 [1]	AIN6 - AIN7 [1]
b0111	AIN7 [1]	AIN7 - AIN6 [1]
b1000	not allowed	+on-chip source/sensor [2]
b1001	not allowed	- on-chip source/sensor [2]
b1010-b1111	not allowed	not allowed

[1] For ADC0-2, AIN0-7 refers to AINA0-7 (GPIO37-44). For ADC3, AIN0-7 refers to AINB0-7 (GPIO51-58). See the IO configuration switch matrix in Table 7 and section 12.3.1 for details.

[2] Each ADC has an on-chip source/sensor input. Different sources/sensors are connected to each ADC. See section 12.3.4 for details and Figure 20 for a block diagram. The internal channel connection is also used for the S/H capacitor precharge feature (section 12.3.17) and can therefore be connected to internal ground via a configurable pull-down. Before sampling the on-chip source/sensor, this pull-down must be disabled by setting ADC.ICFG.OCD=1 (note that the default value at reset of this register field is 0).

[3] In single-ended mode, the preamplifier must be bypassed by setting ADC.SB.CFG2.AB=1.

[4] In differential mode, the internal crossover switch can invert the polarity of the input in the analog signal path before it reaches the ADC.

15: 13 Synchronization trigger type (TSYNC) - Select condition of trigger source to result in start of a sampling operation for this sampling buffer. The possible conditions are as below:

0b000 - Trigger on falling edges
0b001 - Trigger on low level
0b010 - Trigger on rising and falling edges
0b011 - Trigger on any level, but not on edges
0b100 - Trigger on rising and falling edges
0b101 - Trigger on any level, but not on edges
0b110 - Trigger on rising edges
0b111 - Trigger on high level

Note: Settings 0b011 and 0b101 are not expected to be useful in practice.

12: 9 RESERVED

8 Synchronization trigger mask (ASYNCM) - 0 disables the external trigger. 1 enables the external trigger. If ASYNCM=0, then the sampling buffer is effectively disabled in *triggered* sequencing modes.

7: 0 Synchronization trigger (ASYNC) - Index selecting the external trigger source. This is an index into the trigger sources listed in Table 96. Only values between 0 and 127 (inclusive) should be written to this field, indices larger than 127 must not be used.

Note: The TSYNC, ASYNCM, and ASYNC fields are only used in *triggered* sequencing modes (ADC.CFG.SQ=1) and are ignored in *direct* sequencing modes. See section 12.3.11 for details.

Table 109. 0x14+*m**0x10 - ADC.SB.CFG2 - Sampling Buffer *m* Control Register 2
Note: *m*=0-3 for ADC0-2, *m*=0-7 for ADC3.

31 30	29	28	27	26	25	24	23	22 21	20 19	18 17	16	15 0
RES	ES	IC	MH	ML	ME	AM	AB	AG	MDE	RES	HP	TT
0	0	0	0	0	0	0	1	0x0	0x0	0	0	0x63
r	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	rw	rw

31: 30 RESERVED

29 Enable SEE warning flag (ES) - If this bit is 1 and an SEE warning signature is detected during a sampling operation made by this buffer in conversion mode 2 or 3, then ADC.SB.STS.WSEE will be set when the sampling operation completes.

28 Interrupt Auto Clear (IC) - If this bit is 1, then ADC.SB.STS.IE for this sampling buffer is automatically cleared after one system clock cycle. Has no effect on other status bits.

27 Interrupt High Level Mask (MH) - When 1, enables high level threshold detection interrupts for sampling operations made by this sampling buffer. The high level threshold value is configured in ADC.AHT and is shared between all sampling buffers. See section 12.3.9.

26 Interrupt Low Level Mask (ML) - When 1, enables low level threshold detection interrupts for sampling operations made by this sampling buffer. The low level threshold value is configured in ADC.ALT. IRQ Mask Low Level Detection

25 ADC Buffer Interrupt End of Conversion (ME) - IRQ Mask End of Conversion

24 ADC Buffer single-ended (AM) - Set AM=1 and AB=1 for single-ended input. Set AM=0 for differential input.

23 ADC Buffer amplifier bypass (AB) - Set AB=0 to enable the preamplifier, and AB=1 to bypass the preamplifier. Note that single-ended inputs (AM=1) requires the preamplifier to be bypassed (AB=1).

22: 21 ADC Buffer amplifier gain (AG) - ADC amplifier gain setting. 0b00: x1, 0b01: x2, 0b10: x4, 0b11: x4. Has no effect if ADC.SB.CFG2.AB=1.

20: 19 ADC Buffer conversion mode (MDE) - This sets the resolution, see section 12.3.5

18: 17 RESERVED

16 High Priority (HP) - Enable high priority for buffer. When this bit is 1, and a trigger for this buffer occurs, then sampling of this buffer will start immediately even if a conversion is in progress for another sampling buffer (provided that other sampling buffer does not have high priority enabled).

15: 0 Track Time (TT) - The duration of the **Track** state will be TT+1 system clock cycles during sampling operations started by this sampling buffer. See section 12.3.14 for required minimum track time of GPIO analog inputs and section 12.3.4 for the required track time of on-chip source/sensor channels.

Table 110. 0x18+*m**0x10 - ADC.SB.CFG3 - ADC Sampling Buffer *m* Control Register 3
Note: *m*=0-3 for ADC0-2, *m*=0-7 for ADC3.

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12	11 10 9 8	7 6 5 4 3 2 1 0
RESERVED	SH	NO
0	0x0	0x0
r	rw	rw

31: 12 Reserved.

11: 8 Result bit shift (SH) - Prior to being stored in ADC.SB.STS.DATA, sampling operation results for this buffer will be right shifted by SH bits. If rounding is enabled in ADC.CFG.RD then the result will be rounded to the nearest integer. If rounding is disabled, the result will be truncated (rounded down).

7: 0 Number of oversamples (NO) - When this sampling buffer is triggered, the ADC will make NO+1 consecutive conversions. The 24-bit ADC.STS.ACC register field is used to hold the intermediate accumulated results while the final accumulated result is shifted and rounded before stored in the 20-bit ADC.SB.STS.DATA register. If NO=0, a single conversion will be made (no oversampling).

Table 111. 0x1C+*m**0x10 -ADC.SB.STS - ADC Buffer *m* status register (*m*=0-3 for ADC0-2, *m*=0-7 for ADC3).

31	30	29	28	27	26	25	24	23	22	21	20	19 0
TO	IH	IL	IE	TRG	WSEE	WOF	TLO	DLO	CLO	WST	DV	DATA
0	0	0	0	0	0	0	0	0	0	0	0	0
wc	wc	wc	wc	r	wc	wc	wc	wc	wc	r	wc	r

31 Timeout (TO) - This bit is set if a timeout occurs during a sampling operation start by this sampling buffer.

30 High Level Detection Interrupt Flag (IH) - This bit is set if, for this sampling buffer, ADC.SB.CFG2.MH=1 and ADC.SB.STS.DATA > ADC.AHT at the end of a sampling operation. Additionally, a system interrupt is generated if IH was 0 and becomes 1.

29 Low Level Detection Interrupt Flag (IL) - This bit is set if, for this sampling buffer, ADC.SB.CFG2.ML=1 and ADC.SB.STS.DATA < ADC.ALT at the end of a sampling operation. Additionally, a system interrupt is generated if IL was 0 and becomes 1.

28 End of sampling operation interrupt flag (IE) - This bit is set when a sampling operation completes for this sampling buffer, if ADC.SB.CFG2.ME=1. Additionally, a system interrupt is generated if IE was 0 and becomes 1. If, ADC.SB.CFG2.IC=1 for this sampling buffer, then ADC.SB.STS.IE self-clears after one system clock cycle.

27 Buffer trigger (TRG) - When 1, indicates that this sampling buffer has been triggered, but that the corresponding sampling operation has not completed yet. Automatically cleared by the control unit when the sampling operation completes.

26 SEE warning flag (WSEE) - In conversion modes 2 and 3, this bit will be set if during conversion the ADC detects an inconsistency. If this bit is set, the conversion result may be inaccurate.

25 Overflow warning flag (WOF) - In conversion modes 1, 2, and 3 this bit will be set if overflow or underflow occurred during a conversion. In case of underflow the conversion result is 0. In case of overflow the result is 0x3fff=2^{14}-1 in conversion mode 1, 0x7ffe=2^{15}-2 in conversion mode 2 and 0xfffc=2^{16}-4 in conversion mode 3. In case the conversion was part of an oversampling sequence (ADC.SB.CFG3.NO>0) then the averaged result of the sampling operation may be inaccurate.

24 Trigger Lost (TLO) - This bit is set if a trigger condition for this buffer occurs while ADC.SB.STS.TRG is already set. I.e. if a trigger for this buffer occurs before the sampling operation for a previous trigger has completed.

23 Data Lost (DLO) - This bit is set when a sampling operation for this buffer completes while ADC.SB.STS.DV=1. I.e. if it completes without ADC.SB.STS.DV having been cleared. Note that ADC.SB.STS.DV can only be cleared by writing a 1, it is not cleared by simply reading ADC.SB.STS. So DLO=1 does not necessarily mean data was lost.

22 Conversion Lost (CLO) - This bit is set if an ongoing sampling operation for this sampling buffer was canceled due to a trigger occurring on a high-priority sampling buffer.

21 Waiting to start (WST) - This bit is 1 whenever a trigger for this sampling buffer has occurred and the sampling operation has not yet started due to an ongoing sampling operation for a different sampling buffer.

20 Data Valid (DV) - This bit is set at the end of a sampling operation for this sampling buffer. Once this bit has been set, the result is readable from ADC.SB.STS.DATA. DV may optionally be cleared by writing 1. The only side effect of not clearing DV is that DLO (see above) will become set when the next sampling operation completes.

19 0 ADC sampling Buffer digital output (DATA) - The result of the most recent sampling operation completed by this sampling buffer. See section 12.3.6 for data format. This value changes only at the end of a sampling operation and remains valid until completion of the next sampling operation for the same sampling buffer. If enabled, shifting and rounding of the result is performed before it is stored in ADC.SB.STS.DATA.

Note: If oversampling is not enabled (ADC.SB.CFG3.NO=0) for this sampling buffer, then only DATA(10:0), DATA(13:0), DATA(14:0), and DATA(15:0) are non-zero in conversion mode 0, 1, 2, and 3 respectively. A small software optimization is then possible when reading the result from the main processor or from an RTA. Instead of making a 32-bit read and a subsequent AND/ANDN SPARC instruction to filter out flags, a 16-bit load may be made to the least significant half of the register. Since the system is big endian, the 16-bit load should be to address offset 2 relative to the 32-bit register address. I.e. at address offset 0x1E+*m**0x10 from the ADC base address.

13 LDO

13.1 Overview

The internal LDO supports single 3.3 V supply for the GR716B microcontroller. In single supply mode, the internal LDO supplies 1.8 V to V_{DD_CORE} internally. In dual supply mode, this LDO is not in use, and V_{DD_CORE} needs to be supplied by external 1.8 V supply from PCB. See figure 9 for LDO connections, and see datasheet section 2.12 [DS] for LDO electrical specifications.

In addition to the core current consumption, in single supply mode the internal LDO can also supply other 1.8 V loads on PCB. To use this feature, the other PCB loads are simply connected directly between the V_{DD_CORE} and GND planes on PCB.

13.2 Operation

13.2.1 System overview

The internal LDO provides the digital core with a regulated V_{DD_CORE} of 1.8 V, where the LDO output is directly connected to V_{DD_CORE} internally. The LDO needs a 3.3 V input supply connected to V_{DD_LDO}, and increases the internal power dissipation with the load current times the LDO voltage drop. Therefore, the maximum junction temperature should be carefully checked in applications where the core current, and consequently also the LDO current, can become high. Instead of using the internal LDO, the V_{DD_CORE} supply pins can be supplied directly from a 1.8 V supply on PCB, and V_{DD_LDO} should then be connected to V_{DD_CORE}.

The regulated LDO output voltage can be trimmed using the LDOTRM register. See section 8.2.4.

The LDO output voltage may be monitored using on-chip ADC1 via its on-chip source/sensor connection to V_{DD_CORE}. See section 12.

13.2.2 Detailed description

When the internal LDO is in use, i.e., single supply mode, the core supply current is drawn from the 3.3 V PCB supply into V_{DD_LDO}, and flows through the LDO and then internally connected to V_{DD_CORE}. Even though this LDO and connection to V_{DD_CORE} are internal, there must still be external decoupling capacitors connected to the V_{DD_CORE} supply pins on PCB, see below. For further recommended operating conditions for the LDO, see datasheet section 2.2 [DS].

The LDO will cause additional internal power dissipation, equal to the LDO average current times the voltage drop from V_{DD_LDO} to V_{DD_CORE}, which will further increase the GR716B junction temperature. Therefore, when running the core logic such that the core current, and thereby also the LDO current, is high, it is critical to carefully check that the maximum allowed junction temperature is never exceeded in the thermal situation at hand. This should be checked in all application implementations with the GR716B microcontroller, but it is especially important when the LDO is in use at the same time as core current can be high.

In dual supply mode, the V_{DD_CORE} pins are directly fed with 1.8 V regulated supply voltage from PCB, instead of using the internal LDO. In this case, the V_{DD_LDO} supply pins must *not* be connected to any low-impedance node other than V_{DD_CORE}. Another possibility is to leave the V_{DD_LDO} pins open (floating), but in space environment applications it is recommended to connect them to V_{DD_CORE}.

In regard to decoupling on V_{DD_CORE}, it should be done similarly whether the LDO is in use or not. It should always be in the order of 10nF ceramic capacitor per supply pin pair, and a larger, well damped capacitance bank somewhere nearby the GR716B package in the PCB layout. See datasheet section 2.2 [DS] for further details. When the LDO is in use, decoupling on the V_{DD_LDO} pins should be done similarly, i.e., in the order of 10nF per pin pair, and a larger, well damped capacitance bank some-

where nearby (which can be common to other 3.3V loads on PCB, to be decided at convenience by the PCB designer).

13.2.3 Pulsed load currents

The maximum average current rating for the LDO output is $700mA_{DC}$, and repetitive pulse load currents up to $1.1A_{Peak}$ are allowed for a limited pulse length, see datasheet section 2.2 [DS]. Note that the LDO output current is the sum of the internal core current and all external load currents connected to the V_{DD_CORE} net on PCB. This current sum cannot be measured, since the LDO output is connected internally to the core. However, since the quiescent current of the LDO is negligible (typically 2mA) compared to the LDO maximum ratings, this current sum can be measured on the V_{DD_LDO} pins (I_{DD_LDO}) accurately enough, and will in this discussion be regarded equal to the total LDO load current.

The LDO regulator does not guarantee full performance on the LDO output voltage (V_{DD_CORE}) for load currents above 0.7A, so the load current that exceeds 0.7A will be called ‘over-current’ here. Hence, to fully guarantee the output voltage performance during current pulses up to $1.1A_{Peak}$, the pulse duration needs to be short enough for charge in the external decoupling capacitors to handle the pulse with acceptable voltage drop.

The recommended decoupling on the LDO output (V_{DD_CORE}) is 2x (100uF+0.1Ω) for this LDO, see datasheet section 2.2 [DS]. Hence, a 0.4A over-current pulse (1.1A-0.7A=0.4A), with duration in the order of 10us, would give an additional voltage drop of up to 0.4A*50mΩ+0.4A*10us/200uF = 20mV+20mV = 40mV, if the whole over-current pulse would be drawn from the decoupling capacitors only. For a 0.1A over-current pulse, with duration in the order of 100us, it would be a voltage drop of 0.1A*50mΩ+0.1A*100us/200uF = 5mV+50mV = 55mV. But for such a long pulse time, the LDO regulator would compensate for the majority of the pulse drop, resulting in a voltage drop of significantly less than 40mV.

Whether the above added voltage drops are acceptable on the V_{DD_CORE} supply rail or not will depend on the application. For most digital-load applications, such as the microcontroller core logic, they would be highly likely to work perfectly well. Note however that the average load current, $I_{DD\_LDO,ave}$, must stay below 0.7A, to guarantee full DC-regulation performance and maintain device long-term reliability.

Typical applications where pulsed load design can be utilized are in design of real-time software cycles in RTA and LEON3 executions. For example, typical execution times for RTA control-loop routines for DC/DC converters can be in the order of 1-10 us, with cycle repetition times in the range of 2-20 us. According to above, up to 0.4A over-current pulses can be drawn for up to about 10us, which covers the maximum execution time and maximum current consumption for an RTA, see datasheet section 2.3 [DS]. In the software implementation, note that to take full advantage of reduced current consumption during the idle time in the cycle, it needs to be put in true halt/idle state and not continue to execute NOP, etc. The RTAs, therefore, have support for fast start up from halt/idle state.

14 Temperature Sensor

14.1 Overview

There is an integrated temperature sensor on the GR716B microcontroller, which can be used to supervise the die temperature.

14.2 Operation

14.2.1 System overview

The on-chip temperature sensor can be sampled via the internal ADC. It is not accessible externally.

14.2.2 Detailed description

The temperature-sensor is measured by the internal ADC in the same way as any other analog MUX channel. The sensor output signal is a linear voltage versus temperature in Kelvin. The junction temperature value in degree Celsius can be calculated according to the following formula:

$$TEMP = TBD$$

Here, formula constants should be calibrated, e.g. at room temperature, to achieve good accuracy from this sensor. Without calibration the tolerance could be up to ±TBD°C.

For automatic output threshold detection, e.g. used as over-temperature protection, the level detection feature in the internal ADC can be used to get an alarm (interrupt) when the temperature is above/ below a configured level. The alarm (interrupt) is not used in any internal hardware block, so if e.g. an over-temperature protection is desired the user needs to take adequate actions to handle the interrupt in the user system application.

14.2.3 Access control

The temperature sensor is always enabled and the output can be sampled with the ADC interface.

15 DAC

15.1 Overview

The GR716B contains 4 Digital-to-Analog Converters (DAC) of 12-bit resolution that can operate at up to 3 MS/s or 25 MS/s for ramp generation and with optional Dynamic Element Matching (DEM) for improved linearity. The DACs output a current in the range 0-4 mA, which can be converted to a voltage on PCB using a load resistor or a virtual op-amp ground. See datasheet section 2.6 [DS] for performance specifications. Section 2.3.6 shows the analog connections of the DAC, including shared comparators. Table 8 has additional information.

The analog DACs described in this section are distinct from the ΔΣ-modulators named APWMDAC (section 54). The APWMDAC may, together with low-pass filters on PCB, be used to create analog signals of resolution higher than the analog DACs, but at lower bandwidth.

Each DAC is interfaced via a control unit with register interface. The control unit includes logic for ramp generation for processor off-loading and for synchronization of DAC output to timers or external events. The DAC control unit registers are located on an APB bus in the address range from 0x80408000 to 0x8040BFFF. They can be accessed from the main processor, DMA controller and from both RTAs. Figure 25 shows the DAC units connections to pins and system bus.

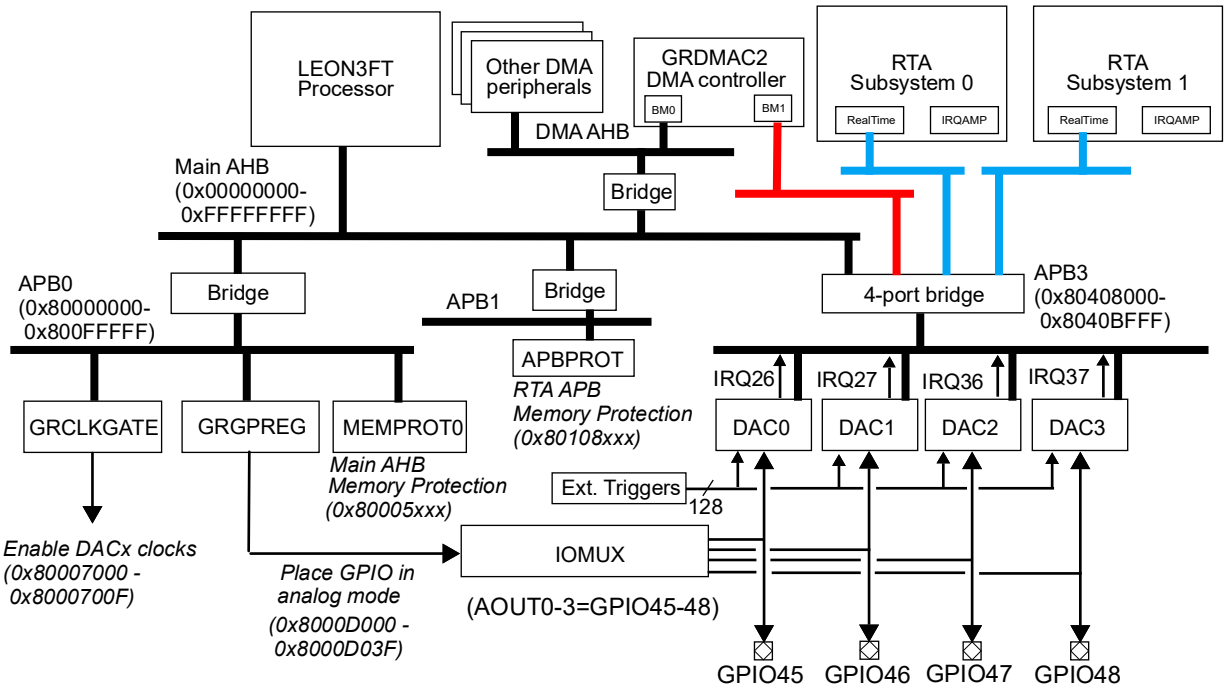


Figure 25. Partial bus diagram showing how the GR716B ADCs and related peripherals are connected to the system bus.

The primary clock gating unit **GRCLKGATE** described in section 27 is used to enable/disable individual DAC units. The unit **GRCLKGATE** can also be used to perform reset of individual DAC units. Software must enable clock and release reset described in section 27 before DAC configuration and transmission can start. When disabled in the clock gating unit, the analog part of the DAC is powered down. After enabling the DAC in the clock gating unit, it may be powered up by setting a bit in the DAC control register.

Each DAC unit is connected to a single GPIO pin without any MUX. However, the GPIO pins are shared with digital functions via the IOMUX. Before enabling DAC output on a pin, the digital function on that pin should be disabled in the IOMUX via the system IO configuration registers (**GRGPREG**) in the address range from 0x8000D000 to 0x8000D03F. See sections 7.1 and 15.4.1.

The system can be configured to protect and restrict access to individual DAC units in the **MEMPROT** unit. See section 47 for more information.

These four blocks are supplied by VDDA_DAC and VSSA_DAC. In the same way as for the ADC, good decoupling is needed at high frequencies (>~1 MHz).

15.2 Minimal steps to use the DAC with direct output

1. Configure the DAC output GPIO pin to be in analog mode. See sections 2.8.3 and 15.4.1.

2. Enable the DAC in the clock gating unit. See section 27.

3. Enable power to the DAC, configure the desired DAC clock frequency, and enable or disable DEM by writing to the DAC.CFG register. Set the DR field to 0 to use direct output (as opposed to ramp generator output). Provided 0 is written to the DTS field of this register, the DAC clock will start immediately.

4. Write the desired output value to the DAC.SDOUT register. The analog output current on the package pin will be changed to the new programmed value within 1 to 2 DAC clock cycles depending on when the write occurs relative to the phase of the DAC clock. Repeat this step as many times as desired.

15.3 Minimal steps to use the DAC self-timed ramp output

1. Configure the DAC output pin in analog mode. See sections 2.8.3 and 15.4.1.

2. Enable the DAC in the clock gating unit. See section 27.

3. Enable power to the DAC, configure the desired DAC clock frequency, and enable or disable by writing to the DAC.CFG register. Set the DR field to 1 to use ramp generator output. Since the DAC clock is not automatically synchronized to ramp generator updates, it is recommended to set DAC.CFG.DTS=1 to synchronize the DAC clock and the *ramp update events*.

4. Configure the ramp start (RSTART), ramp end (REND) and ramp step (RSTEP) parameters using the DAC.SRSTART, DAC.SREND, and DAC.RCFG1.SRSTEP shadow registers.

5. Set the ramp update rate (RDIV) by writing to the DAC.RCFG1.SRDIV shadow register. For constant delay between DAC clock and *ramp update events*, RDIV+1 must be an integer multiple 2*(DAC.CFG.AC+1) (i.e. an integer number of DAC clock cycles).

6. Select ramp waveform type (WT, *single ramp*, *triangle wave*, or *sawtooth wave*), set ramp generator to self-timed *ramp update events* (STEN=1), enable ramp kick (KICK) and enable the ramp generator (EN=1) by writing to the DA.RCFG2 register.

7. Configure the external trigger source to line 127 (constant low) in the DAC.SYNC register.

8. Set SQ=0 (to stop DAC clock) and TSYNC=0b001 (trigger on low level) in the DAC.SEQC register.

9. Start the DAC clock and the ramp by setting DAC.SEQC.SQ=1.

15.4 Detailed description

There are four independent DAC units that can operate at 12-bit and 3 MS/s with full analog performance, and digital ramp generation support up to 25 MS/s. They have sourcing-current single-ended outputs, typically to be loaded by virtual grounds generated by op-amps on PCB, or by passive impedances connected to PCB ground providing the output voltages directly across these impedances. The DAC full scale current is proportional to the current through the external RREF, and 5.11 kΩ gives a nominal full scale current of 4.0 mA. See datasheet section 2.6 [DS] for performance specifications. Figure 26 contains a block diagram of one DAC within its control unit

The DAC can optionally operate with Dynamic Element Matching (DEM) which improves linearity at the cost of added noise from switching transients. See section 15.4.2 for details.

The DAC clock frequency sets an upper limit to the conversion rate. When DEM is enabled, the DAC clock frequency also sets the DEM modulation rate. The DAC clock is generated by dividing the system clock by an even integer and can be synchronized to an external trigger, see section 15.4.5 for details.

Each DAC will convert a 12-bit digital value to an analog output current. The digital value can either be taken directly from a register (*direct output*), or from a digital ramp generator (*ramp output*). The selection between these two sources is made using the DAC.CFG.DR control register bit. Both the direct output and ramp output make use of shadow registers to allow updates of the DAC output to be synchronized to external trigger inputs. But the DAC may also be operated without triggers. Direct output is described in detail in section 15.4.6 and the ramp generator in section 15.4.7.

Triggers can be set to synchronize the DAC clock, DAC output register updates, *ramp kick events*, and *ramp update events* to external timers or events. See sections 15.4.5, 15.4.6 and 15.4.7 for details. The available trigger sources includes GPIO inputs, timer units and PWM ticks. The same trigger sources are available in the ADC (section 12.3.12). The trigger type is configurable to edge (rising, falling or both) or level (high or low). The available synchronization inputs are listed in section 15.4.3. When a selected external trigger occurs, this is referred to as a *synchronization event*.

The DAC control unit can be configured to generate an interrupt when the DAC output is updated with configurable delay (end of conversion interrupt, DAC.INT.EI). An interrupt can also be generated when the ramp generator reaches its programmed endpoint. Interrupt can be enabled to be generated when the DAC output is updated with configurable delay. Option for auto-clear and masking is supported.

Power down of the DAC is controlled via register (DAC.CFG.DE), which defaults to powered down after reset. After the DAC is powered up, a delay of 10 µs is needed before the DAC output provides full performance. In the powered state, its current consumption is independent of the output current, provided the DAC does not reach its compliance voltage. The total current-generator current is constant. The selected output code changes only what fraction of the current that flows out of the package pins and what fraction flows internally to ground.

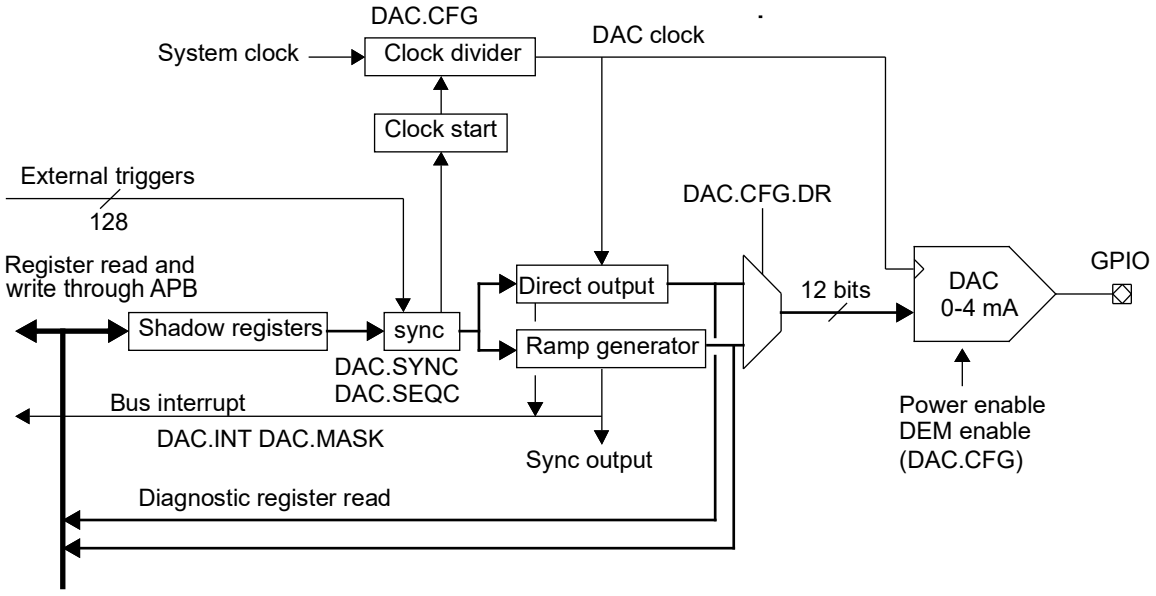


Figure 26. Block diagram of on-chip DAC control unit. Note that the ramp generator is *not* automatically synchronized to the DAC clock.

15.4.1 Analog output from GPIO

Each DAC output is connected to a fixed GPIO pin, with DAC0 connected to GPIO45 (AOUT0), DAC1 to GPIO46 (AOUT1), DAC2 to GPIO47 (AOUT2) and DAC3 to GPIO48 (AOUT3). These pins are also physically connected to digital GPIO functions. Prior to applying an analog signal level

on a GPIO pin, the pin must be placed in analog mode using the I/O switch matrix (see section 2.8.3). Long-term exposure of a GPIO pin not in analog mode to non-digital voltage levels will stress the digital input circuit, so software should perform this action early during startup. In analog mode, the digital output driver of the GPIO is in a high-impedance state and the digital input disabled, preventing internal current consumption and stress from non-digital signal levels. Analog-capable GPIO pins can be placed in analog mode by setting their I/O switch matrix configurations to 0x8 (see table Table 7 and section 7.1).

All DAC outputs GPIO pins are also connected to analog comparator inputs (ACOMP, see section 16). This allows a DAC to be used to generate a variable reference voltage for its associated comparator. DAC0/GPIO45 is internally connected to ACOMP8, DAC1/GPIO46 is connected to ACOMP9, DAC2/GPIO47 is connected to ACOMP10 and DAC3/GPIO48 is connected to ACOMP11. In all cases, it is the reference/negative input of the ACOMP that is connected to the DAC. Note that the GR716B DACs have current-output, not voltage-output. Hence resistors to ground on PCB are needed to convert the current to voltage, if the DAC outputs are to be used as reference inputs for ACOMP8-11.

15.4.2 Dynamic Element Matching (DEM)

The output current of the DAC is sourced from a large number of parallel current-sources that can be individually switched to the DAC output or to internal ground. With Dynamic Element Matching (DEM) disabled, the selection of current sources is static and changes only when the output code changes. This minimizes switching noise and offset/leakage current, but any mismatches between the individual current sources add up and contribute to integral non-linearity (INL).

The DAC can optionally be operated with DEM enabled, in which case current-sources are switched to and from the output pin on every DAC clock cycle. The sources are switched such that the number of connected sources is constant (if the digital output code is constant), but a different set of sources are connected on each cycle. The switching of sources has a period of 64 DAC clock cycles, and is done such that the average current over one complete switching period will perfectly cancel out all mismatches (on average each individual current source contributes in equal proportion). This eliminates the largest source of INL. And the constant switching rate creates a stable operating condition. But, for cases where the data output does not change on every DAC clock cycle, it adds noise from switching transients. And for integration times shorter than 64 DAC clock cycles, any mismatch is seen as noise (deterministic and periodic with period 64).

In applications where the DAC output is at a non-zero voltage rather than a virtual ground, the offset current of the DAC has a weak dependence on this voltage. The offset current is very low when current-source switches are in a stable condition. But during switching transients the current will, for a short time, deviate from the ideal with a dependence on the voltage. With DEM disabled, this means that the offset current may in general depend on how often the output code changes. With DEM enabled, current sources are switched to/from the output pin on every DAC clock cycle, so then the offset current depends only the DAC clock frequency and the voltage on the DAC output pin (assuming no other change in operating conditions).

For output codes less than 64, no switching will occur when DEM is enabled. But switching will occur for all output codes equal to or larger to 64 (this includes the max output code of 0xFFF=4095). If a non-zero voltage is applied to the DAC with DEM enabled, this causes a small voltage dependent discontinuity in the INL curve between codes 63 and 64.

15.4.3 DAC synchronization inputs and outputs

Each DAC can via *synchronization events* be synchronized to one of 128 external trigger sources. This can be used to synchronize the DAC clock (section 15.4.5), direct output updates (section 15.4.6), and *ramp update events* (section 15.4.7). Furthermore, *synchronization events* are required to trigger *ramp kick events* (section 15.4.7).

Which trigger source to use is selected via the 7-bit register field DAC.SYNC.SRC which controls a 128 to 1 MUX. A *synchronization event* can be configured to occur on an edge (falling/rising/any) or level (high/low) as selected by DAC.SEQC.TSYNC. After a trigger has occurred there is a delay of two system clock cycles until the synchronization event is generated. This latency is a consequence of how the edge/level detection logic is implemented. The available trigger sources are listed in Table 112. Note that the ADC units of the GR716B share the same 128 external trigger sources as the DAC units (see section 12.3.12).

Additionally, each DAC has a synchronization output. The synchronization output from DAC *n* is driven onto line 122+*n* of the shared external trigger bus. The source of the synchronization output is the register bit DAC*n*.INT.OI, ramp alarm interrupt, see section 15.4.4.

Table 112. DAC trigger input sources.

Trigger number (DAC.SYNC.SRC)	Description	Documented in
0	Timer unit 0 scaler tick	Section 35
1 to 7	Timer unit 0 counter 1-7 underflow (SRC=*n* selects counter *n*)	
8	Timer unit 1 scaler tick	Section 36
9 to 15	Timer unit 1 counter 1-7 underflow (SRC=8+*n* selects counter *n*)	
16 to 31	GPIO0-15 via Schmitt trigger and sampled into one flip-flop clocked by the system clock prior to reaching the DAC. (SRC=16+*n* selects GPIO*n*) Note that GPIO Schmitt trigger input paths are always active regardless of GPIO MUX configuration in SYS.CFG registers. For example, it is possible to use a GPIO configured as an output to generate a trigger.	This table.
32 to 87	APWM TickLines1(0 to 55) (SRC=32+*n* selects TickLines1(*n*))	Table 701
88 to 117	APWM TickLines0(34 to 63) (SRC=88+n selects TickLines0(34+*n*))	Table 700
118	RTA1 tick output 0 (local GPTIMER subtimer 1 tick)	Section 52.4
119	RTA1 tick output 1 (local GPTIMER subtimer 2 tick)	
120	RTA0 tick output 0 (local GPTIMER subtimer 1 tick)	
121	RTA0 tick output 1 (local GPTIMER subtimer 2 tick)	
122 to 125	DAC0-3 synchronization output (SRC=122+*n* selects DAC*n* synchronization output).	This section.
126 to 127	These two tick lines have the constant value 0. Can be used as continuous trigger by selecting low level trigger in the DAC.SEQC register.	This table.

15.4.4 DAC interrupts

The DAC contains two interrupt flag registers bits, DAC.INT.OI (ramp alarm) and DAC.INT.EI (end of conversion). A system interrupt will be generated whenever any of these flags changes from 0 to 1. DAC0, 1, 2, and 3 are connected to interrupt lines 26, 27, 36, and 37 respectively. See also Figure 25 and section 2.12. The DAC.INT register bits are cleared by writing 1.

If ramp alarm interrupts are enabled (DAC.MASK.OM=1 and DAC.RCFG2.ALEN=1), then DAC.INT.OI will be set to 1 whenever a *ramp alarm event* occurs. See section 15.4.7.2 for information about *ramp alarm events*. DAC.INT.OI is also used as synchronization output, see section 15.4.3.

If end of conversion interrupts are enabled (DAC.MASK.EM=1) then DAC.INT.EI will be set to 1 whenever the DAC.STS.DOUT register is updated with the value from the DAC.SDOUT shadow registers. The precise timing of the interrupt relative to the register update depends on other settings. See the register description in section 15.5.4 for details.

15.4.5 DAC clock generation and synchronization

The DAC control unit generates the DAC clock by dividing the system clock. When the clock divider is enabled, the DAC clock frequency is (system frequency)/(2*(DAC.CFG.DS+1)). The analog out-

put of the DAC changes only on rising edges of the DAC clock, so the DAC clock frequency sets the upper sample rate limit. When DEM is enabled (see section 15.4.2), the analog output noise and offset current also has a direct dependence on DAC clock frequency. The DAC remains functional for DAC clock frequencies up to 25 MHz and can be used for ramp generation up to this frequency, but full analog performance requires lower sampling rates. See datasheet section 2.7 [DS] for electrical characteristics.

The DAC clock is disabled at reset and must be enabled before the analog output of the DAC can be updated. The current status of the clock divider can be inspected by reading the DAC.STS.CLKEN register field. When the DAC is powered down (DAC.CFG.DE=0) the DAC clock is always disabled. The clock will also be disabled if DAC.SEQC.SQ=0 and DAC.CFG.DTS=1.

For *direct output* (section 15.4.6) it can be suitable to have the DAC clock running continuously whenever the DAC is powered up (DAC.CFG.DE=1). This will be the case if DAC.SEQC.SQ=0 and DAC.CFG.DTS=0. This method to start the DAC clock can be used also when synchronization is enabled (SQ=1) by first setting SQ=0 to start the clock and then setting SQ=1 to enable synchronization.

For applications where the precise phase of the DAC clock is important, the DAC clock can also be started by a *synchronization event* (section 15.4.3). Importantly, this allows to synchronize the DAC clock with *ramp update events* (section 15.4.7). It also permits the DAC clocks of different DAC units to be started simultaneously so they are exactly synchronous. Or to synchronize the DAC clock with ADC sampling, APWM timer ticks, or external events.

To synchronize the DAC clock, the clock must first be stopped by setting DAC.SEQC.SQ=0 and DAC.CFG.DTS=1. Then, synchronization is enabled by setting DAC.SEQC.SQ=1. When SQ=1, and DTS=1, the next *synchronization event* (as configured by DAC.SEQC and DAC.SYNC) will cause DAC clock divider to start. The DAC clock starts in the high state, and the time between the *synchronization event* and the first falling edge of the DAC clock is 3 or

3 + (DAC.CFG.DS - DAC.SEQC.DTSD)

system clock cycles, whichever is larger. Thereafter the DAC clock runs continuously with a period of exactly 2*(DAC.CFG+1) system clock cycles with a 50% duty cycle. The DAC.SEQC.DTSD field allows the phase of the DAC clock to be tuned relative to the *synchronization event* by up to one half DAC clock period.

Whenever the digital output code towards the DAC changes (the DAC.STS.DOUT or DAC.ROUT register), the analog output on the package pin is updated after one falling DAC clock edge and one rising DAC clock edge have occurred since the change in register value. The analog output changes on the rising DAC clock edge. The delay between an update of the digital output value and when the analog output changes therefore ranges between 0.5 DAC clock cycles (if the update happens just before a falling DAC clock edge) to 1.5 DAC clock cycles (if the update happens just after a falling DAC clock edge).

In the case of *direct output* (section 15.4.6), the DAC.STS.DOUT register can only change on rising DAC clock edges, so for direct output the delay ranges from 1 to 2 DAC clock cycles. In the case of ramp output, *ramp update events* are not automatically synchronized to the DAC clock so DAC.ROUT can change at any time relative to the DAC clock and the range of delay is 0.5 to 1.5 DAC clock cycles.

Unless data updates are synchronized to the DAC clock, the latency limits the maximum practical sample rate to approximately half of the DAC clock frequency. But when data updates are synchronized to the DAC clock, then the sampling rate can reach the DAC clock frequency.

15.4.6 Direct output

If DAC.CFG.DR=0 then the DAC is configured for *direct output*. With this setting, the digital output value is taken from the DAC.STS.DOUT register. The DOUT value is set indirectly via the DAC.SDOUT shadow register, which is writable from the main processor, RTAs and DMA.

When synchronization is disabled (DAC.CFG.SQ=0), DAC.SDOUT is copied into DAC.STS.DOUT on every rising DAC clock edge. The value takes effect on the analog output on package pin one DAC clock cycle later. Hence it may take anywhere from 1 to 2 DAC clock cycles after a write to DAC.SDOUT until the analog output changes.

When synchronization is enabled (DAC.CFG.SQ=1), the value in the DAC.SDOUT shadow register will be copied into DAC.STS.DOUT only when a *synchronization event* occurs (see section 15.4.3). The DOUT register is updated in the first rising DAC clock edge after the *synchronization event*. The analog output changes on the second rising DAC clock edge following the *synchronization event*.

15.4.7 Ramp generator output

If DAC.CFG.DR=1 then the DAC is configured for *ramp output*. With this setting the digital output value is taken from the DAC.ROUT register. DAC.ROUT is readable over APB for diagnostic purposes, but its value can only be changed by the ramp generator. The ramp generator updates DAC.ROUT (takes a step) according to current ramp parameters when a *ramp update event* occurs. Changing the ramp generator parameters requires a *ramp kick event*. How and when *ramp kick events* and *ramp update events* are generated is described in section 15.4.7.3.

The ramp generator has three basic integer parameters: ramp start (RSTART), ramp end (REND), and ramp step (RSTEP). RSTART and REND are unsigned 12-bit numbers (range 0-4095), while REND is a signed 2s-complement 8-bit number (range -128 to +127). The ramp starts with ROUT set to RSTART on a *ramp kick event*. Then, and on every *ramp update event*, RSTEP is added to ROUT resulting in next(ROUT)=prev(ROUT)+RSTEP. When ROUT reaches REND, the behavior differs depending on configured waveform type. The supported waveform types are *single ramp*, *sawtooth wave* and *triangle wave*, see section 15.4.7.1 for details.

Both upwards ramps (RSTEP>0) and downwards ramps (RSTEP<0) are supported, but REND and RSTART must be compatible with the direction. I.e. when RSTEP>0 it is necessary to set REND>RSTART. And when RSTEP<0 it is necessary to set REND<RSTART. If this rule is not followed, then all ramp update events will generate a *ramp alarm event* (section 15.4.7.2) and ROUT will remain constant for as long as the condition persists.

The ramp parameters are configured indirectly shadow registers SRSTART, SREND, SRSTEP, and SRDIV (section 15.5.5). The shadow registers are copied simultaneously into the current ramp configuration whenever a *ramp kick event* occurs (section 15.4.7.3). Simultaneously, ROUT is set to SRSTART.

15.4.7.1 Ramp waveform types

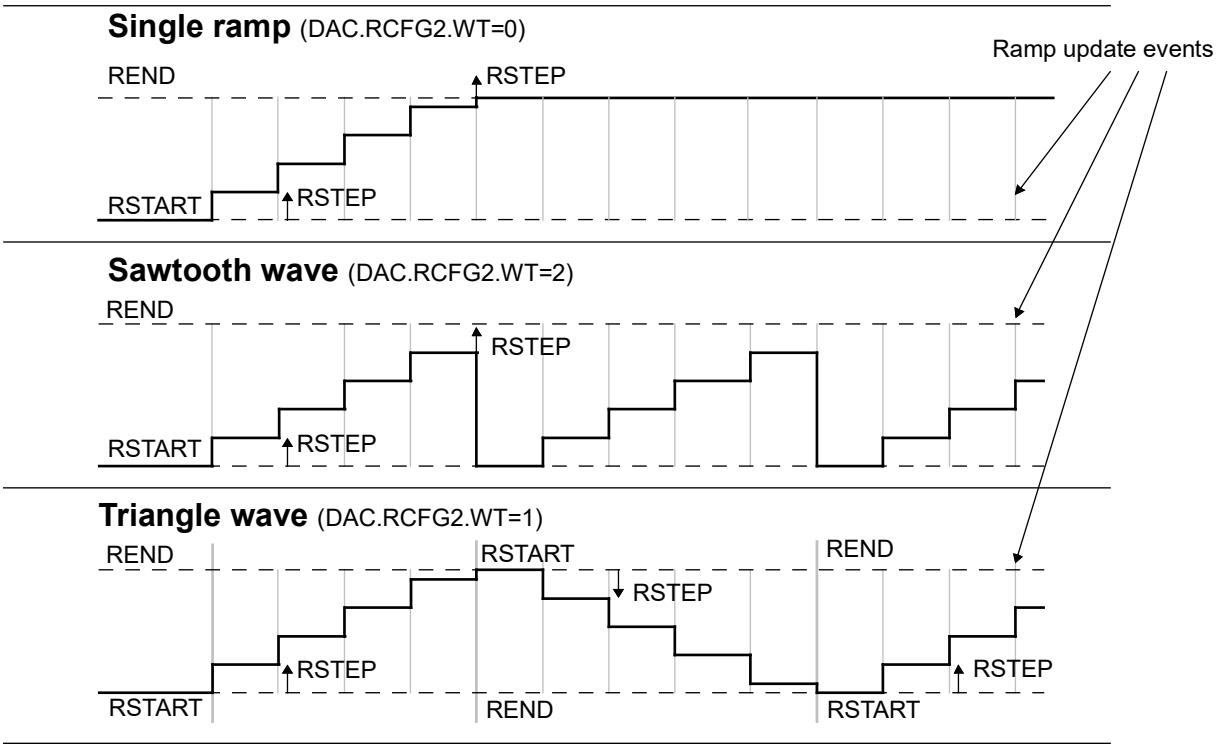


Figure 27. Ramp generator waveform type definitions highlighting the difference in behavior near REND.

For a valid combination of RSTART, REND and RSTEP, what happens when REND is reached depends on the ramp waveform type (WT). With the *single ramp* waveform type (WT=0), the ramp stops exactly at REND. If ROUT+RSTEP>=REND, then the next value of ROUT will be REND, i.e. the last step may be shorter than RSTEP. When RSTART, REND and RSTEP are a consistent combination, then ROUT reaches REND after exactly

N = ceil((REND-RSTART)/RSTEP)

ramp update events have occurred since the *ramp kick event*. Example: RSTART=0, REND=10, RSTEP=3 would result in the output sequence 0, 3, 6, 9, 10 after which the output remains constant at 10 until the *next ramp kick event*.

With the *sawtooth wave* waveform type (WT=2), the ramp never reaches REND. Whenever ROUT+RSTEP equals or passes REND, ROUT is set to RSTART instead of ROUT+RSTEP. The ramp generator output is therefore a periodic waveform with period of exactly

N = ceil((REND-RSTART)/RSTEP)

ramp update events. After *N ramp update events*, ROUT will equal RSTART again. Example: RSTART=0, REND=10, RSTEP=3 would result in the output sequence 0, 3, 6, 9, 0, etc. Note that since REND is a 12-bit unsigned number, the sawtooth wave cannot reach both DAC end points. The maximum value that can be output by the ramp generator using the sawtooth wave ramp type when RSTEP>0 is 0xFFE (when REND=0xFFF). The minimum value when RSTEP<0 is 0x001 (when REND=0).

With the *triangle wave* waveform type (WT=1), the ramp does reach the programmed value of REND. But when the programmed REND value is reached, the ramp generator swaps the contents of its internal RSTART and REND registers and negates RSTEP. It then steps in the opposite direction from the initial REND to the initial RSTART. This results in a periodic waveform with period of exactly

N = 2*ceil((REND-RSTART)/RSTEP)

ramp update events. It reaches the programmed REND value after *N*/2 *ramp update events* and returns to RSTART after *N ramp update events*. Note that if RSTEP is not an integer divisor of (REND-RSTART), then the upwards and downwards ramps will not be identical. Example: RSTART=0, REND=10, RSTEP=3 would result in the output sequence 0, 3, 6, 9, 10, 7, 4, 1, 0, etc. With the triangle wave waveform type, the normal useful range of RSTEP is -127 to +127. If RSTEP=-128, then the direction change will fail after REND is reached, because +128 is not representable in 2s-complement format using 8 bits. This generates a *ramp alarm event*.

15.4.7.2 Ramp alarm events

In general, a *ramp alarm event* will occur on a *ramp update event* if the current ramp output value (ROUT) has reached or exceeded the ramp end (REND). I.e. for an upwards ramp (RSTEP>0), it occurs if ROUT>=REND. For a downwards ramp (RSTEP<0), it instead occurs if ROUT<=REND. A *ramp alarm event* can be configured to generate an interrupt and synchronization output. See sections 15.4.3 and 15.4.4.

Ramp alarm events are a normal occurrence with the *single ramp* waveform type. However, with the *sawtooth wave* and *triangle wave* ramp waveform types, a *ramp alarm event* cannot occur under normal conditions. This is because a sawtooth wave will restart at RSTART before reaching REND. And a triangle wave swaps the identity of RSTART and REND when REND is reached. So with these ramp types, a *ramp alarm event* would be an indication either that a non-allowed combination of values were programmed when a *ramp kick event* occurred, or that a SEU has occurred in some ramp generator register.

15.4.7.3 Generating ramp kick and update events

There are two different ways to generate *ramp update events*: triggered and self-timed. This is selected by the register bit DAC.RCFG2.STEN. When STEN=0, *ramp update events* are triggered by *synchronization events* (section 15.4.5). A *ramp update event* will then occur two system clock cycles after any *synchronization event*. When STEN=1, *ramp update events* are self-timed and are generated once per DAC.RDIV+1 system clock cycles. Here DAC.RDIV is a shadowed register, its value is copied from DAC.RCFG1.SRDIV on any *ramp kick event*. The timing of triggered and self-timed *ramp update events* following a ramp *kick event* is illustrated in Figure 28.

Note that in neither case is the ramp output automatically synchronized to the DAC clock. After a change of ROUT, the analog output on the package pin does not change immediately, but only after one falling and one rising DAC clock edge have occurred as described in section 15.4.5.

Before the ramp generator can start at all, and whenever ramp parameters are changed, a *ramp kick event* is needed. A *ramp kick event* causes the values in the shadow registers DAC.SRSTART, DAC.SREND, DAC.RCFG1.SRSTEP and DAC.RCFG1.SRDIV to be copied into the actual ramp generator registers DAC.RSTART, DAC.REND, DAC.RSTEP and DAC.RDIV. It also causes DAC.ROUT to be set to DAC.SRSTART and the counter use for the timed ramp mode (DAC.RDIVCNT) to be reset.

To generate a *ramp kick event*, first set DAC.RCFG2.KICK=1. A *ramp kick event* will then occur one system clock cycle after the next *synchronization event* (15.4.5). The DAC.RCFG2.KICK register field self-clears after the *ramp kick event* has occurred. To use the ramp generator, a synchronization source must be configured, even when using self-timed *ramp update events*. The constant synchronization bus input lines may be used to defined a continuously active trigger (bottom row of Table 112), in which case a *synchronization event* would occur on every system clock cycle.

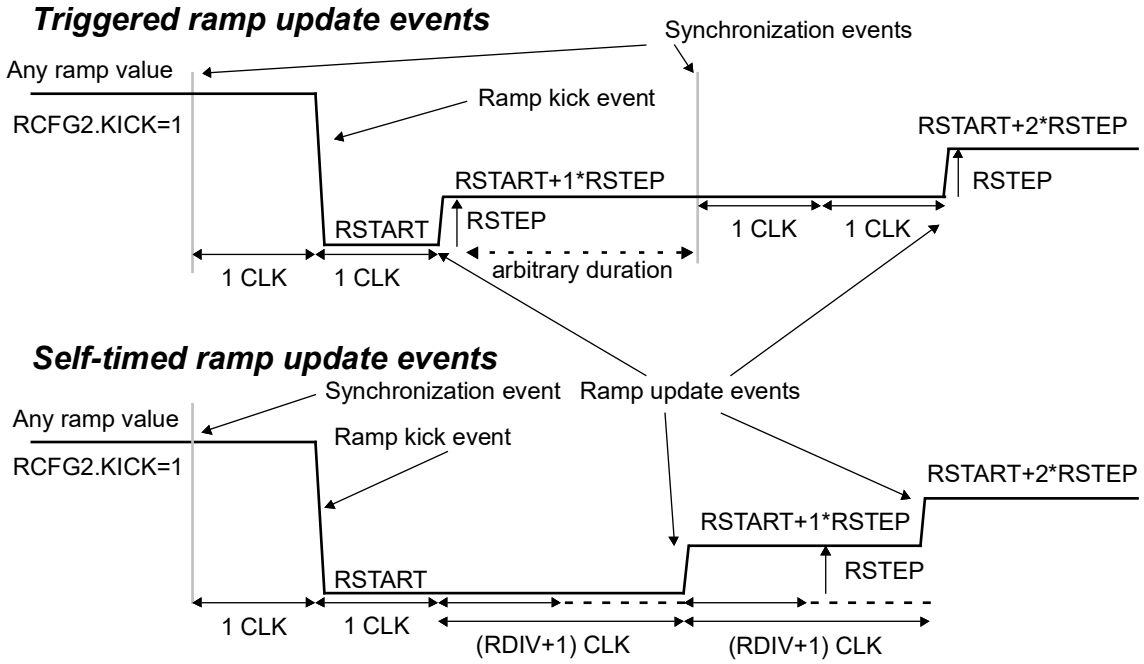


Figure 28. Timing diagram for a *ramp kick event* and the subsequent two *ramp update events*. Here "CLK" means "system clock cycles". Triggered ramp update events are used when DAC.RCFG2.STEN=0. Self-timed ramp update events are used when DAC.RCFG2.STEN=1.pdate events. Note that the diagrams show the value of the DAC.ROUT, which is not the same as the analog output value on package pin. DAC.ROUT is only applied after one falling and one rising DAC clock edge and the DAC clock is not automatically synchronized to either *ramp kick events* nor *ramp update events*.

15.4.7.4 Ramp update event race conditions

When using triggered *ramp update events*, the *synchronization event* that causes the *ramp kick event* will also trigger a *ramp update event* one system clock cycle after the *synchronization event*. Hence ROUT will only be at RSTART for one system clock cycle before changing to RSTART+ROUT. This is shown in Figure 28 together with the timing of *synchronization events*, *ramp kick events* and *ramp update events*.

For self-timed *ramp update events*, there is another kind of race condition. If a *ramp update event* occurs on the same system clock cycle as a *ramp kick event*, then ROUT will not be set to SRSTART by the *ramp kick event*. Instead, ROUT will be set to the value it would have had if the *ramp kick event* had not occurred (typically prev(ROUT)+prev(RSTEP)). Hence the ramp will start from the wrong level if the race condition occurs. If the application needs only a free-running ramp with constant RSTART, RSTEP and REND then no workaround is needed, otherwise care must be taken to prevent simultaneous *ramp kick events* and *ramp update events*.

One way to prevent this race condition is to use RDIV>=1 and a periodic source for the *synchronization events* with a period that is a multiple of (RDIV+1) system clock cycles. To see this, note that *ramp update events* will only occur at 2+*n**(RDIV+1) system clock cycles from the *synchronization event*, where *n* is any positive integer. If the period of *synchronization events* is *m**(RDIV+1) for some integer *m*, then the next *ramp kick event* will occur at 1+*m**(RDIV+1) system clock cycles relative to the previous *synchronization event*. Provided that RDIV>=1 there is no value of *n* for which 2+*n**(RDIV+1)=1+*m**(RDIV+1), so a *ramp kick event* then cannot occur on the same clock cycle as a *ramp update event*.

If the external synchronization input is not exactly periodic with the system clock, then a more general method to avoid the race condition is to stop the ramp generator before issuing a new *ramp kick event*. This can be done by setting DAC.RCFG2.EN=0 before setting DAC.RCFG2.KICK=1. This will however cause ROUT, RSTART, REND, RDIV and RDIVCNT to all be set to 0 before the kick event.

15.4.8 Configuration register access and access control

The configuration registers for DAC 0, 1, 2 and 3 can be accessed directly from the Main AHB, the BM1 interface of the GRDMAC2 DMA controller and from the two RTA subsystems. Indirect accesses from the DMA AHB are also possible via the bridge between the Main and DMA AHB buses. See Figure 25.

In case of simultaneous register access from multiple buses to the same DAC, wait states will be added on one or more of the competing buses. This applies not just to simultaneous accesses to the same register, but to any simultaneous accesses to addresses mapped to the same DAC. Registers in other peripherals on the same APB bus can however be accessed simultaneously without timing interference.

Write access from the Main AHB can be restricted using the MEMPROT0 memory protection unit. From the RTA subsystems, both read and write accesses can be restricted using the APBPROT registers. Access from GRDMAC2 BM1 is always unrestricted.

DAC register reads have no side effects, except for possible bus timing interference.

15.5 Registers

Table 113. DAC control registers

APB address offset[1)]	Register name	Acronym
0x00	DAC control register	DAC.CFG
0x04	DAC direct data output register (shadow register)	DAC.SDOUT
0x08	DAC sequence control register	DAC.SEQC
0x0C	DAC sequence synchronization register	DAC.SYNC
0x10	RESERVED	N/A
0x14	DAC interrupt and synchronization output register	DAC.INT
0x18	DAC interrupt and synchronization output mask register	DAC.MASK
0x1C	RESERVED	N/A
0x20	DAC ramp generation control register 1	DAC.RCFG1
0x24	DAC ramp generation control register 2	DAC.RCFG2
0x28	DAC ramp start configuration register (shadow register)	DAC.SRSTART
0x2C	DAC ramp end configuration register (shadow register)	DAC.SREND
0x30-0xD8	RESERVED	N/A
0xDC	DAC ramp start register (read-only)	DAC.RSTART
0xE0	DAC ramp end register (read-only)	DAC.REND
0xE4	DAC ramp step register (read-only)	DAC.RSTEP
0xE8	DAC ramp update rate register (read-only)	DAC.RDIV
0xEC	DAC ramp update rate counter register (read-only)	DAC.RDIVCNT
0xF0	DAC ramp data output register (read-only)	DAC.ROUT
0xF4	DAC shadowed status register (read-only)	DAC.STS
0xF8	DAC force trigger register 1 (write-only)	DAC.TRG1
0xFC	DAC force trigger register 2 (write-only)	DAC.TRG1

Note 1: Each DAC has an identical register interface at a distinct AMBA base address. The APB address offset is relative to this base address. The base address for DAC*x* is 0x80408000+0x1000**x*, i.e. the base address is 0x80408000 for DAC0, 0x80409000 for DAC1, 0x8040A000 for DAC2 and 0x8040B000 for DAC3.

15.5.1 DAC control register

Table 114. 0x00 - DAC.CFG - DAC control register

31 16	15 6	5	4	3	2	1	0
DS	RESERVED	DR	DC	DTS	R	DD	DE
0x0014	0	0	1	0	0	0	0
rw	r	rw	rw	rw	r	rw	rw

31: 16 DAC Scaler Divider (DS) - Sets the frequency of the DAC clock. When the DAC clock is enabled, the DAC clock frequency will be (system frequency)/(2*(DAC.CFG.DS +1)).

15: 6 Reserved

5 DAC Ramp Select (DR) - Selects the data source for the DAC. When 0, the DAC.STS.DOUT is used (indirectly set from DAC.SDOUT). When 1, the data is instead taken from the DAC.ROUT register (controlled by the ramp generator).

4 DAC Conversion interrupt delay (DC) - When DC=1 and the DAC.INT.EI interrupt is enabled, the end of conversion interrupt occurs on the same DAC clock rising edge as the analog output from the DAC changes on the package pin. If DC=0, the interrupt instead occurs one DAC clock cycle before the package pin is updated. This register field is only applicable when DAC.CFG.DTS=1. See DAC.INT.EI in Table 118 for details.

3 DAC Trigger Synchronize mode (DTS). If DRS=1, then the DAC clock will be started in phase with a external trigger. This feature is only supported for Sequence synchronization mode (SQ = 1) and requires the scaler (DS) to be set to 5 or higher. Set the DTS bit at the same time as SQ is cleared will turn off the DAC clock.

Using this feature requires a certain sequence for proper function.

1) Set DAC Enable (DE)
2) Clear SQ bit and set DTS bit
3) Set SQ bit

When the next trigger is generated, the DAC clock will be generated in phase with the trigger with a configurable phase shift of up to a half DAC clock cycle. See section 15.4.5 for details.

2 Reserved

1 DEM enable (DD) - Disables (0) or enables (1) DEM. See section 15.4.2 for details.

0 DAC Enable (DE) - When DE=1 the analog DAC circuitry is powered up. When DE=0 the analog DAC circuitry is powered down and the DAC clock is disabled. After DE is set to 1, a delay of up to 10 µs is needed before analog performance is guaranteed.

Note that the DAC output is enabled as soon as DE=1. Prior to the first few edges of the DAC clock, the output code depends on the power-up state of registers within the DAC. Normally this will be output code 0 resulting in a high impedance state, but spurious transients may in principle occur on the output when enabling the DAC. The value in SDOUT or DAC.ROUT or DAC.STS.DOUT will be used as output code as soon as the DAC clock is started.

15.5.2 DAC direct output shadow register

Table 115. 0x04 - DAC.SDOUT - Shadow register for direct DAC output value

31 12	11 0
RESERVED	SDOUT
0	0x000
r	rw

Table 115. 0x04 - DAC.SDOUT - Shadow register for direct DAC output value

31: 12 Reserved

11: 0 Shadow register for direct DAC output value (SDOUT) - Digital output code configuration for the DAC when in direct output mode (DAC.CFG.DR=0). This value is copied into DAC.STS.DOUT on the next rising edge of the DAC clock following a *DOUT update event* (see below). The value in DAC.STS.DOUT takes effect on the analog output on package pin one DAC clock cycle after being updated.

If DAC.SEQC.SQ=0 then *DOUT update events* will take place on every rising DAC clock edge and the delay between when a write to SDOUT and when the analog output on the package pin changes will be between 1 and 2 DAC clock cycles, depending on when the write occurred relative the generated DAC clock.

If DAC.SEQC.SQ=1 then a *DOUT update event* will take place on the next rising edge of the DAC clock after a a synchronization trigger has occurred, as configured in the DAC.SEQC and DAC.SYNC registers. The analog output on package pin will change between 1 and 2 DAC clock cycles after the synchronization trigger depending on when it occurs relative to the DAC clock.

15.5.3 DAC sequence and synchronization registers

Table 116. 0x08 - DAC.SEQC - DAC Sequence Control register

31	30 28	27	26	25 23	22 0
SQ	RESERVED	AC	RES	TSYNC	RESERVED
0	0	0	0	0b110	0
rw	r	rw	r	rw	r

31 Sequence synchronization enable (SQ) - This field has two uses. Firstly, it affects how the DAC clock is started. If SQ=0 then the DAC clock is started immediately when power is enabled via DAC.CFG.DE. If SQ=1, DAC clock start is delayed until a trigger condition occurs. See section 15.4.5 for details.

The second use is with *direct output* (DAC.CFG.DR=0). If SQ=0 then DAC.SDOUT is copied to DAC.STS.DOUT on every rising DAC clock edge. If SQ=1, DAC.SGOUR is only copied to DAC.STS.DOUT after a *synchronization event*.

30: 28 Reserved

27 Auto clear interrupt (AC) - If this bit is set to 1, then all interrupt status bits in the DAC.INT register will automatically self-clear after one clock cycle.

26 Reserved

25: 23 Synchronization trigger type (TSYNC) - Select condition of the trigger sources selected by DAC.SYNC at which a *synchronization event* will be generated. The possible conditions are as below:

0b000 - Trigger on falling edges	0b001 - Trigger on low level
0b010 - Trigger on rising and falling edges	0b011 - Trigger on any level, but not on edges
0b100 - Trigger on rising and falling edges	0b101 - Trigger on any level, but not on edges
0b110 - Trigger on rising edges	0b111 - Trigger on high level

Note: Settings 0b011 and 0b101 are not expected to be useful in practice.

22: 0 Reserved

Table 117. 0x0C - DAC.SYNC - DAC Sequence Synchronization register

31 12	11 8	7	6 0
RESERVED	DTSD	R	SRC
0	0x4	0	0
r	rw	r	rw

31: 12 Reserved

11: 8 DAC Trigger Synchronization Delay (DTSD) - This signal is used together with the DAC.CFG.DTS signal to set the DAC clock phase relative to the trigger source. When DTSD=1, the first falling edge of the DAC clock will occur 3, or 3+DAC.CFG.DS-DAC.SYNC.DTSD system clock cycles after the *synchronization event*, whichever is larger. Note that this allows shifting the phase of the clock relative to the trigger by up to half a DAC clock cycle, but not more. To use this feature requires DAC.CFG.DS>=5.

Table 117. 0x0C - DAC.SYNC - DAC Sequence Synchronization register

7	Reserved
6: 0	Synchronization trigger source (SRC) - Select trigger source. SRC is interpreted as a number in the range 0 to 127 and selects which trigger input source to use per Table 112. The trigger type (edge/level/etc) is set by DAC.SEQC.TSYNC (Table 116).

15.5.4 DAC interrupt and synchronization output registers

Table 118. 0x14 - DAC.INT - DAC Interrupt and synchronization output register

31 2	1	0
RESERVED	OI	EI
0	0	0
r	wc	wc

31: 2	Reserved.
1	Interrupt flag for ramp Overflow and alarm (OI) - This interrupt is enabled by setting DAC.MASK.OM=1 and DAC.RCFG1.ALEN=1. A ramp alarm occurs whenever a ramp update occurs and DAC.ROUT has reached or exceeded DAC.REND (i.e. when ROUT>=REND and RSTEP>0, or when ROUT<=REND and RSTEP<0). This register bit is directly connected to the DAC synchronization output which is available as trigger source for other DACs and ADCs, see section 15.4.3. Additionally, whenever OI changes from 0 to 1, a bus interrupt is generated. If DAC.DSEQC.AC=1 then OI self-clears after one system clock cycle.
0	Interrupt flag for DAC End of conversion (EI) - This interrupt is enabled by setting DAC.MASK.EM=1. A bus interrupt is generated whenever EI changes from 0 to 1. The conditions under which EI is set to 1 depends on other settings: DAC.SEQC.SQ=0: EI is set to 1 on every rising DAC clock edge. DAC.SEQC.SQ=1 and DAC.CFG.DC=0: When a trigger has occurred to cause the DAC output to be updated, EI is set to 1 on the rising edge of the DAC clock one DAC clock cycle before the analog output changes. DAC.SEQC.SQ=1 and DAC.CFG.DC=1: When a trigger has occurred to cause the DAC output to be updated, EI is set to 1 on the same rising DAC clock edge that the analog output changes. If DAC.DSEQC.AC=1 then EI self-clears after one system clock cycle.

Table 119. 0x18 - DAC.MASK - DAC Interrupt and synchronization output mask register

31 2	1	0
RESERVED	OM	EM
0	0	0
r	rw	rw

31: 2	
1	Interrupt Mask for ramp overflow and alarm (OM) - Interrupt and synchronization output disabled if OM=0, enabled if OM=1 and DAC.RCFG2.ALEN=1.
0	Interrupt Mask for DAC End of conversion (EM) - Interrupt disabled if EM=0, enabled if EM=1.

15.5.5 DAC ramp configuration registers

Table 120. 0x20 - DAC.RCFG1 - DAC Ramp Configuration register 1

31 28	27 16	15 8	7 0
RESERVED	SRDIV	RESERVED*	SRSTEP
0	0x014	0	0b00000001
r	rw	rw	rw

31: 28 Reserved

27: 16 Shadow register for DAC Ramp clock Divisor (SRDIV) - Sets the rate at which *ramp update events* are generated. The contents of this shadow register are copied into the DAC.RDIV register whenever a *ramp kick event* occurs. When enabled by setting DAC.RCFG2.STEN=1, the ramp generator will generate a *ramp update event* every DAC.RDIV+1 system clock cycles.

Note: The ramp generator is not automatically synchronized to the DAC clock.

15: 8 Reserved. * These bits are readable and writable, but have no effect on DAC operation.

7: 0 Shadow register for DAC ramp step (SRSTEP) - SRSTEP is an 8-bit 2s complement signed number (range -128 to +127). For example 0b11111110 encodes the number "-2" and 0b01000000 encodes the number "+64". The value in DAC.RCFG1.SRSTEP is copied into DAC.RSTEP whenever a ramp kick occurs. When a *ramp update event* occurs, DAC.RSTEP is added to DAC.ROUT. See section 15.4.7 for details.

Table 121. 0x24 - DAC.RCFG2 - DAC Ramp Configuration register 2

31 6	5 4	3	2	1	0
RESERVED	WT	ALEN	STEN	KICK	EN
0	0x0	1	0	0	0
r	rw	rw	rw	rw	rw

31: 6 Reserved.

5: 4 Ramp waveform type (WT) - Selects ramp generator waveform type: See Figure 27 for definitions.

0x0 - Single ramp, 0x1 - Triangle wave, 0x2 - Sawtooth wave, 0x3 - Invalid setting

This register field takes effect immediately when written (this is not a shadow register).

3 Enable ramp overflow alarm (ALEN) - Ramp alarm output is enabled when ALEN=1 and DAC.MASK.OM=1. A ramp overflow alarm occurs when DAC.ROUT reaches or exceeds DAC.REND. See description of DAC.INT.OI in Table 118 for details.

This register field takes effect immediately when written (this is not a shadow register).

2 Enable self-timed ramp update events (STEN) - Select if the ramp generator should use self-timed *ramp updated events* (STEN=1) or triggered *ramp update events* (STEN=0). In the self-timed case, a *ramp update event* occurs once every DAC.RDIV+1 system clock cycles. In the triggered case, *ramp update events* occur two system clock cycles after each *synchronization event*. See Figure 28 and sections 15.4.7 and 15.4.3 for details.

This register field takes effect immediately when written (this is not a shadow register).

1 Enable ramp kick (KICK) - If this bit is 1 then a *ramp kick event* will occur one system clock cycle after the next *synchronization event*. The KICK bit is automatically cleared on the same system clock cycle as the *ramp kick event* occurs. A *ramp kick event* causes the settings in the DAC.RCFG2.SRSTEP, DAC.RCFG2.SRDIV, DAC.SRSTART and DAC.SREND shadow registers to be copied into the DAC.RSTEP, DAC.RDIV, DAC.RSTART and DAC.REND registers respectively. It also causes DAC.ROUT to be set to DAC.RSTART, and resets the trigger counter DAC.RDIVCNT.

Note: If a *ramp kick event* occurs on the same system clock cycle as a *ramp update event*, then DAC.ROUT for the new ramp cycle will start at the previous DAC.ROUT+DAC.RSTEP instead of at DAC.SRSTART. It is the responsibility of the user application to ensure that this does not happen. The race condition will be triggered when using self-timed *ramp update events* unless precautions are taken. See section 15.4.7.4 for details.

0 Enable ramp generator (EN) - When EN=0, DAC.ROUT, DAC.RSTART, DAC.REND, DAC.RSTEP, and DAC.RDIV are forced to 0. When EN=1, the DAC ramp generator operates normally.

Table 122. 0x28 - DAC.SRSTART - Shadow register for ramp start

31 12	11 0
RESERVED	SRSTART
0	0x000
r	rw

31: 12 Reserved

11: 0 Shadow register for DAC ramp start (SRSTART) - SRSTART is copied into DAC.RSTART whenever a *ramp kick event* occurs. This is the only way to set a value for the DAC.RSTART register (Table 124).

Table 123. 0x2C - DAC.SREND - Shadow register for ramp end

31 12	11 0
RESERVED	SREND
0	0x000
r	rw

31: 12 Reserved

11: 0 Shadow register for DAC ramp end (SREND) - SREND is copied into DAC.REND whenever a *ramp kick event* occurs. This is the only way to set a value for the DAC.REND register (Table 125).

15.5.6 Shadowed DAC registers (read-only)

Table 124. 0xDC - DAC.RSTART - DAC current ramp start value register (read-only)

31 12	11 0
RESERVED	RSTART
0	0x000
r	r

31: 12 Reserved

11: 0 Current ramp generator start value (RSTART) - This register can only be indirectly written via the DAC.SRSTART shadow register. The value in DAC.SRSTART is copied to DAC.RSTART whenever a *ramp kick event* occurs. Additionally, with the triangle wave waveform type (DAC.RCFG2.WT=1), the value in DAC.RSTART will be swapped with the value in DAC.REND when the ramp end is reached. If the ramp generator is disabled (DAC.RCFG2.E=0) then DAC.RSTART will be set to 0x000.

Note: Readout of this register is not necessary for operating the DAC. It provides visibility into a shadowed register, which may be useful for debugging purposes during software development.

Table 125. 0xE0 - DAC.REND - DAC current ramp end value register (read-only)

31 12	11 0
RESERVED	REND
0	0x000
r	r

31: 12 Reserved

11: 0 Current ramp generator end value (REND) - This register can only be indirectly written via the DAC.SREND shadow register. DAC.SREND is copied to DAC.RSTART whenever a *ramp kick event* occurs. Additionally, with triangle wave as ramp waveform type (DAC.RCFG2.WT=1), the value in DAC.REND will be swapped with the value in DAC.RSTART when the ramp end is reached. If the ramp generator is disabled (DACRCFG2.E=0) then DAC.REND will be set to 0x000.

Note: Readout of this register is not necessary for operating the DAC. It provides visibility into a shadowed register, which may be useful for debugging purposes during software development.

Table 126. 0xE4 - DAC.RSTEP - DAC ramp step register (read-only)

31 8	7 0
RESERVED	RSTEP
0	0b00000000
r	r

31: 8 Reserved

7: 0 Current internal value of ramp step value (RSTEP) - This field is a signed number with range -128 to +127 in 2s complement format. When a *ramp update event* occurs, DAC.ROUT will be set to DAC.ROUT+DAC.RSTEP or updated according to ramp waveform type if DAC.ROUT has reached the ramp end.

RSTEP can only be written indirectly via the DAC.RCFG1.SRSTEP shadow register. The value in the shadow register is copied into DAC.RSTEP whenever a *ramp kick event* occurs. Additionally, with triangle wave as ramp waveform type (DAC.RCFG2.WT=1), the value in DAC.RSTEP will be negated whenever the ramp end is reached.

Note: Readout of this register is not necessary for operating the DAC. It provides visibility into a shadowed register, which may be useful for debugging purposes during software development.

Table 127. 0xE8 - DAC.RDIV - DAC ramp update rate register (read-only)

31 12	11 0
RESERVED	RDIV
0	0x000
r	

31: 12 Reserved

11: 0 Current internal value of ramp update rate (RDIV) - This field is used for self-timed *ramp update events*, when DAC.RCFG2.STEN=1. The ramp generator will then schedule one *ramp update event* every DAC.RDIV+1 system clock cycles. DAC.RDIV can only be written indirectly via the DAC.RCFG1.SRDIV shadow register. The value in the shadow register is copied into DAC.RDIV whenever a *ramp kick event* occurs.

Note that *ramp update events* are not automatically synchronized to the DAC clock.

Note: Readout of this register is not necessary for operating the DAC. It provides visibility into a shadowed register, which may be useful for debugging purposes during software development.

Table 128. 0xEC - DAC.RDIVCNT - DAC ramp update rate counter register (read-only)

31 12	11 0
RESERVED	RDIVCNT
0	0x000
r	r

31: 12 Reserved

11: 0 Current counter value for self-timed *ramp update events* (RDIVCNT) - When DAC.RCFG2.STEN=1, RDIVCNT increments by 1 on every system clock cycle. When RDIVCNT reaches DAC.RDIV then a *ramp update event* occurs.

Note: Readout of this register is not necessary for operating the DAC. It provides visibility into a shadowed register, which may be useful for debugging purposes during software development.

Table 129. 0xF0 - DAC.ROUT - DAC current ramp data output register (read-only)

31 12	11 0
RESERVED	ROUT
0	0x000
r	r

31: 12 Reserved

11: 0 Current ramp output value (ROUT) - This value is driven to the DAC analog component if DAC.CFG.DR=1. The value cannot be written directly, but is updated on *ramp update events* in accordance with the values in the DAC.RSTART, DAC.REND and DAC.RSTEP registers. and the ramp waveform type (DAC.RCFG2.WT). On *ramp kick events*, ROUT is set to the value in DAC.SRSTART.

When selected by DAC.CFG.DR=1, a change in the DAC.ROUT value will take effect on the analog output on package pin after one falling DAC clock edge and one rising DAC clock edge have occurred.

Note that ramp update events are not automatically synchronized to the DAC clock.

Note: Readout of this register is not necessary for operating the DAC. It provides visibility into a shadowed register, which may be useful for debugging purposes during software development.

Table 130. 0xF4 - DAC.STS - DAC status register (read-only)

31 16	15	14	13	12	11 0
DSC	DCE	DARM	IRQ	CLKEN	DOUT
0x0000	0	0	0	0	0x000
r	r	r	r	r	r

31: 16 DAC clock Scaler Count (DSC) - When DAC.STS.CLKEN=1, DSC increments by 1 on every system clock cycle. Whenever DSC reaches DAC.CFG.DS the DAC clock signal is inverted (a clock edge occurs) and DSC is reset to 0 resulting in a DAC clock cycle period of 2*(DAC.CFG.DS+1) system clock cycles.

15 DAC clock rising edge (DCE) - This bit will be 1 during one system clock cycle after a rising DAC clock edge.

14 Arm update of DAC direct output (DARM) - When DAC.STS.DARM=1, the value in the DAC.SDOUT shadow register will be copied into DAC.STS.DOUT on the next rising DAC clock edge. DARM clears automatically on a rising DAC clock edge.

When DAC.CFG.SQ=1, DARM is set to 1 whenever a *synchronization event* occurs. DARM is forced to zero whenever the DAC is powered down (DAC.CFG.DE=0). DARM is forced to 1 when DAC.CFG.SQ=0. When DAC.CFG.SQ=1, DARM is set to 1 on a *synchronization event*.

13 Bus interrupt (IRQ) - Pulses high for one clock cycle whenever a bus interrupt is generated.

12 DAC clock enable (CLKEN) - When CLKEN=0 the DAC clock is stopped. When CLKEN=1 the DAC clock is generated from the system clock with the period specified in DAC.CFG.DS.

CLKEN is forced to zero whenever the DAC is powered down (DAC.CFG.DE=0). CLKEN is also forced to zero if DAC.SEQC.SQ=0 and DAC.CFG.DTS=1.

CLKEN is forced to 1 if DAC.SEQC.SQ=0 and DAC.CFG.DTS=0.

If DAC.SEQC.SQ=1 and DAC.SEQC.DTS=1 then CLKEN will be set to 1 one system clock cycle after a *synchronization event* (as selected by DAC.SEQC and DAC.SYN) has occurred.

11: 0 Current direct output value (DOUT) - When using direct output (DAC.CFG.DR=0), this is the value sent to the analog DAC circuitry. When DOUT changes, the effect is seen on package pin after one falling and one rising DAC clock edge have occurred.

Note: Readout of this register is not necessary for operating the DAC. It provides visibility into shadowed registers, which may be useful for debugging purposes during software development.

15.5.7 DAC forced trigger registers

Table 131. 0xF8 - DAC.TRG1 - DAC forced trigger register 1 (write-only)

31 2	1	0
RESERVED	SE	CLKEN
0		
r	w	w

31: 2 Reserved

1 DAC synchronization event (SE) - Writing 1 to this bit will force a partial *synchronization event*. With direct output, this has the same effect as a regular synchronization (as configured in DAC.SEQC and DAC.SYNC). But writing this bit will not generate *ramp kick events* or *ramp update events*.

0 DAC clock enable (CLKEN) - The value written to this bit will overwrite the current value of DAC.STS.CLKEN.

Note: This register is not necessary for operating the DAC. The ability to write the fields in this register has been added as a debugging aid for forcing DAC clock start and *synchronization events* when using direct output.

Table 132. 0xFC - DAC.TRG2 - DAC forced trigger register 2 (write-only)

31 1	0
RESERVED	DARM
0	
r	w

31: 1 Reserved

0 Arm update of direct DAC output register (DARM) - The value written to this bit will overwrite the current value of DAC.STS.DARM. See Table 130 for details.

Note: This register is not necessary for operating the DAC.

16 Fast Analog Comparator (ACOMP)

16.1 Overview

The GR716B microcontroller contains 20 Fast Analog Comparators (ACOMP) units. The response time depends on operating mode (fast/disturbance-tolerant) and on overdrive voltage but can be in the range of 10-100 ns. They can use programmable internal reference voltage levels or take both inputs from external pins. See datasheet section 2.9 [DS] for electrical specification. The comparator analog are taken from the 16 ADC input pins and 4 DAC outputs pins. The pin connections are illustrated in figures 10, 11, and 12 in section 2.3 and defined by Table 8. The ACOMP digital outputs can be used as inputs to FIR filters (section 28) and most APWM units (section 53) for tasks such as duty cycle control, timestamps and alarm generation. They can also generate system interrupts directly.

The analog comparators (ACOMP) are controlled by an APB bus register interface mapped to the address range 0x80107000 to 0x80108FFF. They are accessible from the main processor, RTAs and the DMA. See bus diagram in Figure 29.

The secondary clock gating unit **GRCLKGATE** described in section 27 is used to enable/disable the ACOMP and FIR unit. The unit **GRCLKGATE** can also be used to perform reset of the ACOMP units. Software must enable clock and release reset described in section 27 before ACOMP configuration and operation. Additionally, the clock for APWM bus subsystem must be enabled (clock number 23 in the primary clock gating unit) must be enabled before the ACOMP register interface is functional.

System can be configured to protect and restrict access to ACOMP units in the **MEMPROT** unit. The **APBPROT** unit can be used to restrict access per RTA and per APB slave. See section 47 for more information.

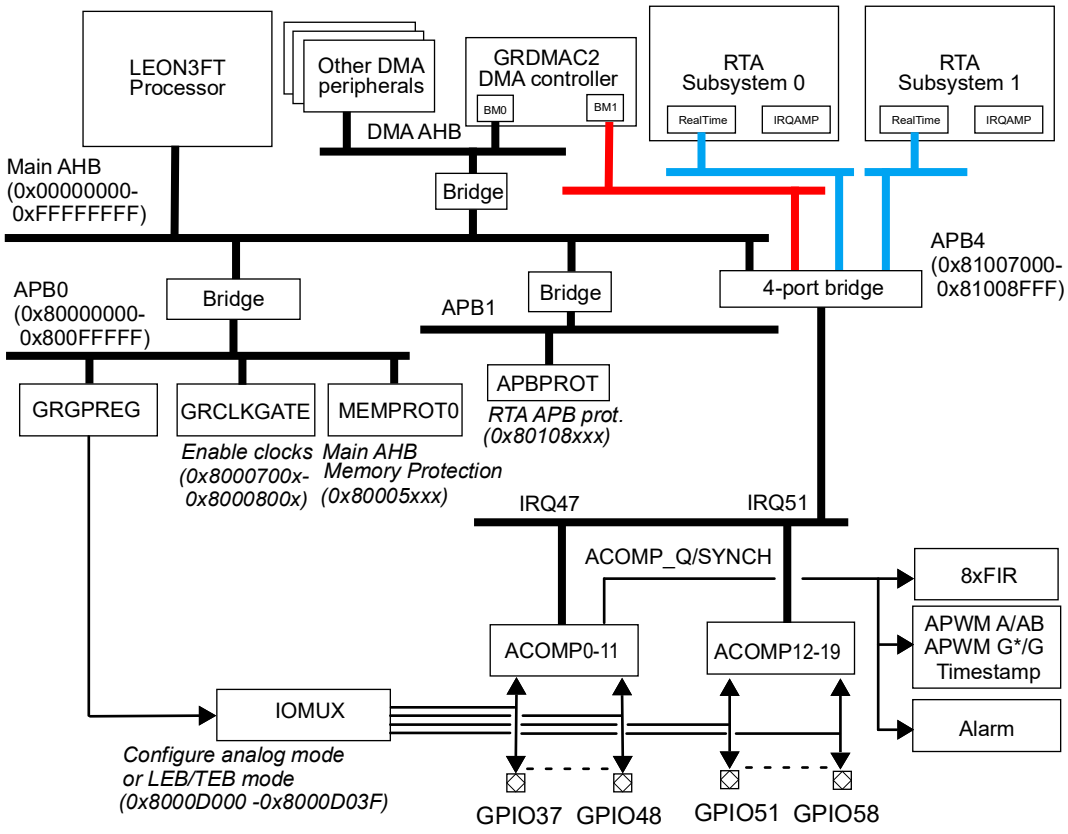


Figure 29. Partial bus diagram of GR716B ACOMP, related peripherals and pin connections.

16.2 Detailed description

The GR716B contains 20 analog comparators ACOMP0-19, each with associated control logic. They take their analog inputs from the 16 ADC input pins (GPIO37-44 and GPIO51-58) and the 4 DAC output pins (GPIO45-48). The pins connected to each comparator are defined in Table 8. The positive input of the comparator is always taken from a GPIO pin, but the negative input can be programmed to either use a GPIO input or an internally generated reference voltage of (nominally) 0 mV, 125 mV, 250 mV, 500 mV, 1000 mV, 1500 mV, or 2000 mV. In addition to the direct fast comparator output, a disturbance tolerant output that rejects short input transients is also available. This is illustrated in Figure 30. For analog characteristics of the comparators, such as offset voltage, hysteresis, response time, refer to datasheet sections [DS].

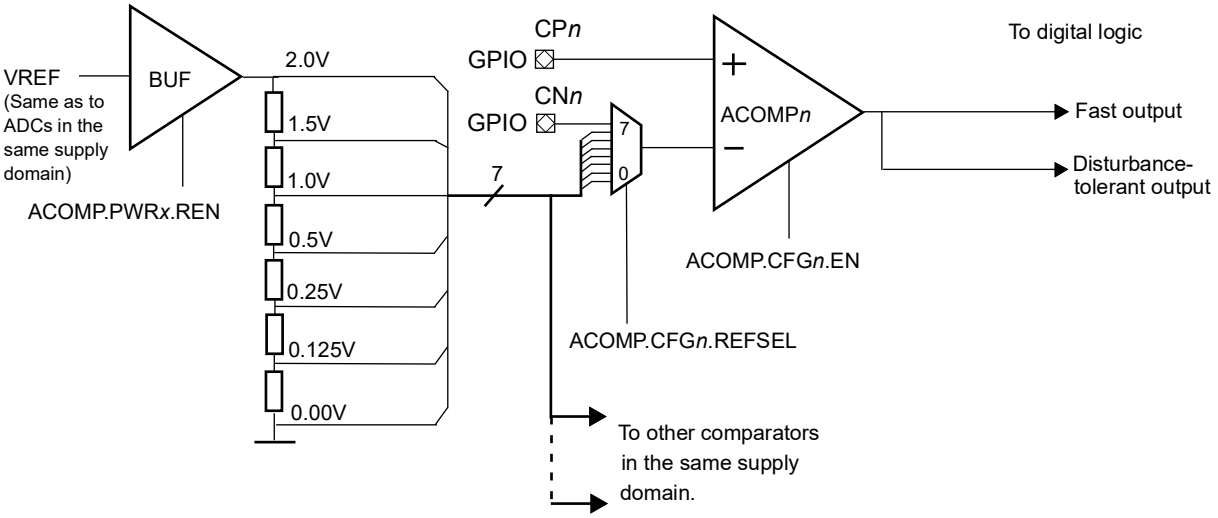


Figure 30. Block diagram of the analog connections of one of 20 ACOMP units in the GR716B. ACOMP0-11 all share one set of internally generated reference voltages. ACMP12-19 all share a second set of internal reference voltages.

Before processed by more complex digital logic, the output of each analog comparator passes through muxes, inverters and an SR latch as shown in Figure 31. The first mux allows to select the fast or disturbance tolerant outputs, or their digital inversion. The selected signal can optionally be used as the (active high) asynchronous clear input of the SR latch. Both the output of the SR latch and the selected signal itself are propagated to the rest of the system as the signals ACOMP_Q[*n*] and ACOMP_SYNCH[*n*] respectively (where *n* ranges from 0 to 19).

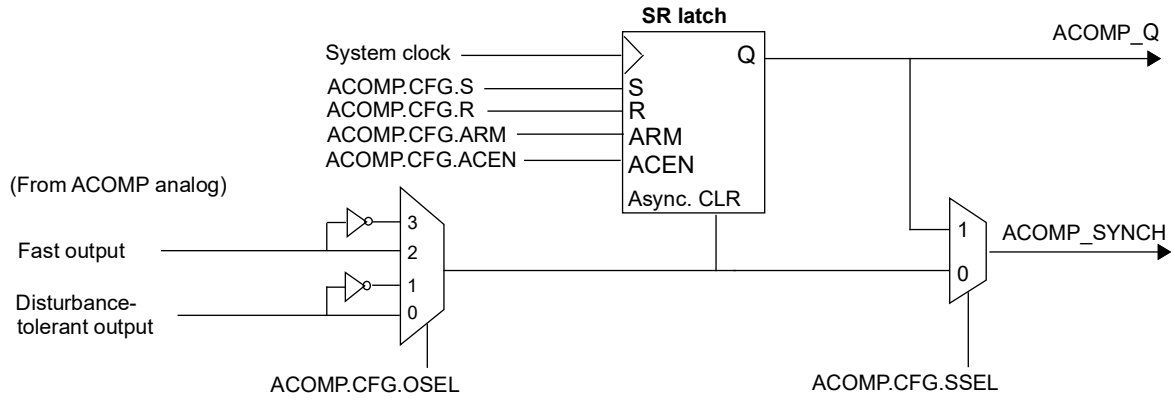


Figure 31. Block diagram of digital logic connected between one GR716B ACOMP unit and the rest of the system.

The Q and SYNCH outputs can be used as inputs to other blocks performing the following functions:

- Switching power and power application to provide PWM peak control (see section 53)
- General FIR filter for latch-up detection applications (see section 28)

16.2.1 ACOMP GPIO usage

The ACOMP analog inputs of the GR716B are connected to the same physical pins as digital GPIO functions. Prior to applying an analog voltage on a GPIO pin, the pin must be placed in analog mode using the I/O switch matrix. Long-term exposure of a GPIO pin not in analog mode to non-digital voltage levels will stress the digital input circuit, so software should perform this action early during startup. In analog mode, the digital output driver of the GPIO in a high-impedance state and the digital input disabled, preventing internal current consumption and stress from non-digital signal levels.

Analog-capable GPIO pins can be placed in analog mode by setting their I/O switch matrix configurations to 0x8 (see table Table 7 and section 7.1). This applies also to GPIO37-44 and GPIO51-58 when used as ADC inputs and GPIO45-48 when used as DAC outputs. See section 2.8.3 for more on analog mode. See Table 8 for which comparator is connected to which GPIO pin.

For comparators used in APWM_AB or APWM_A applications (see sections 53.2, 53.3 and 53.13), the GPIO input can also be configured for leading/trailing edge blanking (LEB/TEB) in the I/O switch matrix. Configuration 0x9 for APWM_AB LEB/TEB, and configuration 0xA for APWM_A LEB, see Table 7 for applicable GPIO pins. In LEB/TEB mode the GPIO will be in analog mode except for a short duration when the pin is blanked by the APWM unit. During the blanking duration, the GPIO driver is enabled in the low state, electrically forcing the pin to the VSSA_ADC/GND voltage. This feature allows the analog input of the comparator to be held at a stable condition during large PCB transients in APWM switched power applications.

Most ACOMP GPIO inputs are also ADC inputs (GPIO37-44 and GPIO51-58, see Table 8). The pins may be used simultaneously as comparator and ADC inputs. However, when sampling a GPIO pin, the ADC will eject transient disturbances that can be large enough to cause false switching of the ACOMP. Such false switching can be mitigated by the use of programmable ADC features combined with PCB level filters. See sections 12.3.15 and 12.3.16 for details.

The four comparators ACOMP 8-11 can optionally connect their negative inputs to GPIO45-48 which are also DAC outputs (see Table 8). This allows the reference voltage of these comparators to be tuned precisely or be varied in time as a linear ramp, see section 15 for DAC capabilities. But note that the GR716B DAC outputs are current sources. To use a DAC output to generate a reference voltage for a comparator therefore requires a resistor to ground on PCB to convert the current to a voltage.

16.2.2 Power down control

There are a total of 24 configuration register bits that control whether the comparators are powered down or in an active operational states. All 24 bits reset to 0, which powers down all comparators and reference voltage buffers. The comparators are split across two power supply domains, with ACOMP0-11 powered by V_{DDA_ADC} and ACOMP12-19 powered by V_{DD_IO}, and therefore separate bias control bits are provided for these two sets of comparators.

Each comparator supply domain has two control bits, one for the reference voltage buffer and one for bias current generation of the entire block. ACOMP.PWR0 for ACOMP0-11 and ACOMP.PWR1 for ACOMP12-19. Before any use of the comparators in a given domain, ACOMP.PWRx.BEN should be set to 1. And if the internal reference voltage is to be used (except when selecting 0.0 V), then ACOMP.PSRx.REN must also be set to 1. The remaining 20 bits are individual power control for each comparator set by ACOMP.CFG*n*.EN (*n*=0-19).

After enabling power to the bias current generation, reference voltage buffer, or any given comparator, they must be powered during a minimum of 10 µs before the comparator output is used.

16.2.3 SR latch logic table

Each comparator output is connected to an SR latch as depicted in Figure 31. The SR latch has one asynchronous clear input (CLR) that can be connected directly to the output of the comparator. When asynchronous clear is enabled (ACOMP.CFG*n*.ACEN=11b) a high value of CLR immediately causes the output (Q) of the SR latch to become 0.

The other SR latch inputs only take effect on rising system clock edges, and are driven directly from the comparator's configuration register (ACOMP.CFG*n*). They allow software to synchronously set or clear the state of the SR latch. The complete truth table of the latch is given in table 133.

Table 133. Truth table for SR latch. ACEN is in fact composed of two bits. In the table ACEN=1 corresponds to these bits being 0b11 while ACEN=0 corresponds to any other value (0b00, 0b01 or 0b10).

S	R	ARM	ACEN	CLR	Q
X	X	0	0	X	Retains its previous state
0	0	1	0	X	Retains its previous state
X	1	1	0	X	Becomes 0 on the next rising system clock edge
1	0	1	0	X	Becomes 1 on the next rising system clock edge
X	X	0	X	0	Retains its previous state
0	0	1	X	0	Retains its previous state
X	1	1	X	0	Becomes 0 on the next rising system clock edge
1	0	1	X	0	Becomes 1 on the next rising system clock edge
X	X	X	1	1	Asynchronously becomes 0

Additional SR latches and muxes as in Figure 31 exist for ACOMP-derived signals in the APWM_AB and APWM_A units. That logic is in that case controlled by registers in the APWM_AB and APWM_A units respectively. See sections 53.2 and 53.3 for details.

16.2.4 Direct status readout and interrupt generation

The state of all ACOMP digital outputs can be read at any time from the ACOMP.STAT0 and ACOMP.STAT1 registers. Using the ACOMP.IMASK0/1 and ACOMP.IEDGE0/1 register interface can generate system interrupts on interrupt line 47 and 51 in response to triggers. An interrupt will be generated whenever any of the interrupt flags changes from 0 to 1. Only edge triggered interrupts are implemented.

The ACOMP outputs are asynchronous to the system clock and not resynchronized to the system clock before used in interrupt generation. Therefore there is a non-negligible probability for the interrupt generation logic to miss edges on the ACOMP outputs. These interrupts cannot be relied on. For reliable interrupt generation from ACOMP inputs, the FIR unit is recommended. See section 28.

16.3 Registers

The cores are programmed through registers mapped into APB address space.

Table 134. ACOMP registers

AMBA ADDRESS	Register	Acronym
0x81007000	ACOMP0-11 Status register	ACOMP.STAT0
0x81007004	ACOMP0-11 Interrupt Flag register	ACOMP.IRQ0
0x81007008	ACOMP0-11 Interrupt Mask register	ACOMP.IMASK0
0x8100700C	ACOMP0-11 Interrupt Edge register	ACOMP.IEDGE0
0x81007010	ACOMP 0 Configuration register	ACOMP.CFG0
0x81007014	ACOMP 1 Configuration register	ACOMP.CFG1
0x81007018	ACOMP 2 Configuration register	ACOMP.CFG2
0x8100701C	ACOMP 3 Configuration register	ACOMP.CFG3
0x81007020	ACOMP 4 Configuration register	ACOMP.CFG4
0x81007024	ACOMP 5 Configuration register	ACOMP.CFG5
0x81007028	ACOMP 6 Configuration register	ACOMP.CFG6
0x8100702C	ACOMP 7 Configuration register	ACOMP.CFG7
0x81007030	ACOMP 8 Configuration register	ACOMP.CFG8
0x81007034	ACOMP 9 Configuration register	ACOMP.CFG9
0x81007038	ACOMP 10 Configuration register	ACOMP.CFG10
0x8100703C	ACOMP 11 Configuration register	ACOMP.CFG11
0x81007040	ACOMP0-11 Power Configuration register	ACOMP.PWR0
0x81007044 - 0x810070FC	RESERVED	N/A
0x81008000	ACOMP12-19 Status register	ACOMP.STAT1
0x81008004	ACOMP12-19 Interrupt Flag register	ACOMP.IRQ1
0x81008008	ACOMP12-19 Interrupt Mask register	ACOMP.IMASK1
0x8100800C	ACOMP12-19 Interrupt Polarity register	ACOMP.IEDGE1
0x81008010	ACOMP 12 Configuration register	ACOMP.CFG12
0x81008014	ACOMP 13 Configuration register	ACOMP.CFG13
0x81008018	ACOMP 14 Configuration register	ACOMP.CFG14
0x8100801C	ACOMP 15 Configuration register	ACOMP.CFG15
0x81008020	ACOMP 16 Configuration register	ACOMP.CFG16
0x81008024	ACOMP 17 Configuration register	ACOMP.CFG17
0x81008028	ACOMP 18 Configuration register	ACOMP.CFG18
0x8100802C	ACOMP 19 Configuration register	ACOMP.CFG19
0x81008030	ACOMP12-19 Power Configuration register	ACOMP.PWR1
0x81008034 - 0x810080FC	RESERVED	N/A

16.3.1 ACOMP Status registers

*Table 135.*0x81007000 - ACOMP.STAT0 - ACOMP0-11 Status register

31 25	24	23 22	21 20	19 18	17 16	15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
RESERVED	P	C11	C10	C9	C8	C7	C6	C5	C4	C3	C2	C1	C0
0	0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
r	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*

31: 25 Reserved

24 Configuration parity error status (P). This parity check covers the 13 registers ACOMP.CFG0-11 and ACOMP.PWR0. Becomes 1 when a configuration parity error is detected.

23: 22 ACOMP11 output status (C11).

21: 20 ACOMP10 output status (C10).

19: 18 ACOMP9 output status (C9).

17: 16 ACOMP8 output status (C8).

15: 14 ACOMP7 output status (C7).

13: 12 ACOMP6 output status (C6).

11: 10 ACOMP5 output status (C5).

9: 8 ACOMP4 output status (C4).

7: 6 ACOMP3 output status (C3).

5: 4 ACOMP2 output status (C2).

3: 2 ACOMP1 output status (C1).

1: 0 ACOMP0 output status (C0).

Each comparator status output (C*n*, *n*=0-11) listed above is composed of two bits:
C*n*[1]: ACOMP_SYNCH[*n*] output (from ACOMP*n*)
C*n*[0]: ACOMP_Q[*n*] output (from ACOMP*n*)

* For testing purposes, bits 24:0 in this register are writable. The written data overrides the normal data source for exactly one system clock cycle. This provides a means for triggering the associated interrupt from software.

*Table 136.*0x81008000 - ACOMP.STAT1 - ACOMP12-19 Status register

31 17	16	15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
RESERVED	P	C19	C18	C17	C16	C15	C14	C13	C12
0	0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
r	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*

31: 17 Reserved

16 Configuration parity error status (P). This parity check covers the 13 registers ACOMP.CFG12-19 and ACOMP.PWR1. Becomes 1 when a configuration parity error is detected.

15: 14 ACOMP19 output status (C19).

13: 12 ACOMP18 output status (C18).

11: 10 ACOMP17 output status (C17).

9: 8 ACOMP16 output status (C16).

7: 6 ACOMP15 output status (C15).

5: 4 ACOMP14 output status (C14).

3: 2 ACOMP13 output status (C13).

1: 0 ACOMP12 output status (C12).

Each comparator status output (C*n*, *n*=12-19) listed above is composed of two bits:
C*n*[1]: ACOMP_SYNCH[*n*] output (from ACOMP*n*)
C*n*[0]: ACOMP_Q[*n*] output (from ACOMP*n*)

16.3.2 ACOMP Interrupt Flag Registers

*Table 137.*0x81007004 - ACOMP.IRQ0 - ACOMP0-11 Interrupt Flag register

31	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	P	C11		C10		C9		C8		C7		C6		C5		C4		C3		C2		C1		C0	
0	0	0x0		0x0		0x0		0x0		0x0		0x0		0x0		0x0		0x0		0x0		0x0		0x0	
r	wc	wc		wc		wc		wc		wc		wc		wc		wc		wc		wc		wc		wc	

31: 25 RESERVED

24: 0 Bit *x* of this register is set to 1 whenever an interrupt has been generated due to bit *x* of ACOMP.STAT0. Each bit in this stays set to 1 until software clears it by writing 1. Whenever any bit in this register changes from 0 to 1, a system interrupt is generated on interrupt line 47.

*Table 138.*0x81008004 - ACOMP.IRQ1 - ACOMP12-19 Interrupt Flag register

31	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED		P	C19		C18		C17		C16		C15		C14		C13		C12	
0		0	0x0		0x0		0x0		0x0		0x0		0x0		0x0		0x0	
r		r	r		r		r		r		r		r		r		r	

31: 17 Reserved

16: 0 Bit *x* of this register is set to 1 whenever an interrupt has been generated due to bit *x* of ACOMP.STAT1. Each bit in this stays set to 1 until software clears it by writing 1. Whenever any bit in this register changes from 0 to 1, a system interrupt is generated on interrupt line 51.

16.3.3 ACOMP Interrupt Mask Registers

*Table 139.*0x81007008 - ACOMP.IMASK0 - ACOMP0-11 Interrupt Mask register

31	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED		P	C11		C10		C9		C8		C7		C6		C5		C4		C3		C2		C1		C0	
0		0	0x0		0x0		0x0		0x0		0x0		0x0		0x0		0x0		0x0		0x0		0x0		0x0	
r		rw	rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw	

31: 25 Reserved

24: 0 Bit *x* of this register controls whether the event corresponding to bit *x* of ACOMP.STAT0 can generate an interrupt or not. When bit *x* is 0, that event will not generate interrupts.

*Table 140.*0x81008008 - ACOMP.IMASK1 - ACOMP12-19 Interrupt Mask register

31	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED		P	C19		C18		C17		C16		C15		C14		C13		C12	
0		0	0x0		0x0		0x0		0x0		0x0		0x0		0x0		0x0	
r		r	r		r		r		r		r		r		r		r	

31: 17 Reserved

16: 0 Bit *x* of this register controls whether the event corresponding to bit *x* of ACOMP.STAT1 can generate an interrupt or not. When bit *x* is 0, that event will not generate interrupts.

16.3.4 ACOMP Interrupt Polarity Registers

*Table 141.*0x8100700C - ACOMP.IEDGE0 - ACOMP0-11 Interrupt Polarity register

31 25	24	23 22	21 20	19 18	17 16	15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
RESERVED	P	C11	C10	C9	C8	C7	C6	C5	C4	C3	C2	C1	C0
0	1	0x3	0x3	0x3	0x3	0x3	0x3	0x3	0x3	0x3	0x3	0x3	0x3
r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

31: 25 Reserved

24: 0 Bit *x* of this registers controls the polarity of interrupt generation from bit *x* of the ACOMP.STAT0 register.
0 => falling edge: interrupt generated when bit *x* changes from 1 to 0
1 => rising edge: interrupt generated when bit *x* changes from 0 to 1

*Table 142.*0x8100800C - ACOMP.IEDGE1 - ACOMP12-19 Interrupt Polarity register

31 17	16	15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
RESERVED	P	C19	C18	C17	C16	C15	C14	C13	C12
0	0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
r	r	r	r	r	r	r	r	r	r

31: 17 Reserved

16: 0 Bit *x* of this registers controls whether interrupt generation from bit *x* of the ACOMP.STAT1 register occurs on rising (1) or falling edges (0) of the status bit. See also ACOMP.IEDGE0.

16.3.5 ACOMP Configuration Registers

*Table 143.*0x810070010-0x81007003C, 0x810080010-81007002C ACOMP.CFG*n* - ACOMP *n* Configuration register

31 12	11 9	8	7	6	5	4	3 2	1 0
RESERVED	REFSEL	ARM	EN	S	R	SSEL	ACLEN	OSEL
0	0	0	0	0	0	0	0	0
r	rw	rw	rw	rw	rw	rw	rw	rw

31: 12 Reserved

11: 9 Comparator reference voltage selection (REFSEL):
000b - 0V (Ground)
001b - 0.125V
010b - 0.25V
011b - 0.5V
100b - 1.0V
101b - 1.5V
110b - 2.0V
111b - GPIO

8 Arm SR latch (ARM) - When 0, the S and R bits are ignored.

7 Comparator power enable (EN) - 0 to power down, 1 to enable. After setting EN to 1, up to 10 µs of delay is needed until analog performance is guaranteed.

6 Set SR latch (S) - Set to 1 to clock logic 1 into SR latch. Ignored if ARM=0 or R=1.

5 Clear SR Latch (R) - Set to 1 to clock logic 0 into SR latch. Ignored if ARM=0.

4 SYNCH signal source selection (SSEL)
0 => Asynchronously pass through the input selected by OSEL.
1 => SYNCH output will equal Q output.

3: 2 Enable asynchronous SR Latch clear (ACLEN) - When 0b11 the comparator output selected by OSEL will be used as the asynchronously clear input of the SR Latch. Other values of ACLEN disable asynchronous clear.

1: 0 Comparator output mode selection (OSEL)
00b - Disturbance tolerant comparator output (non-inverted)
01b - Disturbance tolerant comparator output (digitally inverted)
10b - Fast comparator output (non-inverted)
11b - Fast comparator output (digitally inverted)

16.3.6 ACOMP Power configuration registers

Table 144. ACOMP.PWR0 - ACOMP0-11 Power configuration register

31	2	1	0
RESERVED		REFEN	BIASEN
0		0	0
r		rw	rw

31: 2 Reserved

1 Reference voltage buffer power enable for ACOMP0-11 (REFEN) - 0 to power down, 1 to enable. Bit must be set to 1 to use non-zero internally generated reference voltage for any of ACOMP0-11. After setting REFEN to 1, up to 10 µs of delay is needed until analog performance is guaranteed.

0 Bias enable for ACOMP0-11 (BIASEN) - 0 to power down, 1 to enable. Bit must be set to 1 for any use of ACOMP0-11. After setting BIASEN to 1, up to 10 µs of delay is needed until analog performance is guaranteed.

Table 145. ACOMP.PWR1 - ACOMP12-19 Power configuration register

31	2	1	0
RESERVED		REFEN	BIASEN
0		0	0
r		rw	rw

31: 2 Reserved

1 Reference voltage buffer power enable for ACOMP12-19 (REFEN) - 0 to power down, 1 to enable. Bit must be set to 1 to use non-zero internally generated reference voltage for any of ACOMP12-19. After setting REFEN to 1, up to 10 µs of delay is needed until analog performance is guaranteed.

0 Bias enable for ACOMP12-19 (BIASEN) - 0 to power down, 1 to enable. Bit must be set to 1 for any use of ACOMP12-19. After setting BIASEN to 1, up to 10 µs of delay is needed until analog performance is guaranteed.

17 LEON3/FT - High-performance SPARC V8 32-bit Processor

17.1 Overview

LEON3 is a 32-bit processor core conforming to the IEEE-1754 (SPARC V8) architecture. It is designed for embedded applications, combining high performance with low complexity and low power consumption.

The LEON3 core has the following main features: 7-stage pipeline with Harvard architecture, hardware multiplier and divider, on-chip debug support and multi-processor extensions.

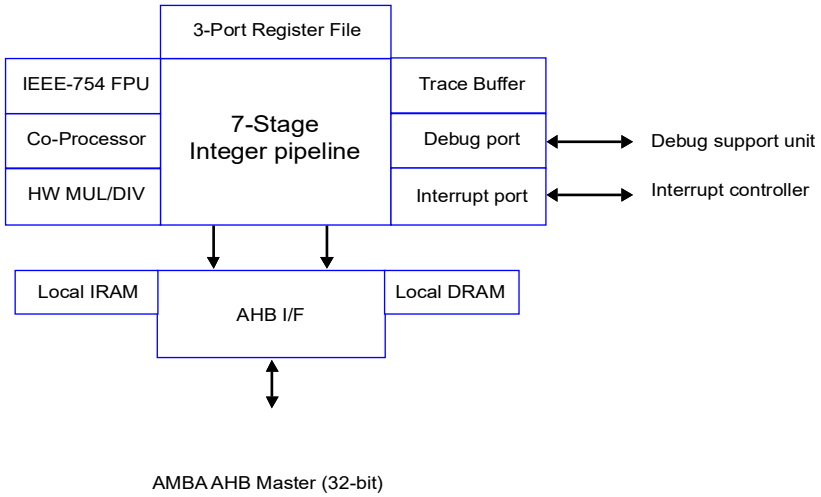


Figure 32. LEON3 processor core block diagram

17.1.1 Integer unit

The LEON3 integer unit implements the full SPARC V8 manual, including hardware multiply and divide instructions. The number of register windows is 31. The pipeline consists of 7 stages with a separate local instruction and data interface (Harvard architecture).

17.1.2 Floating-point unit and co-processor

The LEON3 integer unit provides interfaces for a floating-point unit (FPU). The floating-point processors execute in parallel with the integer unit, and does not block the operation unless a data or resource dependency exists.

17.1.3 On-chip debug support

The LEON3 pipeline includes functionality to allow non-intrusive debugging on target hardware. To aid software debugging, 4 watchpoint registers can be enabled. Each register can cause a breakpoint trap on an arbitrary instruction or data address range. When the debug support unit is attached, the watchpoints can be used to enter debug mode. Through a debug support interface, full access to all processor registers is provided. The debug interfaces also allows single stepping, instruction tracing and hardware breakpoint/watchpoint control. An internal trace buffer can monitor and store executed instructions, which can later be read out via the debug interface.

17.1.4 Interrupt interface

LEON3 supports the SPARC V8 interrupt model with a total of 15 asynchronous interrupts. The interrupt interface provides functionality to both generate and acknowledge interrupts.

17.1.5 AMBA interface

LEON3 implements an AMBA AHB master to load and store data. The interface is compliant with the AMBA-2.0 standard.

17.1.6 Power-down mode

The LEON3 processor core implements a power-down mode, which halts the pipeline until the next interrupt. The processor also supports clock gating during the power down period. A small part of the CPU is always awake and needs to run during power-down to check for wake-up conditions.

17.2 LEON3 integer unit

17.2.1 Overview

The LEON3 integer unit implements the integer part of the SPARC V8 instruction set. The implementation is focused on high performance and low complexity. The LEON3 integer unit has the following main features:

- 7-stage instruction pipeline
- 31 register windows
- Hardware multiplier
- Radix-2 divider (non-restoring)
- Static branch prediction
- Single-vector trapping for reduced code size

Figure 33 shows a block diagram of the integer unit.

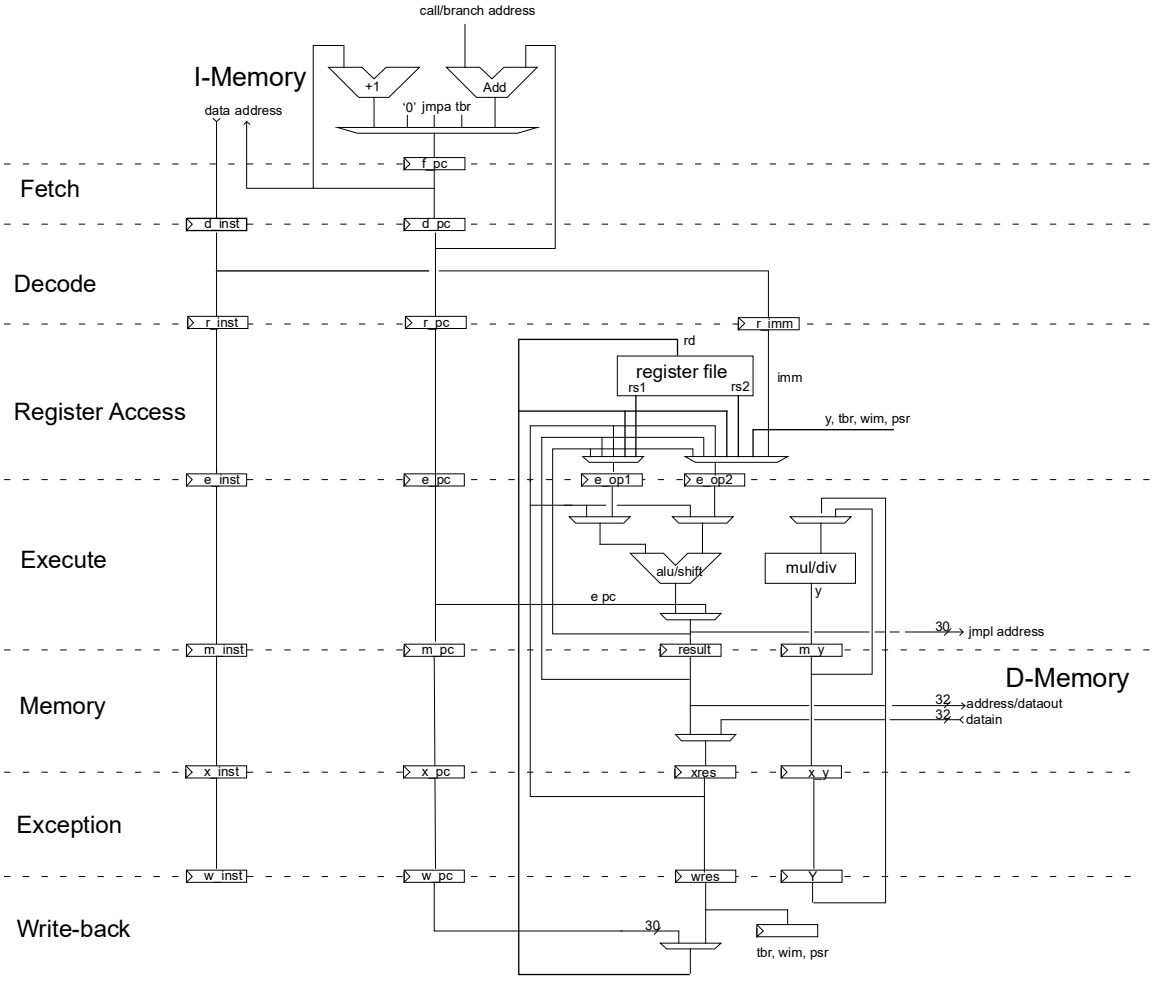


Figure 33. LEON3 integer unit datapath diagram

17.2.2 Instruction pipeline

The LEON3 integer unit uses a single instruction issue pipeline with 7 stages:

1. FE (Instruction Fetch): The instruction is fetched from the local instruction memory or external memory located on the AMBA bus. The instruction is valid at the end of this stage and is latched inside the IU.
2. DE (Decode): The instruction is decoded and the CALL and Branch target addresses are generated.
3. RA (Register access): Operands are read from the register file or from internal data bypasses.
4. EX (Execute): ALU, logical, and shift operations are performed. For memory operations (e.g., LD) and for JMPL/RETT, the address is generated.
5. ME (Memory): Data memory is read or written at this time.
6. XC (Exception) Traps and interrupts are resolved. For internal memory reads, the data is aligned as appropriate.
7. WR (Write): The result of any ALU, logical, shift, or internal memory operations are written back to the register file.

Table 146 lists the cycles per instruction (assuming local instruction and data memory are used):

*Table 146.*Instruction timing

Instruction	Cycles
JMPL	3[1]
JMPL,RETT pair	4
Double load	2
Single store	2
Double store	3
SMUL/UMUL	4
SDIV/UDIV	35
Taken Trap	5
Atomic load/store	3
All other instructions	**1**

[1] Assuming instruction in JMPL delay slot takes one cycle. Additional cycles spent in the delay slot will reduce the effective time of the JMPL to 2 or 1.

A number of conditions can extend an instruction's duration in the pipeline:

Branch interlock: When a conditional branch or trap is performed 1-2 cycles after an instruction which modifies the condition codes, 1-2 cycles of delay is added to allow the condition to be computed. If static branch prediction is enabled, this extra delay is incurred only if the branch is not taken.

Load delay: When using data resulting on a load shortly after the load, the instruction will be delayed to satisfy the pipeline's load delay. The processor pipeline is configured for two cycles of load delay.

Hold cycles: When blocking on the store buffer, the pipeline will be held still until the data is ready, effectively extending the execution time of the instruction causing the miss by the corresponding number of cycles. Note that since the whole pipeline is held still, hold cycles will not mask load delay or interlock delays.

FPU/Coprocessor: The floating-point unit or coprocessor may need to hold the pipeline or extend a specific instruction. When this is done is specific to the FP/CP unit.

17.2.3 SPARC Implementor's ID

Frontgrade Gaisler is assigned number 15 (0xF) as SPARC implementor's identification. This value is hard-coded into bits 31:28 in the %psr register. The version number for LEON3 is 3, which is hard-coded in to bits 27:24 of the %psr.

17.2.4 Divide instructions

Full support for SPARC V8 divide instructions is provided (SDIV, UDIV, SDIVCC & UDIVCC). The divide instructions perform a 64-by-32 bit divide and produce a 32-bit result. Rounding and overflow detection is performed as defined in the SPARC V8 manual.

17.2.5 Multiply instructions

The LEON processor supports the SPARC integer multiply instructions UMUL, SMUL UMULCC and SMULCC. These instructions perform a 32x32-bit integer multiply, producing a 64-bit result. SMUL and SMULCC performs signed multiply while UMUL and UMULCC performs unsigned multiply. UMULCC and SMULCC also set the condition codes to reflect the result. The multiply instructions are performed using a 16x16 hardware multiplier which is iterated four times.

17.2.6 Compare and Swap instruction (CASA)

LEON3 implements the SPARC V9 Compare and Swap Alternative (CASA) instruction. The CASA operates as described in the SPARC V9 manual. The instruction is privileged but setting ASI = 0xA (user data) will allow it to be used in user mode.

17.2.7 Hardware breakpoints

The integer unit is configured to include 4 hardware breakpoints. Each breakpoint consists of a pair of ancillary state registers (see section 17.6.5). Any binary aligned address range can be watched for instruction or data access, and on a breakpoint hit, trap 0x0B is generated.

17.2.8 Instruction trace buffer

The instruction trace buffer consists of a circular buffer that stores executed instructions. This is enabled and accessed only through the processor's debug port via the Debug Support Unit. When enabled, the following information is stored in real time, without affecting performance:

- Instruction address and opcode
- Instruction result
- Load/store data and address
- Trap information
- 30-bit time tag

17.2.9 Processor configuration register

The ancillary state register 17 (%asr17) provides information on how various configuration options. This can be used to enhance the performance of software. See section 17.6.2 for layout.

17.2.10 Exceptions

LEON3 adheres to the general SPARC trap model. The table below shows the implemented traps and their individual priority. When PSR (processor status register) bit ET=0, an exception trap causes the processor to halt execution and enter error mode, and the external error signal will then be asserted.

Table 147. Trap allocation and priority

Trap	TT	Pri	Description	Class
reset	0x00	1	Power-on reset	Interrupting
data_store_error	0x2b	2	write buffer error during data store	Interrupting
instruction_access_exception	0x01	3	Error or MMU page fault during instruction fetch	Precise
privileged_instruction	0x03	4	Execution of privileged instruction in user mode	Precise
illegal_instruction	0x02	5	UNIMP or other un-implemented instruction	Precise
fp_disabled	0x04	6	FP instruction while FPU disabled	Precise
cp_disabled	0x24	6	CP instruction while Co-processor disabled	Precise
watchpoint_detected	0x0B	7	Hardware breakpoint match	Precise
window_overflow	0x05	8	SAVE into invalid window	Precise
window_underflow	0x06	8	RESTORE into invalid window	Precise
r_register_access_error	0x20	9	register file EDAC error (LEON3FT only)	Interrupting
mem_address_not_aligned	0x07	10	Memory access to un-aligned address	Precise
fp_exception	0x08	11	FPU exception	Deferred
cp_exception	0x28	11	Co-processor exception	Deferred
data_access_exception	0x09	13	Access error during data load, MMU page fault	Precise
tag_overflow	0x0A	14	Tagged arithmetic overflow	Precise
division_by_zero	0x2A	15	Divide by zero	Precise
trap_instruction	0x80 - 0xFF	16	Software trap instruction (TA)	Precise
interrupt_level_15	0x1F	17	Asynchronous interrupt 15	Interrupting
interrupt_level_14	0x1E	18	Asynchronous interrupt 14	Interrupting
interrupt_level_13	0x1D	19	Asynchronous interrupt 13	Interrupting
interrupt_level_12	0x1C	20	Asynchronous interrupt 12	Interrupting
interrupt_level_11	0x1B	21	Asynchronous interrupt 11	Interrupting
interrupt_level_10	0x1A	22	Asynchronous interrupt 10	Interrupting
interrupt_level_9	0x19	23	Asynchronous interrupt 9	Interrupting
interrupt_level_8	0x18	24	Asynchronous interrupt 8	Interrupting
interrupt_level_7	0x17	25	Asynchronous interrupt 7	Interrupting
interrupt_level_6	0x16	26	Asynchronous interrupt 6	Interrupting
interrupt_level_5	0x15	27	Asynchronous interrupt 5	Interrupting
interrupt_level_4	0x14	28	Asynchronous interrupt 4	Interrupting
interrupt_level_3	0x13	29	Asynchronous interrupt 3	Interrupting
interrupt_level_2	0x12	30	Asynchronous interrupt 2	Interrupting
interrupt_level_1	0x11	31	Asynchronous interrupt 1	Interrupting

The prioritization follows the SPARC V8 manual except for a minor difference for r_register_access_error, which has lower priority than window_over/underflow because the window condition is detected before the register file is accessed.

The data_store_error is delivered as a deferred exception but is non-resumable and therefore classed as interrupting. Likewise, r_register_access_error is delivered as a precise trap but since it is non-resumable it is classed as an interrupting trap.

17.2.11 Single vector trapping (SVT)

Single-vector trapping (SVT) is an SPARC V8e option to reduce code size for embedded applications. When enabled, any taken trap will always jump to the reset trap handler (%tbr.tba + 0). The trap type will be indicated in %tbr.tt, and must be decoded by the shared trap handler. SVT is enabled by setting bit 13 in %asr17.

17.2.12 Address space identifiers (ASI)

In addition to the address, a SPARC processor also generates an 8-bit address space identifier (ASI), providing up to 256 separate, 32-bit address spaces. During normal operation, the LEON3 processor accesses instructions and data using ASI 0x8 - 0xB as defined in the SPARC manual. Using the LDA/STA instructions, alternative address spaces can be accessed. The different available ASIs are described in section 17.6.

17.2.13 Partial WRPSR

The processor has support for partial WRPSR. Partial write %PSR (WRPSR) is a SPARC V8e option that allows WRPSR instructions to only affect the %PSR.ET field.

17.2.14 Alternative window pointer

Alternative window pointer (AWP) is a SPARC V8e option intended to reduce interrupt latency by allowing code that manipulates the current window pointer, mainly window over and underflow handlers and context switching code, to run with traps enabled.

Two bits are added to the PSR register, AW (alternative window) and PAW (previous alternative window). Also an AWP (alternative window pointer) field is added in an ASR register.

When the AW bit is set, the current register window used for reading/writing non-global registers is taken from the AWP register field instead of the normal CWP register field, and SAVE and RESTORE operations modify the AWP field instead of the CWP. SAVE and RESTORE can do not trigger the window over/underflow traps while AW is set.

When both AW and PAW are zero, the AWP field is kept equal to the CWP field.

When a trap occurs, the value of AW is copied into the PAW field, and AW is cleared. When returning from a trap using the RETT instruction, the PAW field is copied back into AW. The RETT will not trigger the window underflow trap if PAW is set regardless of if CWP or AWP point to an invalid window.

17.2.15 Register file partitioning

Register file partitioning is an optional extension to allow a subrange of the register windows to be used as if it was the whole register file. The selected subset is connected in a ring so that the outs of the lowest register window is aliased to the ins of the highest register window in the range. Other register windows outside this range are not accessible and will be kept at their old values while the partitioning is enabled.

The partitioning is activated by setting the STWIN and CWPMAX fields of the %asr20 register. This selects the subset of windows between STWIN and STWIN+CWPMAX so that they map to CWP values 0 to CWPMAX. STWIN and CWPMAX must be set so they map to a valid range, CWPMAX+STWIN must not exceed the highest possible CWP value supported in the normal case. Also, for correct operation, CWP must be set to a value between 0 and CWPMAX before accessing any non-global register.

Writing CWPMAX to (otherwise illegal value) 0 in %asr20 will result in writing only AWP and keeping the values of STWIN and CWPMAX.

A special write-only bit in the %asr20 register can be used to write CWP in the PSR at the same time as writing the STWIN,CWPMAX,AWP fields, this is intended to allow switching between two register file partitions without disabling interrupts.

The WIM register is not managed by the partitioning logic, therefore the lowest bits of the WIM will map to the partitioned windows. The highest bits of the WIM will be masked to 0 on read to simulate a smaller register file, however these bits are still writable.

17.2.16 Power-down

The processor can enter a power-down mode to minimize power consumption during idle periods. The power-down mode is entered by performing a WRASR instruction to %asr19:

```
wr %g0, %asr19
```

During power-down, the pipeline is halted until the next interrupt occurs. Signals inside the processor pipeline are then static, reducing power consumption from dynamic switching.

Note: %asr19 must always be written with the data value zero to ensure compatibility with future extensions.

Note: This instruction must be performed in supervisor mode with interrupts enabled.

When resuming from power-down, the pipeline will be re-filled from the point of power-down and the first instruction following the WRASR instruction will be executed prior to taking the interrupt trap. Up to six instructions after the WRASR instruction will be fetched prior to fetching the trap handler.

17.2.17 Processor reset operation

The processor is reset by asserting the RESET input for at least 4 clock cycles. The following table indicates the reset values of a subset of the registers which are affected by the reset..

*Table 148.*Processor reset values

Register	Reset value
Trap Base Register	Trap Base Address field reset (value 0)
PC (program counter)	0x0
nPC (next program counter)	0x4
PSR (processor status register)	ET=0, S=1

By default, the execution will start from address 0 and is taken from the register processor boot address register in the interrupt controller. This allows processor to be dynamically restarted and the reset address to be changed dynamically and can e.g. when new software has been remotely uploaded and processor should restart.

17.2.18 LEON-REX extension

The processor supports the LEON-REX addition to the SPARC instruction set, allowing a more compact code representation than the regular SPARC machine code, see reference document [LEON-REX].

Detection whether support is present can be done by checking the REXV field in the asr17 register (see section 17.6.2). The REX support can be set to enabled, illegal or transparent mode via the REXEN/REXILL bits in the asr17 register. Enabled: REXEN=0, REXILL=0. Illegal: REXEN=0, REXILL=1. Transparent: REXEN=1, REXILL=0. After reset the default setting is illegal (TBC, REXEN=1, REXILL=1) so any LEON-REX code will cause an illegal instruction trap.

17.2.19 Constant interrupt delay

The LEON3FT is enhanced with an interrupt zero jitter feature. When the interrupt zero jitter feature is enabled all sources of interrupt jitter introduced by the hardware can be eliminated. The latency is controlled via a 12 bit counter register which also determines the interrupt latency. If the 12 bit counter is set in the range 0 to 4, the LEON3FT will start the to process the interrupt request as soon as possible. If the counter is set to a specific value depending on the timing of the memory system, then it can enable the zero jitter behavior to force the interrupt latency to higher number of cycles, but it is guaranteed to have zero jitter.

The processor interrupt delay bit fields are found and in the ASI2 register. Example of setting the interrupt delay to 10 clock cycles:

```
asm volatile (" sta %0, [%1] 2" : : "r"(10), "r"(4) : "memory");
```

17.3 Local instruction and data RAM

Local instruction and data ram is attached to the processor. The local instruction ram is 64 KiB and the local data ram is 64 KiB. Accesses performed to the local RAMs will not appear on the AHB bus. The address for the instruction ram is 0x31000000, and for the data ram 0x30000000.

The local instruction RAM is intended for executing instructions and will serve instructions without any wait states. Initializing the local instruction RAM is done from software via stores or remotely via the local instruction memory AHB interface on the DMA bus.

The local data RAM will serve data accesses of any size without adding wait states. The local data RAM can be accessed via AHB from any AMBA master with access to the DMA bus.

17.4 GRFPU-Lite floating-point unit

Frontgrade Gaisler's GRFPU-Lite is connected with the LEON3 pipeline. The GRFPU-Lite is a smaller version of the GRFPU, suitable for implementations with limited logic resources. The GRFPU-Lite is not pipelined and thus executes only one instruction at a time. To improve performance, the FPU controller (GRLFPC) allows GRFPU-Lite to execute in parallel with the processor pipeline as long as no new FPU instructions are pending. Below is a table of worst-case throughput of the GRFPU-Lite:

*Table 149.*GRFPU-Lite worst-case instruction timing with GRLFPC

Instruction	Throughput	Latency
FADDS, FADDD, FSUBS, FSUBD,FMULS, FMULD, FSMULD, FITOS, FITOD, FSTOI, FDTOI, FSTOD, FDTOS, FCMPS, FCMPD, FCMPES. FCMPED	8	8
FDIVS	31	31
FDIVD	57	57
FSQRTS	46	46
FSQRTD	65	65

The GRLFPC controller implements the SPARC deferred trap model, but the FPU trap queue (FQ) can contain only one queued instructions when an FPU exception is taken.When the GRFPU-Lite is enabled in the model, the version field in %fsr has the value of 3.

17.5 AMBA interface

17.5.1 Overview

The LEON3 processor uses one AHB master interface for all data and instruction accesses. Instructions and data are fetched with single READ cycles. Store data is performed using single accesses or a two-beat incremental burst in case of 64-bit store.

The HPROT signals of the AHB bus are driven to indicate if the accesses is instruction or data, and if it is a user or supervisor access.

*Table 150.*HPROT values

Type of access	User/Super	HPROT
Instruction	User	1100
Instruction	Super	1110
Data	User	1101
Data	Super	1111

In case of atomic accesses, a locked access will be made on the AMBA bus to guarantee atomicity as seen from other masters on the bus.

17.5.2 Error handling

An AHB ERROR response received while fetching instructions will normally cause an instruction access exception (tt=0x1).

An AHB ERROR response while fetching data will normally trigger a data_access_exception trap (tt=0x9).

17.6 Configuration registers

17.6.1 PSR, WIM, TBR registers

The %psr, %wim, %tbr registers are implemented as required by the SPARC V8 manual.

*Table 151.*LEON3 Processor state register (%psr)

31 28	27 24	23 20	19 16
IMPL	VER	ICC	RESERVED
0xF	0x3	0	0
r	r	r	r

15 14	13	12	11 8	7	6	5	4 0
RESERVED	EC	EF	PIL	S	PS	ET	CWP
0	0	0	0	1	1	0	0
r	r	rw	rw	rw	rw	rw	rw

31:28	Implementation ID (IMPL), read-only hardwired to “1111” (15)
27:24	Implementation version (VER), read-only hardwired to “0011” (3) for LEON3.
23:20	Integer condition codes (ICC), see sparcv8 for details
19:14	Reserved
13	Enable coprocessor (EC), always set to ’0’ to indicate no coprocessor available in microcontroller
12	Enable floating-point (EF)
11:8	Processor interrupt level (PIL) - controls the lowest IRQ number that can generate a trap
7	Supervisor (S)
6	Previous supervisor (PS), see sparcv8 for details
5	Enable traps (ET)
4:0	Current window pointer

*Table 152.*LEON3 Window invalid mask (%wim)

31	30 0
R	WIM
0	NR
r	rw

*Table 153.*LEON3 Trap base address register (%tbr)

31 12	11 4	3 0
TBA	TT	R
*	0	0
rw	r	r

31:12	Trap base address (TBA) - Top 20 bits used for trap table address
11:4	Trap type (TT) - Last taken trap type. Read only.
3:0	Always zero, read only

17.6.2 ASR17, LEON3 configuration register

The ancillary state register 17 (%asr17) provides information on current LEON3FT configuration. This can be used to enhance the performance of software. There are also a few bits that are writable to configure certain aspects of the processor.

*Table 154.*LEON3 configuration register (%asr17)

31 28	27	26	25	24 23	22	21	20 18	17	16
INDEX	R	NOTAG	R	REXV	REXEN	REXILL	RESERVED	R	R
0	0	1	1	01b	1	1	0	0	0
r	r	r	r	r	rw	rw	r	r	r

15	14	13	12	11 10	9	8	7 5	4 0
R	DW	SV	LD	FPU	M	V8	NWP	NWIN
0	0	0	1	3	0	1	4	30
r	rw	rw	r	r	r	r	r	r

31:28	Processor index (INDEX) -Processor index is set to zero.
27	Reserved and not used
26	Tagged arithmetic (NOTAG) - Then the processor supports tagged arithmetic.and compare-and-swap (CASA) instruction.
27	Reserved and not used
24:23	REX version (REXV) - REX version
22	REX enable (REXEN) - Set to 0 to enable REX, 1 to disable. Writable. Reset value 1.
21	REX illegal (REXILL) - Set to 0 to enable REX, 1 to generate illegal instruction exceptions. Writable. Reset value 1. See section 17.2.18 for how this bit together with REXEN sets the REX mode.
20:18	Reserved for future implementations
17	Reserved for future implementations
16:15	Reserved for future implementations
14	Disable write error trap (DWT). When set, a write error trap (tt = 0x2b) will be ignored. Set to zero after reset.
13	Single-vector trapping (SVT) enable. If set, will enable single-vector trapping. Set to zero after reset.
12	Load delay (LDDEL) - Indicates 2-cycle load delay is used.
11:10	FPU Option (FPU) - Indicates system has a GRFPU-Lite
9	MAC instruction (M) - Set to zero to indicate multiply-accumulate (MAC) instruction is NOT available
8	MUL/DIV instructions available (V8) - Indicates SPARC V8 multiply and divide instructions are available
7:5	Hardward watchpoints (NWP) - Number of watchpoints available is 4
4:0	Register windows (NWIM) - Number of implemented registers windows corresponds to NWIN+1 i.e. 31 for GR716B.

17.6.3 ASR20, Alternative window register

This register allows access to the alternative window pointer.

*Table 155.*LEON3 alternative window register (%asr20)

31	26	25	21	20	16
RESERVED		STWIN		CWPMAX	
0		0		30	
r		rw		rw	

15	5	4	0
RESERVED	WCWP	AWP	
0	-	*	
r	w	rw	

31:26	Reserved for future implementations
25:21	Starting window (STWIN) - Starting window of partition.
20:16	Maximum value of current window pointer (CWPMAX) - Partition size minus 1. Reset value is number of windows minus 1, which with STWIN=0 maps whole register file into partition. If this field is written with value 0, STWIN and CWPMAX fields are unmodified.
15:5	Reserved for future implementations
5	Write CWP - If written with 1, then the CWP field in PSR will simultaneously be written with the value written to AWP.
4:0	Alternative Window Pointer (AWP). Continuously updated with the value of CWP when the alternative window feature is disabled.

17.6.4 ASR22-23 - Up-counter

The ancillary state registers 22 and 23 (%asr22-23) contain an internal up-counter that can be read by software without causing any access on the on-chip AMBA bus. The number of available bits in the counter is 32 bits and is the same as the number of counter bits in the DSU time tag counter. %ASR23 contains the least significant part of the counter value and %ASR22 contains the most significant part.

The time tag value accessible in these registers is the same time tag value used for the system's trace buffers and for all processors connected to the same debug support unit. The time tag counter will increment when any of the trace buffers is enabled, or when the time tag counter is forced to be enabled via the DSU register interface, or when any processor has its %ASR22 Disable Up-counter (DUCNT) field set to zero.

The up-counter value will increment even if all processors have entered power-down mode.

Table 156. LEON3 up-counter MSbs (%ASR22)

31	30 … 0
DUCNT	Not Used

31	Disable Up-counter (DUCNT) - Disable upcounter. When set to '1' the up-counter may be disabled. When cleared, the counter will increment each processor clock cycle. Default (reset) value is '1'.
30:0	Reserved and not used

Table 157. LEON3 up-counter LSbs (%ASR23)

31 … 0
UPCNT(31:0)

31:0	Counter value (UPCNT(31:0)) - Least significant bits of internal up-counter. Read-only.

17.6.5 ASR24-31, Hardware watchpoint/breakpoint registers

Each breakpoint consists of a pair of ancillary state registers (%asr24/25, %asr26/27, %asr28/29 and %asr30/31) registers; one with the break address and one with a mask:

%asr24, %asr26
%asr28, %asr30

31	2	1	0
WADDR[31:2]			IF
NR		0	0
rw		r	rw

%asr25, %asr27
%asr29, %asr31

31	2		0
WMASK[31:2]		DL	DS
NR		0	0
rw		rw	rw

Figure 34. Watch-point registers

WADDR - Address to compare against

WMASK - Bit mask controlling which bits to check (1) or ignore (0) for match

IF - break on instruction fetch from the specified address/mask combination

DL - break on data load from the specified address/mask combination

DS - break on data store to the specified address/mask combination

Note: Setting IF=DL=DS=0 disables the breakpoint

When there is a hardware watchpoint match and DL or DS is set then trap 0x0B will be generated. Hardware watchpoints can be used with or without the LEON3 debug support unit (DSU) enabled.

17.6.6 Register protection control register

ASR register 16 (%asr16) is used to control the IU register file SEU protection. It is possible to disable the SEU protection by setting the IDI bits, and to inject errors using the ITE bits. Corrected errors in the register file are counted, and available in ICNT fields. The counters saturate at their maximum value (7), and should be reset by software after read-out. The FPU register file is rad hard by design.

*Table 158.*LEON3FT Register protection control register (%asr16)

31 30	29 27	26 20	19 18	17	16
RESERVED	RESERVED	RESERVED	EIUFT	R	R
0	0	0	1	0	0
r	r	r	r	r	r

15 14	13 11	10 3	2	1	0
IUFT	ICNT	RFTB[7:0]	DP	ITE	IDI
1	0	0	0	0	0
r	rw	rw	rw	rw	rw

31:30	Reserved
29:27	Reserved
26: 20	Reserved
19: 18	Extended IU FT ID - Top bits of IUFT field to indicate FT values higher than 3
17	Reserved
16	Reserved
15:14	IU FT ID - SEU protection is available for IU
13:11	IU RF error counter - Number of detected errors in the IU register file.
10:3	RF Test bits (RFTB) - In test mode, these bits are xored with correct parity bits before written to the register file.
2	DP ram select (DP) - Only applicable if the IU register files consists of two dual-port rams. See table 159 below.
1	IU RF Test Enable - Enables register file test mode. Parity bits are xored with TB before written to the register file.
0	IU RF protection disable (IDI) - Disables IU RF parity protection when set.

*Table 159.*DP ram select usage

ITE/FTE	DP	Function
1	0	Write to IU register (%i, %l, %o, %g) will only write location of %rs2
1	1	Write to IU register (%i, %l, %o, %g) will only write location of %rs1
0	X	IU registers written nominally

17.7 Software considerations

17.7.1 Register file initialization on power up for LEON3FT

This section is only valid if internal boot ROM is bypassed.

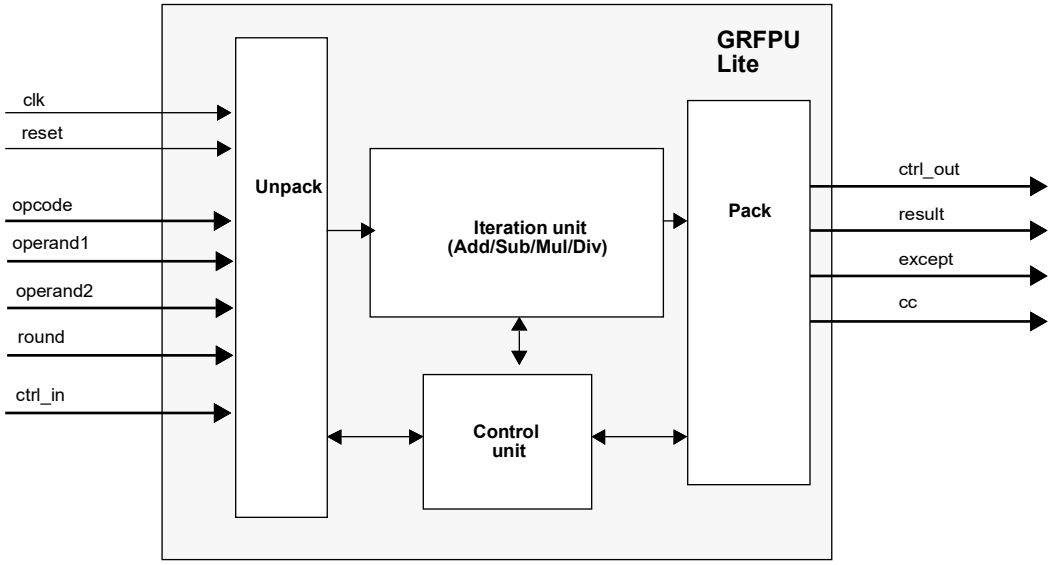
After power-on and internal boot ROM is bypassed, the check bits in the IU and FPU register files are not initialized. This means that access to an un-initialized (un-written) general-purpose register could cause a register access trap (tt = 0x20). Such behavior is considered as a software error, as the software should not read a register before it has been written. It is recommended that the boot code for the processor writes all registers in the IU and FPU register files before launching the main application. Initialization of IU and FPU register files is performed by the internal boot ROM before handover to application software.

18 IEEE-754 Floating-Point Unit

18.1 Overview

The floating-point unit implements floating-point operations as defined in IEEE Standard for Binary Floating-Point Arithmetic (IEEE-754) and SPARC V8 standard (IEEE-1754).

Supported formats are single and double precision floating-point numbers. The floating-point unit is not pipelined and executes one floating-point operation at a time.



18.2 Functional Description

18.2.1 Floating-point number formats

The floating-point unit handles floating-point numbers in single or double precision format as defined in IEEE-754 standard.

18.2.2 FP operations

The floating-point unit supports four types of floating-point operations: arithmetic, compare, convert and move. The operations, summarized in the table below, implement all FP instructions specified by the SPARC V8 instruction set except FSMULD and instructions with quadruple precision.

Table 160.:Floating-point operations

Operation	Op1	Op2	Result	Exceptions	Description
Arithmetic operations					
FADDS FADDD	SP DP	SP DP	SP DP	NV, OF, UF, NX	Addition
FSUBS FSUBD	SP DP	SP DP	SP DP	NV, OF, UF, NX	Subtraction
FMULS FMULD	SP DP	SP DP	SP DP	NV, OF, UF, NX NV, OF, UF, NX	Multiplication
FDIVS FDIVD	SP DP	SP DP	SP DP	NV, OF, UF, NX, DZ	Division
FSQRTS FSQRTD	- -	SP DP	SP DP	NV, NX	Square-root
Conversion operations					
FITOS FITOD	-	INT	SP DP	NX -	Integer to floating-point conversion
FSTOI FDTOI	-	SP DP	INT	NV, NX	Floating-point to integer conversion. The result is rounded in round-to-zero mode.
FSTOD FDTOS	-	SP DP	DP SP	NV NV, OF, UF, NX	Conversion between floating-point formats
Comparison operations					
FCMPS FCMPD	SP DP	SP DP	CC	NV	Floating-point compare. Invalid exception is generated if either operand is a signaling NaN.
FCMPES FCMPED	SP DP	SP DP	CC	NV	Floating point compare. Invalid exception is generated if either operand is a NaN (quiet or signaling).
Negate, Absolute value and Move					
FABSS	-	SP	SP	-	Absolute value.
FNEGS	-	SP	SP	-	Negate.
FMOVS		SP	SP	-	Move. Copies operand to result output.

SP - single precision floating-point number

DP - double precision floating-point number

INT - 32 bit integer

CC - condition codes

NV, OF, UF, NX - floating-point exceptions, see section 18.2.3

Below is a table of worst-case throughput of the floating point unit.

*Table 161.*Worst-case instruction timing

Instruction	Throughput	Latency
FADDS, FADDD, FSUBS, FSUBD,FMULS, FMULD, FITOS, FITOD, FSTOI, FDTOI, FSTOD, FDTOS, FCMPS, FCMPD, FCMPES. FCMPED	8	8
FDIVS	31	31
FDIVD	57	57
FSQRTS	46	46
FSQRTD	65	65

18.2.3 Exceptions

The floating-point unit detects all exceptions defined by the IEEE-754 standard. This includes detection of Invalid Operation (NV), Overflow (OF), Underflow (UF), Division-by-Zero (DZ) and Inexact (NX) exception conditions. Generation of special results such as NaNs and infinity is also supported.

18.2.4 Rounding

All four rounding modes defined in the IEEE-754 standard are supported: round-to-nearest, round-to-+inf, round-to--inf and round-to-zero.

19 UART Serial Interface

The GR716B comprises 6 separate UART units and 2 debug and remote access UART units. The 2 debug and remote access UART units also called AHBUART units are described in section 48. This chapter only describes the UART units also called APBUART. The main difference between the UART units described in this section and the debug UART units are the debug and remote access UART units capability to respond to external UART signaling without software support. The two debug and remote access UART units can also act as a master on the internal bus without software support. The UART units described in this section requires software support for all operations.

The APB UART units are located on APB bus in the address range from 0x80300000 to 0x80305FFF. See UART units connections in the next drawing. The figure shows memory locations and functions used for UART configuration and control.

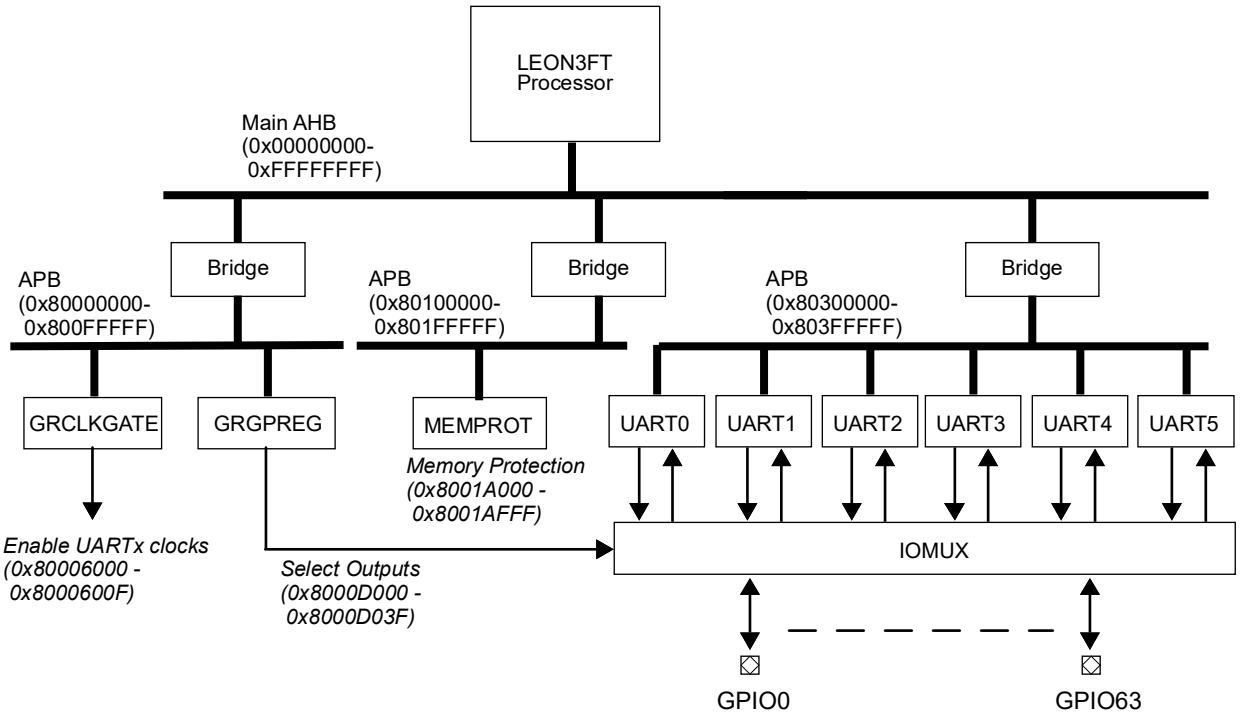


Figure 35. GR716B UART bus and pin connection

The primary clock gating unit **GRCLKGATE** described in section 27 is used to enable/disable individual UART units. The unit **GRCLKGATE** can also be used to perform reset of individual UART units. Software must enable clock and release reset described in section 27 before UART configuration and transmission can start.

External IO selection per UART unit is made in the system IO configuration register (**GRGPREG**) in the address range from 0x8000D000 to 0x8000D03F. See section 7.1 for further information.

Each **UARTx** unit controls its own external pins and has a unique AMBA address described in chapter 2.10. UART unit 0, 1, 2, 3, 4 and 5 have identical configuration and status registers. Configuration and status registers are described in section 19.7.

The system can be configured to protect and restrict access to individual UART unit in the **MEMPROT** unit. See section 47 for more information.

19.1 Overview

The universal asynchronous receiver-transmitter (UART) interface takes bytes and transmits the individual bit sequentially. The UART supports data frames with 8 data bits, one optional parity bit and one or two stop bits. To generate the bit-rate, each UART has a programmable 12-bit clock divider. To minimize the interrupts two 16-byte FIFOs are used for data transfer between the APB bus and UART, one FIFO for reception and one for transmission. Hardware flow-control is supported through the RTSN/CTSN hand-shake signals. Parity checking can be enabled per UART interface.

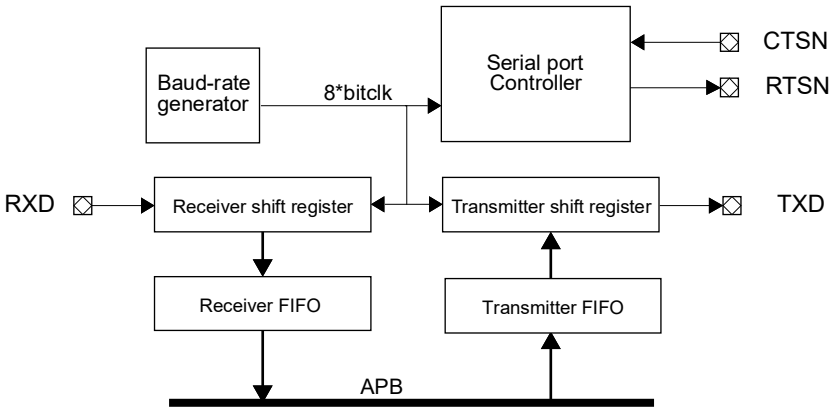


Figure 36. UART unit block diagram

19.2 Operation

19.2.1 Transmitter operation

The transmitter is enabled through the TE bit in the UART control register. Data that is to be transferred is stored in the 16-byte FIFO by writing to the data register. When ready to transmit, data is transferred from the transmitter FIFO to the transmitter shift register and converted to a serial stream on the transmitter serial output pin (TXD). It automatically sends a start bit followed by eight data bits, an optional parity bit, and one stop bit (figure 37). The least significant bit of the data is sent first. It is also possible to use two stop bits, this is configured via the control register.

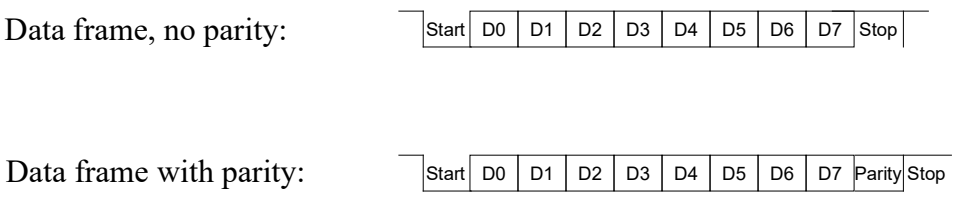


Figure 37. UART data frames

Following the transmission of the stop bit, if a new character is not available in the transmitter FIFO, the transmitter serial data output remains high and the transmitter shift register empty bit (TS) will be set in the UART status register. Transmission resumes and the TS is cleared when a new character is

loaded into the transmitter FIFO. When the FIFO is empty the TE bit is set in the status register. If the transmitter is disabled, it will immediately stop any active transmissions including the character currently being shifted out from the transmitter shift register. The transmitter FIFO may not be loaded when the transmitter is disabled or when the FIFO is full. If this is done, data might be overwritten and one or more frames are lost.

The TF status bit (not to be confused with the TF control bit) is set if the transmitter FIFO is currently full and the TH bit is set as long as the FIFO is *less* than half-full (less than half of entries in the FIFO contain data). The TF control bit enables FIFO interrupts when set. The status register also contains a counter (TCNT) showing the current number of data entries in the FIFO.

When flow control is enabled, the CTSN input must be low in order for the character to be transmitted. If it is deasserted in the middle of a transmission, the character in the shift register is transmitted and the transmitter serial output then remains inactive until CTSN is asserted again. If the CTSN is connected to a receivers RTSN, overrun can effectively be prevented.

19.2.2 Receiver operation

The receiver is enabled for data reception through the receiver enable (RE) bit in the UART control register. The receiver looks for a high to low transition of a start bit on the receiver serial data input pin. If a transition is detected, the state of the serial input is sampled a half bit clocks later. If the serial input is sampled high the start bit is invalid and the search for a valid start bit continues. If the serial input is still low, a valid start bit is assumed and the receiver continues to sample the serial input at one bit time intervals (at the theoretical centre of the bit) until the proper number of data bits and the parity bit have been assembled and one stop bit has been detected.

The receiver also has a 16-byte FIFO which is identical to the one in the transmitter.

During reception, the least significant bit is received first. The data is then transferred to the receiver FIFO and the data ready (DR) bit is set in the UART status register as soon as the FIFO contains at least one data frame. The parity, framing and overrun error bits are set at the received byte boundary, at the same time as the receiver ready bit is set. The data frame is not stored in the FIFO if an error is detected. Also, the new error status bits are or:ed with the old values before they are stored into the status register. Thus, they are not cleared until written to with zeros from the AMBA APB bus. If both the receiver FIFO and shift registers are full when a new start bit is detected, then the character held in the receiver shift register will be lost and the overrun bit will be set in the UART status register. A break received (BR) is indicated when a BREAK has been received, which is a framing error with all data received being zero.

RTSN will be negated (high) when a valid start bit is detected and the receiver FIFO is full. When the FIFO is read, the RTSN will automatically be reasserted again. This behavior applies regardless of the value of the FL bit in the UART control register.

The RF status bit (not to be confused with the RF control bit) is set when the receiver FIFO is full. The RH status bit is set when the receiver FIFO is half-full (at least half of the entries in the FIFO contain data frames). The RF control bit enables receiver FIFO interrupts when set. A RCNT field is also available showing the current number of data frames in the FIFO.

19.3 Baud-rate generation

Each UART contains a 12-bit down-counting scaler to generate the desired baud-rate. The scaler is clocked by the system clock and generates a UART tick each time it underflows. It is reloaded with the value of the UART scaler reload register after each underflow. The resulting UART tick frequency should be 8 times the desired baud-rate. One appropriate formula to calculate the scaler value for a desired baud rate, using integer division where the remainder is discarded, is:

*scaler value = (system_clock_frequency) / (baud_rate * 8 + 7).*

To calculate the exact required scaler value use:

*scaler value = (system_clock_frequency) / (baud_rate * 8) - 1*

19.4 Loop back mode

If the LB bit in the UART control register is set, the UART will be in loop back mode. In this mode, the transmitter output is internally connected to the receiver input and the RTSN is connected to the CTSN. It is then possible to perform loop back tests to verify operation of receiver, transmitter and associated software routines. In this mode, the outputs remain in the inactive state, in order to avoid sending out data.

19.5 FIFO debug mode

FIFO debug mode is entered by setting the debug mode bit in the control register. In this mode it is possible to read the transmitter FIFO and write the receiver FIFO through the FIFO debug register. The transmitter output is held inactive when in debug mode. A write to the receiver FIFO generates an interrupt if receiver interrupts are enabled.

19.6 Interrupt generation

Two different kinds of interrupts are available: normal interrupts and FIFO interrupts.

Normal interrupts from the transmitter are generated when transmitter interrupts are enabled (TI), the transmitter is enabled and the transmitter FIFO goes from containing data to being empty. For the receiver normal interrupts are generated when receiver interrupts are enabled (RI), the receiver is enabled and a character is received. The interrupt is generated if the character is correct and stored in the receive FIFO or if an error, such as parity; framing or overrun occurred.

Transmitter FIFO interrupts are generated when the transmitter FIFO interrupts are enabled (TF), transmissions are enabled (TE) and the UART is less than half-full (that is, whenever the TH status bit is set). This is a level interrupt and the interrupt signal is continuously driven high as long as the condition prevails. Receiver FIFO interrupts are generated when receiver FIFO interrupts are enabled (RF), the receiver is enabled and the FIFO is half-full. The interrupt signal is continuously driven high as long as the receiver FIFO is half-full (at least half of the entries contain data frames).

Note that the processor acknowledges and clears the corresponding interrupt pending register but for FIFO interrupts the interrupt signal from the UART is continuously driven high, resulting in a new pending interrupt immediately being set in the interrupt controller. If FIFO interrupts are used for controlling FIFO handling, an interrupt handler need to check that there is room in the transmit FIFO before writing and that characters are available in the receive FIFO before reading.

To reduce interrupt occurrence a delayed receiver interrupt is available. It is enabled using the delayed interrupt enable (DI) bit. When enabled a timer is started each time a character is received and an interrupt is only generated if another character has not been received within 4 character + 4 bit times. If receiver FIFO interrupts are enabled a pending character interrupt will be cleared when the FIFO interrupt is active since the character causing the pending irq state is already in the FIFO and is noticed by the driver through the FIFO interrupt. In order to not take one additional interrupt, software should clear the corresponding pending bit after the FIFO has been emptied.

There is also a separate interrupt for break characters. When enabled an interrupt will always be generated immediately when a break character is received even when delayed receiver interrupts are enabled. When break interrupts are disabled no interrupt will be generated for break characters when delayed interrupts are enabled.

When delayed interrupts are disabled the behavior is the same for the break interrupt bit except that an interrupt will be generated for break characters if receiver interrupt enable is set even if break interrupt is disabled.

An interrupt can also be enabled for the transmitter shift register. When enabled the core will generate an interrupt each time the shift register goes from a non-empty to an empty state.

19.7 Registers

The core is controlled through registers mapped into APB address space.

*Table 162.*UART registers

APB address offset	Register
0x80300000	UART0 Data register (UART0.DATA)
0x80300004	UART0 Status register (UART0.STATUS)
0x80300008	UART0 Control register (UART0.CTRL)
0x8030000C	UART0 Scaler register (UART0.SCALER)
0x80300010	UART0 FIFO debug register (UART0.FIFO)
0x80301000	UART1 Data register (UART1.DATA)
0x80301004	UART1 Status register (UART1.STATUS)
0x80301008	UART1 Control register (UART1.CTRL)
0x8030100C	UART1 Scaler register (UART1.SCALER)
0x80301010	UART1 FIFO debug register (UART1.FIFO)
0x80302000	UART2 Data register (UART2.DATA)
0x80302004	UART2 Status register (UART2.STATUS)
0x80302008	UART2 Control register (UART2.CTRL)
0x8030200C	UART2 Scaler register (UART2.SCALER)
0x80302010	UART2 FIFO debug register (UART2.FIFO)
0x80303000	UART3 Data register (UART3.DATA)
0x80303004	UART3 Status register (UART3.STATUS)
0x80303008	UART3 Control register (UART3.CTRL)
0x8030300C	UART3 Scaler register (UART3.SCALER)
0x80303010	UART3 FIFO debug register (UART3.FIFO)
0x80304000	UART4 Data register (UART4.DATA)
0x80304004	UART4 Status register (UART4.STATUS)
0x80304008	UART4 Control register (UART4.CTRL)
0x8030400C	UART4 Scaler register (UART4.SCALER)
0x80304010	UART4 FIFO debug register (UART4.FIFO)
0x80305000	UART5 Data register (UART5.DATA)
0x80305004	UART5 Status register (UART5.STATUS)
0x80305008	UART5 Control register (UART5.CTRL)
0x8030500C	UART5 Scaler register (UART5.SCALER)
0x80305010	UART5 FIFO debug register (UART5.FIFO)

19.7.1 UART Data Register

Table 163. 0x00 - DATA - UART data register

31 … 8	7 … 0
RESERVED	DATA
	NR
	rw

7: 0 Receiver FIFO (read access)

7: 0 Transmitter FIFO (write access)

19.7.2 UART Status Register

Table 164. 0x04 - STAT - UART status register

31 … 26	25 … 20	19 … 11	10	9	8	7	6	5	4	3	2	1	0
RCNT	TCNT	RESERVED	RF	TF	RH	TH	FE	PE	OV	BR	TE	TS	DR
0	0	0	0	0	0	0	0	0	0	0	1	1	0
r	r	r	r	r	r	r	rw	rw	rw	rw	r	r	r

31: 26 Receiver FIFO count (RCNT) - shows the number of data frames in the receiver FIFO. Reset: 0

25: 20 Transmitter FIFO count (TCNT) - shows the number of data frames in the transmitter FIFO. Reset: 0

10 Receiver FIFO full (RF) - indicates that the Receiver FIFO is full. Reset: 0

9 Transmitter FIFO full (TF) - indicates that the Transmitter FIFO is full. Reset: 0

8 Receiver FIFO half-full (RH) -indicates that at least half of the FIFO is holding data. Reset: 0

7 Transmitter FIFO half-full (TH) - indicates that the FIFO is less than half-full. Reset: 0

6 Framing error (FE) - indicates that a framing error was detected. Reset: 0

5 Parity error (PE) - indicates that a parity error was detected. Reset: 0

4 Overrun (OV) - indicates that one or more character have been lost due to overrun. Reset: 0

3 Break received (BR) - indicates that a BREAK has been received. Reset: 0

2 Transmitter FIFO empty (TE) - indicates that the transmitter FIFO is empty. Reset: 1

1 Transmitter shift register empty (TS) - indicates that the transmitter shift register is empty. Reset: 1

0 Data ready (DR) - indicates that new data is available in the receiver FIFO. Reset: 0

19.7.3 UART Control Register

Table 165. UART control register

31	30 16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FA	RESERVED	NS	SI	DI	BI	DB	RF	TF	R	LB	FL	PE	PS	TI	RI	TE	RE
1	0	NR	NR	NR	NR	NR	NR	NR	0	NR	0	NR	NR	NR	NR	0	0
r	r	rw	rw	rw	rw	rw	rw	rw	r	rw	rw	rw	rw	rw	rw	rw	rw

31 FIFOs available (FA) - Set to 1 when receiver and transmitter FIFOs are available.

30: 16 RESERVED

15 Number of stop bits (NS) - When set to '1' then two stop bits will be used, otherwise one stop bit will be used.

14 Transmitter shift register empty interrupt enable (SI) - When set, an interrupt will be generated when the transmitter shift register becomes empty. See section 19.6 for more details.

13 Delayed interrupt enable (DI) - When set, delayed receiver interrupts will be enabled and an interrupt will only be generated for received characters after a delay of 4 character times + 4 bits if no new character has been received during that interval. This is only applicable if receiver interrupt enable is set. See section 19.6 for more details.

12 Break interrupt enable (BI) - When set, an interrupt will be generated each time a break character is received. See section 19.6 for more details.

11 FIFO debug mode enable (DB) - when set, it is possible to read and write the FIFO debug register.

10 Receiver FIFO interrupt enable (RF) - when set, Receiver FIFO level interrupts are enabled.

9 Transmitter FIFO interrupt enable (TF) - when set, Transmitter FIFO level interrupts are enabled.

8 RESERVED and should always be set to '0' for the GR716B device

7 Loop back (LB) - if set, loop back mode will be enabled.

6 Flow control (FL) - if set, enables flow control using CTS/RTS

5 Parity enable (PE) - if set, enables parity generation and checking

4 Parity select (PS) - selects parity polarity (0 = even parity, 1 = odd parity)

3 Transmitter interrupt enable (TI) - if set, interrupts are generated when characters are transmitted (see section 19.6 for details).

2 Receiver interrupt enable (RI) - if set, interrupts are generated when characters are received (see section 19.6 for details).

1 Transmitter enable (TE) - if set, enables the transmitter.

0 Receiver enable (RE) - if set, enables the receiver.

19.7.4 UART Scaler Register

Table 166. 0x0C - SCALER - UART scaler reload register

31 20	19 0
RESERVED	SCALER RELOAD VALUE
0	NR
r	rw

19:0 Scaler reload value

19.7.5 UART FIFO Debug Register

Table 167. 0x10 - DEBUG - UART FIFO debug register

31 8	7 0
RESERVED	DATA
0	NR
r	rw

7: 0 Transmitter FIFO (read access)

7: 0 Receiver FIFO (write access)

20 Hardware Debug Support Unit

20.1 Overview

To simplify debugging on target hardware, the LEON3 processor implements a debug mode during which the pipeline is idle and the processor is controlled through a special debug interface. The LEON3 Debug Support Unit (DSU) is used to control the processor during debug mode. The DSU acts as an AHB slave and can be accessed by any AHB master. An external debug host can therefore access the DSU through several different interfaces.

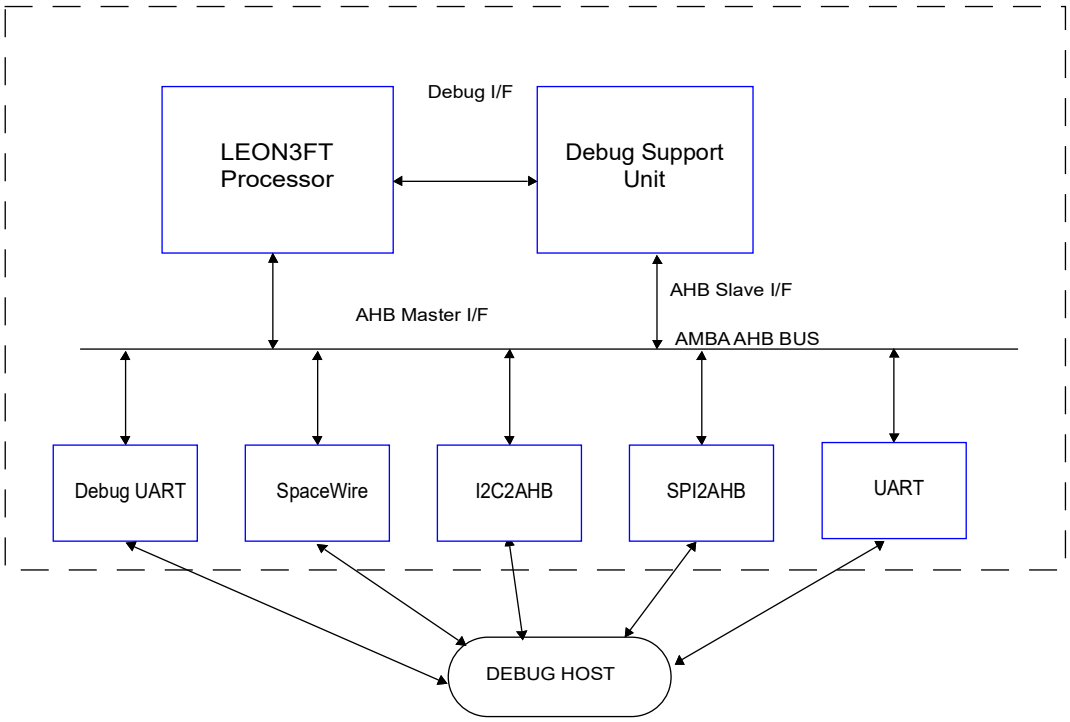


Figure 38. LEON3FT/DSU Connection

20.2 Operation

Through the DSU AHB slave interface, any AHB master can access the processor registers and the contents of the instruction trace buffer. The DSU control registers can be accessed at any time, while the processor registers and trace buffer can only be accessed when the processor has entered debug mode. In debug mode, the processor pipeline is held and the processor state can be accessed by the DSU. Entering the debug mode can occur on the following events:

- executing a breakpoint instruction (ta 1)
- integer unit hardware breakpoint/watchpoint hit (trap 0xb)
- rising edge of the external break signal (DSUBRE)
- setting the break-now (BN) bit in the DSU control register
- a trap that would cause the processor to enter error mode
- occurrence of any, or a selection of traps as defined in the DSU control register
- after a single-step operation
- the processor has entered the debug mode
- DSU AHB breakpoint or watchpoint hit

The debug mode can only be entered when the debug support unit is enabled through an external signal (DSUEN). For DSU break (DSUBRE), and the break-now BN bit, to have effect the Break-on-IU-watchpoint (BW) bit must be set in the DSU control register. This bit is set when DSUBRE is active after reset and should also be set by debug monitor software (like Frontgrade Gaisler's GRMON) when initializing the DSU. When the debug mode is entered, the following actions are taken:

- PC and nPC are saved in temporary registers (accessible by the debug unit)
- an output signal (DSUACT) is asserted to indicate the debug state
- the timer unit is (optionally) stopped to freeze the LEON timers and watchdog

The instruction that caused the processor to enter debug mode is not executed, and the processor state is kept unmodified. Execution is resumed by clearing the BN bit in the DSU control register or by de-asserting DSUEN. The timer unit will be re-enabled and execution will continue from the saved PC and nPC. Debug mode can also be entered after the processor has entered error mode, for instance when an application has terminated and halted the processor. The error mode can be reset and the processor restarted at any address.

When a processor is in the debug mode, an access to ASI diagnostic area is forwarded to the IU which performs access with ASI equal to value in the DSU ASI register and address consisting of 20 LSB bits of the original address.

20.3 AHB trace buffer

The AHB trace buffer consists of a circular buffer that stores AHB data transfers, the monitored AHB bus is either the same bus as the DSU AHB slave interface is connected to, or a completely separate bus. The address, data and various control signals of the AHB bus are stored and can be read out for later analysis. The trace buffer is 128 wide. The way information stored is indicated in the table below:

*Table 168.*AHB Trace buffer data allocation

Bits	Name	Definition
127	AHB breakpoint hit	Set to '1' if a DSU AHB breakpoint hit occurred.
126	-	Not used
125:96	Time tag	DSU time tag counter
95:80	-	Not used
79	Hwrite	AHB HWRITE
78:77	Htrans	AHB HTRANS
76:74	Hsize	AHB HSIZE
73:71	Hburst	AHB HBURST
70:67	Hmaster	AHB HMASTER
66	Hmastlock	AHB HMASTLOCK
65:64	Hresp	AHB HRESP
63:32	Load/Store data	AHB HRDATA/HWDATA(31:0)
31:0	Load/Store address	AHB HADDR

In addition to the AHB signals, the DSU time tag counter is also stored in the trace.

The trace buffer is enabled by setting the enable bit (EN) in the trace control register. Each AHB transfer is then stored in the buffer in a circular manner. The address to which the next transfer is written is held in the trace buffer index register, and is automatically incremented after each transfer. Tracing is stopped when the EN bit is reset, or when a AHB breakpoint is hit. Tracing is temporarily suspended when the processor enters debug mode, unless the trace force bit (TF) in the trace control register is set. If the trace force bit is set, the trace buffer is activated as long as the enable bit is set.

The force bit is reset if an AHB breakpoint is hit and can also be cleared by software. Note that neither the trace buffer memory nor the breakpoint registers (see below) can be read/written by software when the trace buffer is enabled.

The DSU has an internal time tag counter and this counter is frozen when the processor enters debug mode. When AHB tracing is performed in debug mode (using the trace force bit) it may be desirable to also enable the time tag counter. This can be done using the timer enable bit (TE). Note that the time tag is also used for the instruction trace buffer and the timer enable bit should only be set when using the DSU as an AHB trace buffer only, and not when performing profiling or software debugging. The timer enable bit is reset on the same events as the trace force bit.

20.3.1 AHB trace buffer filters

The DSU is implemented with filters that can be applied to the AHB trace buffer, breakpoints and watchpoints. These filters are controlled via the AHB trace buffer filter control and AHB trace buffer filter mask registers. The fields in these registers allows masking access characteristics such as master, slave, read, write and address range so that accesses that correspond to the specified mask are not written into the trace buffer. Address range masking is done using the second AHB breakpoint register set. The values of the LD and ST fields of this register has no effect on filtering.

20.3.2 AHB statistics

The DSU generates statistics from the traced AHB bus. Statistics is collected and output to LEON statistics unit (L3STAT). The statistical outputs can be filtered by the AHB trace buffer filters, this is controlled by the Performance counter Filter bit (PF) in the AHB trace buffer filter control register. The DSU can collect data for the events listed in table 169 below.

*Table 169.*AHB events

Event	Description	Note
idle	HTRANS=IDLE	Active when HTRANS IDLE is driven on the AHB slave inputs and slave has asserted HREADY.
busy	HTRANS=BUSY	Active when HTRANS BUSY is driven on the AHB slave inputs and slave has asserted HREADY.
nseq	HTRANS=NONSEQ	Active when HTRANS NONSEQ is driven on the AHB slave inputs and slave has asserted HREADY.
seq	HTRANS=SEQ	Active when HTRANS SEQUENTIAL is driven on the AHB slave inputs and slave has asserted HREADY.
read	Read access	Active when HTRANS is SEQUENTIAL or NON-SEQUENTIAL, slave has asserted HREADY and the HWRITE input is low.
write	Write access	Active when HTRANS is SEQUENTIAL or NON-SEQUENTIAL, slave has asserted HREADY and the HWRITE input is high.
hsize[5:0]	Transfer size	Active when HTRANS is SEQUENTIAL or NON-SEQUENTIAL, slave has asserted HREADY and HSIZE is BYTE (hsize[0]), HWORD (HSIZE[1]), WORD (hsize[2]), DWORD (hsize[3]), 4WORD hsize[4], or 8WORD (hsize[5]).
ws	Wait state	Active when HREADY input to AHB slaves is low and AMBA response is OKAY.
retry	RETRY response	Active when master receives RETRY response
split	SPLIT response	Active when master receives SPLIT response

*Table 169.*AHB events

Event	Description	Note
spdel	SPLIT delay	Active during the time a master waits to be granted access to the bus after reception of a SPLIT response. The core will only keep track of one master at a time. This means that when a SPLIT response is detected, the core will save the master index. This event will then be active until the same master is re-allowed into bus arbitration and is granted access to the bus. This also means that the delay measured will include the time for re-arbitration, delays from other ongoing transfers and delays resulting from other masters being granted access to the bus before the SPLIT:ed master is granted again after receiving SPLIT complete. If another master receives a SPLIT response while this event is active, the SPLIT delay for the second master will not be measured.
locked	Locked access	Active while the HMASTLOCK signal is asserted on the AHB slave inputs.

20.4 Instruction trace buffer

The instruction trace buffer consists of a circular buffer that stores executed instructions. The instruction trace buffer is located in the processor, and read out via the DSU. The trace buffer is 128 bits wide, the information stored is indicated in the table below:

*Table 170.*Instruction trace buffer data allocation

Bits	Name	Definition
127	-	Unused
126	Multi-cycle instruction	Set to ‘1’ on the second and third instance of a multi-cycle instruction (LDD, ST or FPOP)
125:96	Time tag	The value of the DSU time tag counter
95:64	Load/Store parameters	Instruction result, Store address or Store data
63:34	Program counter	Program counter (2 lsb bits removed since they are always zero)
33	Instruction trap	Set to ‘1’ if traced instruction trapped
32	Processor error mode	Set to ‘1’ if the traced instruction caused processor error mode
31:0	Opcode	Instruction opcode

During tracing, one instruction is stored per line in the trace buffer with the exception of multi-cycle instructions. Multi-cycle instructions are entered two or three times in the trace buffer. For store instructions, bits [95:64] correspond to the store address on the first entry and to the stored data on the second entry (and third in case of STD). Bit 126 is set on the second and third entry to indicate this. A double load (LDD) is entered twice in the trace buffer, with bits [95:64] containing the loaded data. Bit 126 is set for the second entry.

When the processor enters debug mode, tracing is suspended. The trace buffer and the trace buffer control register can be read and written while the processor is in the debug mode. During the instruction tracing (processor in normal mode) the trace buffer and trace buffer control register 0 can not be written. The traced instructions can optionally be filtered on instruction types. Which instructions are traced is defined in the instruction trace register [31:28], as defined in the table below:

*Table 171.*Trace filter operation

Trace filter	Instructions traced
0x0	All instructions
0x1	SPARC Fomat 2 instructions
0x2	Control-flow changes. All Call, branch and trap instructions including branch targets
0x4	SPARC Format 1 instructions (CALL)
0x8	SPARC Format 3 instructions except LOAD or STORE
0xC	SPARC Format 3 LOAD or STORE instructions
0xD	SPARC Format 3 LOAD or STORE instructions to alternate space
0xE	SPARC Format 3 LOAD or STORE instructions to alternate space 0x80 - 0xFF

20.5 Using the DSU trace buffer

The debug monitor GRMON3 has build-in support for using trace buffer in the DSU. For more information see chapter for using the trace buffer in the GRMON3 User's Manual [GRMON3].

20.6 DSU memory map

The DSU memory map can be seen in table 172 below.

Note: The DSU memory interface is intended to be accessed by a debug monitor. Software running on the LEON processors should not access the DSU interface. Registers, such as ASR registers, may not have all fields available via the DSU interface

*Table 172.*DSU memory map

Address offset	Register
0x000000	DSU control register
0x000008	Time tag counter
0x000020	Break and Single Step register
0x000024	Debug Mode Mask register
0x000040	AHB trace buffer control register
0x000044	AHB trace buffer index register
0x000048	AHB trace buffer filter control register
0x00004c	AHB trace buffer filter mask register
0x000050	AHB breakpoint address 1
0x000054	AHB mask register 1
0x000058	AHB breakpoint address 2
0x00005c	AHB mask register 2
0x100000 - 0x10FFFF	Instruction trace buffer (..0: Trace bits 127 - 96, ..4: Trace bits 95 - 64, ..8: Trace bits 63 - 32, ..C : Trace bits 31 - 0)
0x110000	Instruction Trace buffer control register 0
0x110004	Instruction Trace buffer control register 1
0x200000 - 0x210000	AHB trace buffer (..0: Trace bits 127 - 96, ..4: Trace bits 95 - 64, ..8: Trace bits 63 - 32, ..C : Trace bits 31 - 0)
0x300000 - 0x3007FC	IU register file, port1 (%asr16.dpsel = 0) IU register file, port 2 (%asr16.dpsel = 1)
0x300800 - 0x300FFC	IU register file information for correctable and uncorrectable errors
0x301000 - 0x30107C	FPU register file
0x400000 - 0x4FFFFC	IU special purpose registers

*Table 172.*DSU memory map

Address offset	Register
0x400000	Y register
0x400004	PSR register
0x400008	WIM register
0x40000C	TBR register
0x400010	PC register
0x400014	NPC register
0x400018	FSR register
0x40001C	CPSR register
0x400020	DSU trap register
0x400024	DSU ASI register
0x400040 - 0x40007C	ASR16 - ASR31
0x700000 - 0x7FFFFC	ASI diagnostic access (ASI = value in DSU ASI register, address = address[19:0]) ASI = 0x9 : Local instruction RAM, ASI = 0xB : Local data RAM

20.7 DSU registers

20.7.1 DSU control register

The DSU is controlled by the DSU control register:

*Table 173.*0x000000 - CTRL - DSU control register

31	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED		PW	HL	PE	EB	EE	DM	BZ	BX	BS	BW	BE	TE
0		0	0	0	*	*		*	*	*	*	*	*
r		r	rw	rw	r	r	r	rw	rw	rw	rw	rw	rw

31: 12	Reserved
11	Power down (PW) - Returns ‘1’ when processor is in power-down mode.
10	Processor halt (HL) - Returns ‘1’ on read when processor is halted. If the processor is in debug mode, setting this bit will put the processor in halt mode.
9	Processor error mode (PE) - returns ‘1’ on read when processor is in error mode, else ‘0’. If written with ‘1’, it will clear the error and halt mode.
8	External Break (EB) - Value of the external DSUBRE signal (read-only)
7	External Enable (EE) - Value of the external DSUEN signal (read-only)
6	Debug mode (DM) - Indicates when the processor has entered debug mode (read-only).
5	Break on error traps (BZ) - if set, will force the processor into debug mode on all *except* the following traps: priviledged_instruction, fpu_disabled, window_overflow, window_underflow, asynchronous_interrupt, ticc_trap.
4	Break on trap (BX) - if set, will force the processor into debug mode when any trap occurs.
3	Break on S/W breakpoint (BS) - if set, debug mode will be forced when an breakpoint instruction (ta 1) is executed.
2	Break on IU watchpoint (BW) - if set, debug mode will be forced on a IU watchpoint (trap 0xb).
1	Break on error (BE) - if set, will force the processor to debug mode when the processor would have entered error condition (trap in trap).
0	Trace enable (TE) - Enables instruction tracing. If set the instructions will be stored in the trace buffer. Remains set when then processor enters debug or error mode

20.7.2 DSU Break and Single Step register

This register is used to break or single step the processor(s).

Table 174. 0x000020 - BRSS - BRSS - DSU Break and Single Step register

31 16	15 0
SS[15:0]	BN[15:0]

31: 16 Single step (SSx) - if set, the processor x will execute one instruction and return to debug mode. The bit remains set after the processor goes into the debug mode. As an exception, if the instruction is a branch with the annul bit set, and if the delay instruction is effectively annulled, the processor will execute the branch, the annulled delay instruction and the instruction thereafter before returning to debug mode.

15: 0 Break now (BNx) -Force processor x into debug mode if the Break on watchpoint (BW) bit in the processors DSU control register is set. If cleared, the processor x will resume execution.

20.7.3 DSU Debug Mode Mask Register

When the processors enters the debug mode the value of the DSU Debug Mode Mask register determines if the other processor is forced in the debug mode.

Table 175. 0x000024 - DBGM - DSU Debug Mode Mask register

31 17	16	15 1	0
Reserved	DM	Reserved	ED

31: 16 Debug mode mask (DMx) - If set, the processor will not be able to force running processor into debug mode even if it enters debug mode.

15: 0 Enter debug mode (ED) - Force processor into debug mode If 0, the processor will not enter the debug mode.

20.7.4 DSU trap register

The DSU trap register is a read-only register that indicates which SPARC trap type that caused the processor to enter debug mode. When debug mode is force by setting the BN bit in the DSU control register, the trap type will be 0xb (hardware watchpoint trap).

Table 176. 0x400020 - DTR - DSU Trap register

31 13	12	11 4	3 0
RESERVED	EM	TRAPTYPE	R

31: 13 RESERVED

12 Error mode (EM) - Set if the trap would have cause the processor to enter error mode.

11: 4 Trap type (TRAPTYPE) - 8-bit SPARC trap type

3: 0 Read as 0x0

20.7.5 DSU time tag counter

The trace buffer time tag counter is incremented each clock as long as the processor is running. The counter is stopped when the processor enters debug mode and when the DSU is disabled (unless the

timer enable bit in the AHB trace buffer control register is set), and restarted when execution is resumed.

Table 177. 0x000008 - DTTC - DSU time tag counter

31	0
TIMETAG	
0	
rw	

31: 0 DSU Time Tag Value (TIMETAG)

The value is used as time tag in the instruction and AHB trace buffer.

20.7.6 DSU ASI register

The DSU can perform diagnostic accesses to different ASI areas. The value in the ASI diagnostic access register is used as ASI while the address is supplied from the DSU.

Table 178. 0x400024 - DASI - ASI diagnostic access register

31	8	7	0
RESERVED		ASI	
0		NR	
r		rw	

31: 8 RESERVED

7: 0 ASI (ASI) - ASI to be used on diagnostic ASI access

20.7.7 AHB Trace buffer control register

The AHB trace buffer is controlled by the AHB trace buffer control register:

Table 179. 0x000040 - ATBC - AHB trace buffer control register

31	16	15	8	7	6	5	4	3	2	1	0
DCNT		RESERVED	DF	SF	TE	TF	BW		BR	DM	EN
0		0	0	0	0	0	0		0	0	0
rw		r	rw	rw	rw	rw	r		rw	rw	rw

31: 16 Trace buffer delay counter (DCNT)

15: 8 RESERVED

7 Sample Force (SF) - If this bit is written to ‘1’ it will have the same effect on the AHB trace buffer as if HREADY was asserted on the bus at the same time as a sequential or non-sequential transfer is made. This means that setting this bit to ‘1’ will cause the values in the trace buffer’s sample registers to be written into the trace buffer, and new values will be sampled into the registers. This bit will automatically be cleared after one clock cycle.

Writing to the trace buffer still requires that the trace buffer is enabled (EN bit set to ‘1’) and that the CPU is not in debug mode or that tracing is forced (TF bit set to ‘1’). This functionality is primarily of interest when the trace buffer is tracing a separate bus and the traced bus appears to have frozen.

6 Timer enable (TE) - Activates time tag counter also in debug mode.

5 Trace force (TF) - Activates trace buffer also in debug mode. Note that the trace buffer must be disabled when reading out trace buffer data via the core’s register interface.

4: 3 Bus width (BW) - This value corresponds to log2(Supported bus width / 32)

2 Break (BR) - If set, the processor will be put in debug mode when AHB trace buffer stops due to AHB breakpoint hit.

1 Delay counter mode (DM) - Indicates that the trace buffer is in delay counter mode.

0 Trace enable (EN) - Enables the trace buffer.

20.7.8 AHB trace buffer index register

The AHB trace buffer index register contains the address of the next trace line to be written.

Table 180. 0x000044 - ATBI - AHB trace buffer index register

31	4	3	0
INDEX		R	
NR		0	
rw		r	

31: 4 Trace buffer index counter (INDEX)

3: 0 Read as 0x0

20.7.9 AHB trace buffer filter control register

*Table 181.*0x000048 - ATBFC - AHB trace buffer filter control register

31 14	13 12	11 10	9 8	7 4	3	2	1	0
RESERVED	WPF	R	BPF	RESERVED	PF	AF	FR	FW
0	0	0	0	0	0	0	0	0
r	rw	r	rw	r	rw	rw	rw	rw

31: 14 RESERVED

13: 12 AHB watchpoint filtering (WPF) - Bit 13 of this field applies to AHB watchpoint 2 and bit 12 applies to AHB watchpoint 1. If the WPF bit for a watchpoint is set to '1' then the watchpoint will not trigger unless the access also passes through the filter. This functionality can be used to, for instance, set a AHB watchpoint that only triggers if a specified master performs an access to a specified slave.

11: 10 RESERVED

9: 8 AHB breakpoint filtering (BPF) - Bit 9 of this field applies to AHB breakpoint 2 and bit 8 applies to AHB breakpoint 1. If the BPF bit for a breakpoint is set to '1' then the breakpoint will not trigger unless the access also passes through the filter. This functionality can be used to, for instance, set a AHB breakpoint that only triggers if a specified master performs an access to a specified slave. Note that if a AHB breakpoint is coupled with an AHB watchpoint then the setting of the corresponding bit in this field has no effect.

7: 4 RESERVED

3 Performance counter Filter (PF) - If this bit is set to '1', the cores performance counter (statistical) outputs will be filtered using the same filter settings as used for the trace buffer. If a filter inhibits a write to the trace buffer, setting this bit to '1' will cause the same filter setting to inhibit the pulse on the statistical output.

2 Address Filter (AF) - If this bit is set to '1', only the address range defined by AHB trace buffer breakpoint 2's address and mask will be included in the trace buffer.

1 Filter Reads (FR) - If this bit is set to '1', read accesses will not be included in the trace buffer.

0 Filter Writes (FW) - If this bit is set to '1', write accesses will not be included in the trace buffer.

20.7.10 AHB trace buffer filter mask registeri

*Table 182.*0x00004C - ATBFM - AHB trace buffer filter mask register

31 16	15 0
SMASK[15:0]	MMASK[15:0]
0	0
rw	rw

31: 16 Slave Mask (SMASK) - If SMASK[n] is set to '1', the trace buffer will not save accesses performed to slave n.

15: 0 Master Mask (MMASK) - If MMASK[n] is set to '1', the trace buffer will not save accesses performed by master n.

20.7.11 AHB trace buffer breakpoint registers

The DSU contains two breakpoint registers for matching AHB addresses. A breakpoint hit is used to freeze the trace buffer by automatically clearing the enable bit. Freezing can be delayed by programming the DCNT field in the trace buffer control register to a non-zero value. In this case, the DCNT value will be decremented for each additional trace until it reaches zero, after which the trace buffer is frozen. A mask register is associated with each breakpoint, allowing breaking on a block of addresses. Only address bits with the corresponding mask bit set to '1' are compared during breakpoint detection. To break on AHB load or store accesses, the LD and/or ST bits should be set.

*Table 183.*0x000050, 0x000058 - ATBBA - AHB trace buffer break address register

31 2	1 0
BADDR[31:2]	R

*Table 183.*0x000050, 0x000058 - ATBBA - AHB trace buffer break address register

NR	0
rw	r

31: 2 Break point address (BADDR) - Bits 31:2 of breakpoint address

1: 0 Read as 0b00

*Table 184.*0x000054, 0x00005C - ATBBM - AHB trace buffer break mask register

31 2	1	0
BMASK[31:2]	LD	ST
NR	0	0
rw	rw	rw

31: 2 Breakpoint mask (BMASK) - (see text)

1 Load (LD) - Break on data load address

0 Store (ST) - Break on data store address

20.7.12 Instruction trace control register 0

The instruction trace control register 0 contains a pointer that indicates the next line of the instruction trace buffer to be written.

*Table 185.*0x110000 - ITBCO - Instruction trace control register 0

31 29	28 16	15 0
	RESERVED	ITPOINTER
	0	NR
	r	rw

31: 28 Trace filter configuration

27: 16 RESERVED

15: 0 Instruction trace pointer (ITPOINTER)

20.7.13 Instruction trace control register 1

The instruction trace control register 1 contains settings used for trace buffer overflow detection. This register can be written while the processor is running.

*Table 186.*0x110004 - ITBCI - Instruction trace control register 1

31 28	27	26 24	23	22 0
RESERVED	WO	TLIM	OV	RESERVED
0	0	0	0	0
r	rw	rw	rw	r

31: 28 RESERVED

27 Watchpoint on overflow (WO) - If this bit is set, and Break on iu watchpoint (BW) is enabled in the DSU control register, then a watchpoint will be inserted when a trace overflow is detected (TOV field in this register gets set).

26: 24 Trace Limit (TLIM) - TLIM is compared with the top bits of ITPOINTER in Instruction trace control register 0 to generate the value in the TOV field below.

23 Trace Overflow (TOV) - Gets set to ‘1’ when the DSU detects that TLIM equals the top three bits of ITPOINTER.

22: 0 RESERVED

21 On-chip Dual-port Memory with EDAC Protection

The GR716B has 6 Local on-chip SRAM with EDAC (LRAM) units. Two within the main LEON3FT processor providing 64 KiB of local instruction RAM (ILRAM) and another 64 KiB of local data RAM (DLRAM). Additionally there are 2 LRAM units within each RTA subsystem providing 16 KiB of ILRAM and 16 KiB of DLRAM for each RTA.

Each 16 KiB section of Local on-chip SRAM with EDAC (LRAM) has a unique control and status register interfaces. In total there are 12 such sets of registers. The AMBA addresses are listed in chapter 2.10 and the register interface is documented in section 21.3. The registers for the main processor LRAM are located in the address ranges 0x80001000-0x8000107F (DLRAM) and 0x8000B000-0x8000B07F (ILRAM). For the RTAs they are instead located in the address ranges 0x62020000-0x6202001F (RTA0 ILRAM), 0x62030000-0x6203001F (RTA0 DLRAM), 0x72020000-0x7202001F (RTA1 ILRAM), and 0x72030000-0x7203001F (RTA1 DLRAM). The LRAM unit bus connections are illustrated in Figure 39.

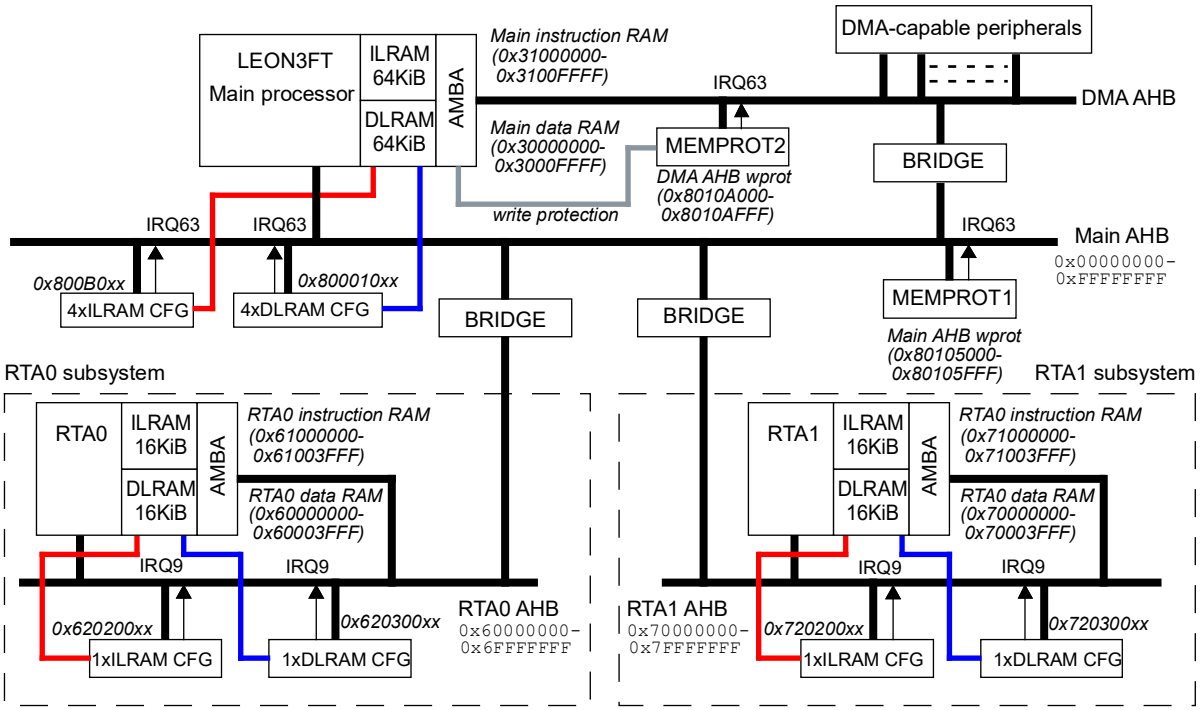


Figure 39. GR716B LRAM bus connection, including both main CPU and RTAs.

21.1 Overview

The LEON3FT microcontroller includes a 64 KiB Dual port SRAM with EDAC for local instruction (ILRAM) execution and 64 KiB Dual port SRAM with EDAC for local data storage (DLRAM). This chapter describes the functionality of the instruction and data memory in the LEON3FT microcontroller. It also describes the 16 KiB instruction and 16 KiB data memories of RTA0 and RTA1.

The local instruction and data memory can be accessed both directly from the processor or RTA they are connected to and from their AMBA bus interface. For the main processor, the AMBA interface is connected to the DMA AHB while for the RTAs it is connected to the AHB subsystem bus. The different AHB buses are interconnected by bridges and share a global coherent memory map (section 2.10), but accesses across a bridge add latency to all accesses. Accesses to an LRAM from a processor always have precedence over accesses made via the AMBA interface. The processor interface and priority scheme guarantees single cycle instruction execution and data load in the LEON3FT processor and RTAs. Data single-word (32 bits) stores takes 2 cycles, double-word store (64 bits) completes in 3 cycles, limited by the processor pipeline (see section 17.2.2).

The instruction and data memory implements a control interface accessible via the AMBA APB interface. See section 21.3 and 2.10 for register description and base addresses. The instruction and data on-chip memory implements volatile memory that is protected by means of Error Detection And Correction (EDAC). One error can be corrected and two errors can be detected, which is performed by using a (39, 32, 7) BCH code.

The system can be configured to protect and restrict access to the Local on-chip SRAM with EDAC (LRAM) units configuration and memory area in the **MEMPROT** units. A write to a protected area without write permission to will result in a AMBA error and the write request to the memory will be ignored. See section 47 for more information.

Figure 40 shows a block diagram of the internals of the controller for one 16 KiB LRAM slice. The larger 64 KiB LRAM are built from four such slices and each 16 KiB slice therefore has its own scrubber.

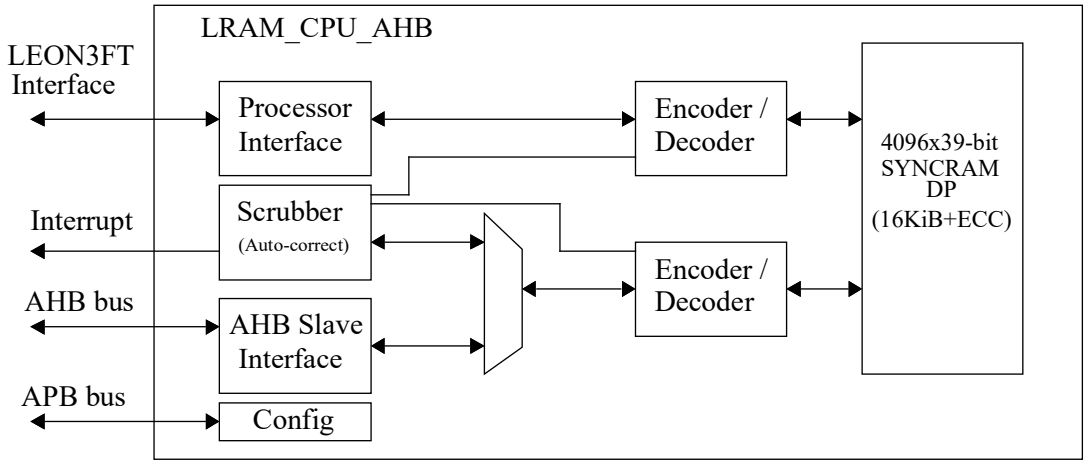


Figure 40. Block diagram for one 16 KiB LRAM slice.

21.2 Operation

21.2.1 EDAC

The EDAC checksum is always updated for write operations, but only checked during reads when enabled by the LRAMCFG.EN configuration field. When correctable error is detected a counter LRAMCFG.ECNT is incremented and can be used to monitor the error rate.

When an uncorrectable error is detected the read operation will complete with an error response. For the processor port, this means an exception. For the AHB port it means an AHB error response. Detection of an uncorrectable error will also cause LRAMCFG.MECNT to be incremented.

Note that EDAC is disabled on reset and must be enabled by software or the boot ROM (section 51).

At power-up, the LRAM has unpredictable contents and nearly every address will have contents that would be interpreted by the EDAC as an uncorrectable error. Before using the memory, software or the boot ROM (section 51) must write valid data and checksums to every word location. This can be doe using the scrubber's wash feature, see section 21.2.4.1.

Diagnostic read, write, and error injection is possible via the scrubber register interface. See sections 21.2.4.2 and 21.2.4.3.

21.2.2 AHB interface

For single read or the first beat in a read burst the access is performed with 2 wait-states. For the continuing read burst, no wait-states are added to the access. Sub-word writes has the same bus timing as single read and the access is performed with 2 wait-states. Sub-word writes are performed as read-modify-write operations by the memory controller to be able to correctly update the checksum. For

word write no wait-states are added. The access on the AHB port can be stalled due to scrub operations or when a write conflict is detected between the AHB port and the CPU port. When the CPU port performs a read to a specific address, a write on the AHB port to the same address is stalled until the CPU read has completed. This feature is enabled by the LRAMCFG.PC configuration field.

In a read where a correctable error is detected, the correction is performed on the fly and corrected data is delivered to the AHB bus. Additionally, write-back of the corrected data to memory is performed. The automatic write-back feature can optionally be disabled by the LRAMCFG.ACOR configuration field. When an uncorrectable error is detected the access will terminate with an AMBA ERROR response. Errors can be detected during read accesses of any width and on sub-word write accesses. Word writes and multi-word burst writes will never cause detection of errors.

When a write-protected area (defined by the AMBA protection unit) is written, the access will be terminated with a AMBA ERROR response.

21.2.3 Processor interface

The CPU port is designed to allow word reads and writes with no stalling. Sub-word writes to local data memory (DLRAM) are implemented as read-modify-write by the CPU. This port is not affected by the scrubbing operations. Data store/load operations to local instruction memory (ILRAM) is restricted to word size only. Double-word, half-word and byte load/store to ILRAM will cause data_access_exception (see section 17.2.10). Data load/store operations to DLRAM are allowed for all access sizes (byte, half-word, word, or double-word).

In case a read access results in detection of a correctable error, the data is corrected on the fly before passed to the CPU. Additionally, the corrected data is automatically written back to the memory without stalling the CPUport. The automatic write-back can optionally be disabled using the LRAMCFG.ACOR configuration field.

When an uncorrectable error is detected, the access will terminate with an exception. In case of an instruction fetch, this will be instruction_access_exception. For a data load, this will be data_access_exception. In case of a sub-word store (8-bit or 16-bit), it will be data_store_error. case of a sub-word (8-bit or 16-bit) write access, see section 17.2.10. Word and double-word (32-bit and 64-bit) writes cannot cause detection of uncorrectable errors since the prior memory contents are then overwritten without readback.

21.2.4 Scrubber

Each instruction and data memory for the main processor is divided into four slices, with each slice provided with individual configuration registers and scrubber. The configuration registers and their address map are described in section 21.3. For the RTAs, the instruction and data memory each consist of a single slice and scrubber each. In total there are 12 LRAM slices and scrubbers for in the GR716B (8 for the main processor and 2 for each RTA).

The scrubber is designed to loop through all memory locations and check for errors. The scrubbing is performed as a dummy read access which in case of a detected correctable error will trigger the auto-correction feature (which would perform a read-modify-write to update the data and checksum). When a uncorrectable error is detected, the scrubber will not alter the memory location. Instead an interrupt can be generated (by setting the LRAMCFG.IE field to '1'). The scrubber can also be configured to be disabled once a uncorrectable error is detected (by setting the SCRUBCFG.DISE field to '1'). In this case the offset of the failing memory location can be read out from the SCRUBCTRL.ADDR field (this field would in this case point to the memory location directly after the failing location). The scrubbing rate can be configured with the SCRUBCFG.DELAY field. The value of this field sets the number of clock cycles between each scrubbing access. One scrub access will be made every DELAY+1 system clock cycles, and scrubbing a complete slice therefore takes 4096*(DELAY+1). I.e. any given address in the slice will be scrubbed once per 4096*(DELAY+1) clock cycles.

Note that each scrubber covers only one 16 KiB slice. For the main processor there are four 16 KiB per LRAM instance and hence four scrubbers per LRAM instance, each controlled by an independent set of registers.

21.2.4.1 Wash

The scrubber can be configured to wash the memory (writing to all memory locations) and generate valid checksums. To start a wash operation, first ensure that the scrubber is disabled (SCRUBCTRL.SEN=0). Then set the SCRUBCFG.WASH field to '1', and the XCB, RCB and WCB fields in the same register to 0. The wash operation is started by writing '1' to the SEN field in the SCRUBCTRL register. The same write will also set the start address of the wash operation. The address should be set to 0 to wash the entire memory slice. The wash operation writes to one address per clock cycle and hence takes 4096 clock cycles to complete, if started at address 0. During this time, the SCRUBCTRL.PEN field will read '1'. When the wash operation completes, the scrubber will clear the SCRUBCFG.WASH register field. After the wash operation completes, the scrubber will remain enabled, but in scrub mode. If this behavior is not wanted, then SCRUBCFG.DISW can be set to '1' in which case the scrubber will instead be disabled after the wash operation completes.

21.2.4.2 Diagnostic read/write

The scrubber can be configured to reads or write the data and checksum directly using the SCRUBDATA register and SCRUBCFG.CB register field. Before performing a diagnostic read or write, the scrubber must be disabled (SCRUBCTRL.SEN=0). At most one of the RCB, WCB and XCB fields in SCRUBCFG should be set to 1 at any given time.

Diagnostic read mode is enabled by setting the SCRUBCFG.RCB field to 1. To perform a diagnostic read, write to the SCRUBCTRL register with the PEN field set to 1. This will cause a diagnostic read to the address specified by the ADDR field. When the access has completed (SCRUBCTRL.PEN = '0'), the undecoded data can be read from the SCRUBDATA register and the checksum can be read from the SCRUBCFG.CB register field. If EDAC is enabled (LRAMCFG.EN=1) during a diagnostic readout, the correctable and uncorrectable error counters will increment if an error is detected during the diagnostic read, but the SCRUBDATA register will still be the undecoded data from the codeword. And automatic write-back of the corrected data will not be performed.

Diagnostic write mode is enabled by setting the SCRUBCFG.WCB field to 1. To perform a diagnostic write access first write the data to the SCRUBDATA register and the checksum to the SCRRUBCFG.CB register field. Then write to the SCRUBCTRL register with the PEN field set to 1. This will cause a diagnostic write to the address specified by the ADDR field, with data taken from the SCRUBDATA register and checksum from the SCRUBCFG.CB register field.

Additionally, the LRAMCFG.RB register field provides a method for diagnostic checksum read (but not write) directly from the AHB port without use of the scrubber. When LRAMCFG.RB=1 a read to the AHB port will return the 7-bit checksum in bits 6:0 with bits 31:7 reading 0. The CPU port is unaffected by this setting. Writes to the AHB port are also affected by this setting.

Note that the register interface of a given slice and only perform diagnostic reads and writes to the same 16 KiB slice. The ADDR field of the SCRUBCTRL register is

21.2.4.3 Error injection

Besides diagnostic read and writes, the scrubber also implements a read-modify-write operation for where the checksum is XOR:ed with a bit mask for convenient error injection. The scrubber must be disabled (SCRUBCTRL.SEN=0) to use this mode. At most one of the RCB, WCB and XCB fields in SCRUBCFG should be set to 1 at any given time.

When injecting an error in a memory location, the scrubber will read the data from an address, calculate the checksum for this data, XOR the checksum with the SCRUBCFG.CB field, then write back both the data and the XOR:ed checksum to the same address. This means that a single-bit error (correctable) can be injected into the checksum part of the codeword by setting a single bit in the CB field

non-zero. A two-bit error (detectable uncorrectable error) can be injected into the checksum part of the codeword by setting two bits in the CB field non-zero. Injecting errors into the data part of the codeword is not possible with this method, but can be done using diagnostic reads and writes (section 21.2.4.2).

If EDAC is enabled (LRAMCFG.EN=1), then the data that is the basis of the checksum calculation before the XOR operation is the decoded codeword. If EDAC is disabled (LRAMCFG.EN=0) then the basis of the data is instead the undecoded data portion of the codeword. This has a functional impact in case the address that an error is injected into already contains a correctable or uncorrectable error. In case of an uncorrectable error during the XCB operation, it is undefined what the corrected data will be. The correctable and uncorrectable error counters (ECNT and MECNT fields of LRAMCFG) will not increment due to an error injection operation with XCB=1.

To perform error injection with XOR, first set the SCRUBCFG.XCB field to '1' and configure the XOR pattern in the SCRUBCFG.CB field. Then write to the SCRUBCTRL register with the PEN field set to 1. The error injection will be performed to the address that was specified in the ADDR field in the write to SCRUBCTRL.

Note that each scrubber covers only one 16 KiB memory slice and can only inject errors into its own slice. Attempting to inject an error outside of this range will result in an overflow and cause an injection in the same slice. The SCRUBCTRL.ADDR field is only 12 bits wide and the 18 most significant address bits will be ignored. In order to inject errors into the entire memory, the correct scrubber register offset must be used.

21.3 Local memory address map

The local memory control registers is programmed via registers mapped into APB address space and the memory area is accessible via processor local interface or via AHB address space.

*Table 187.*Local Data and Instruction memory map and configuration registers

AMBA address range or offset	Register	Acronym
0x30000000 - 0x3000FFFF [1) 2)]	Main processor local data memory area	DLRAM
0x31000000 - 0x3100FFFF [3)]	Main processor local instruction memory	ILRAM
0x60000000 - 0x60003FFF [4) 5)]	RTA0 local data memory area	RTA0.DLRAM
0x61000000 - 0x61003FFF [6)]	RTA0 local instruction memory area	RTA0.ILRAM
0x70000000 - 0x70003FFF [4) 5)]	RTA1 local data memory area	RTA1.DLRAM
0x71000000 - 0x71003FFF [6)]	RTA1 local instruction memory area	RTA1.ILRAM
0x80001000-0x8000107F	Control and status registers for main data LRAM	DLRAM.*
0x80001080-0x80001FFF	Reserved	N/A
0x8000B000-0x8000B07F	Control and status registers for main instruction LRAM	ILRAM.*
0x8000B080-0x8000BFFF	Reserved	N/A
0x62030000-0x6203001F	Control and status registers for RTA0 data LRAM	RTA0.DLRAM.*
0x62030020-0x6203FFFF	Reserved	N/A
0x62020000-0x6202001F	Control and status registers for RTA0 instruction LRAM	RTA0.ILRAM.*
0x62020020-0x6202FFFF	Reserved	N/A
0x72030000-0x7203001F	Control and status registers for RTA1 data LRAM	RTA1.DLRAM.*
0x72030020-0x7203FFFF	Reserved	N/A
0x72020000-0x7202001F	Control and status registers for RTA1 instruction LRAM	RTA1.ILRAM.*
0x72020020-0x7202FFFF	Reserved	N/A
0x00+*m**0x20 [7)]	LRAM memory slice *m* configuration register	*.LRAMCFG*m*
0x04+*m**0x20 [7)]	LRAM memory slice *m* scrubber data	*.SCRUBDATA*m*
0x08+*m**0x20 [7)]	LRAM memory slice *m* scrubber control	*.SCRUBCTRL*m*
0x0C+*m**0x20 [7)]	LRAM memory slice *m* scrubber configuration	*.SCRUBCFG*m*
0x10+*m**0x20 to 0x1F+*m**0x20 [7)]	Reserved.	N/A

1) LEON3FT processor access address range for data fetch or store via local processor interface. Access is always single cycle access. Accessible from DMA-capable peripherals via DMA bus interface.

2) LEON3FT processor access address range for Instruction fetch via system and DMA bus interface. Latency for instruction fetch is 10 clock cycles per instruction.

3) LEON3FT processor access address range for instruction and data fetch via local processor interface. Access is always single cycle access. Data accesses (load and store) are restricted to word size only. Data store/load of double-word, half-word and byte sizes from the LEON3FT processor cause data_access_exception. Accessible from DMA-capable peripherals via DMA bus interface.

4) RTA address range for data fetch or store via local processor interface. Access is always single-cycle access. Accessible to main processor and DMA via system and RTA local bus interfaces.

5) RTA address range for instruction fetch via local bus interface. Latency of several cycles per instruction.

6) RTA address range for instruction and data fetch via local processor interface. Access is always single cycle access and data store is restricted. Data accesses (load and store) are restricted to word size only. Double-word, half-word and byte sized load and store from the RTA result in data_access_exception. Accessible from main processor and DMA -capable peripherals via system and RTA local bus interface.

7) This is an offset to be added to the control and status register base address, e.g. 0x80001000, 0x8000B000, 0x62030000, 0x62020000, 0x72030000, 0x72020000. The main 64 KiB instruction and data memory (ILRAM/DLRAM) are each divided into four slices (*m* = 0 to 3), with each slice provided with individual configuration registers. For the 16 KiB RTA memories, there is a single slice per memory with its own register interface (*m*=0).

21.4 Local memory register descriptions

Table 188. 0x00+00+*m**0x20 - LRAMCFG*m* - LRAM Configuration Register

31	30 29	28	27 24	23 16	15	14	13	12	11 10	9 8	7 4	3	2	1	0
FT	AOP	SC	MECNT	MEMSIZE	MS	RB	PC	IE	ACOR	SERR	ECNT	ES	AC	R	EN
1	0	1	0	*	0	0	0	0	0	0	0	0	0	0	0
r	r	r	wc	r	rw	rw	rw	rw	rw	wc	wc	wc	r	r	rw

31 EDAC support implemented (FT) - Reads as 1 to indicate that EDAC is implemented.

30: 29 Atomic operation implemented (AOP) - Reads as 0 to indicate that atomic operations are not implemented.

28 Scrubber implemented (SC) - Reads as 1 to indicate that scrubber is implemented.

27: 24 Uncorrectable error counter (MECNT) - The usage of this field changes depending on the value of the MS field. See below.

23: 16 LRAM memory size (MEMSIZE) - Reads 6 for the main processor memory slices (ILRAM.LRAMCFG0-3 and DLRAM.LRAMCFG0-3) to indicate a total memory size of 64 KiB = 2^6 KiB and that they are composed of 64/16=4 slices each of size 16 KiB. Reads 4 for the RTA memory slices (RTA0-1.*.LRAMCFG0) to indicate total size of 16 KiB = 2^4 KiB and 16/16=1 slice.

15 Error counter mode select (MS) - Selects between two different modes for readout and clearing the correctable and uncorrectable error counters (ECNT and MECNT). Also affects the ES field.

MS=0: Direct counter mode. In this mode any read of the ECNT and MECNT fields returns the then current value of the counter and ES reads as 0. ECNT and MECNT are cleared by writing 0b1111 to the respective fields (the fields are bitwise write-clear). In this mode there is a possibility of miscounting errors in case an error is detected in between the readout of the counters and the write that clears them.

MS=1: Shadowed counter mode. This mode eliminates the race condition described above. In this mode, reads of ECNT and MECNT return shadow registers. The ES field reads as 1 if any of the error counters is non-zero. To clear and read out the error counters in this mode, write 1 to the ES field (values written to ECNT and MECNT are ignored). This copies the current value of the error counters into the ECNT and MECNT shadow registers for readout and simultaneously clears the error counters.

14 Read Bypass (RB) - Enable diagnostic readout of checkbits from the AHB port. When RB=0, the AHB interface operates normally delivering data as configured by ACOR[0] and EN. When RB=1, the AHB interface instead delivers undecoded checkbits for the accessed address. Bits 31:7 of the returned data will then be 0 and bits 6:0 will contain the 7 checkbits.

RB has no effect on the processor port, nor on write operations from the AHB port.

13 Port write conflict detection (PC) - When enabled, a write on the AHB port to the same address as the access on the CPU port is stalled.

12 Interrupt enable (IE) - Enable the assertion of an interrupt when the scrubber detects an uncorrectable error.

11: 10 Auto-correction disable (ACOR) - Disable auto-correction write-back for detected errors. Bit[11]: processor port, bit[10]: AHB port.

9: 8 Scrub error status (SERR) - Bit[9] indicates a correctable error has been detected by the scrubber. Bit[8] indicates that an uncorrectable error has been detected by the scrubber. Write '1' to clear.

7: 4 Correctable error counter (ECNT) - The usage of this field changes depending on the value of the MS field. See above.

3 Error status (ES) - The usage of this bit changes depending on the value of the MS field. See above.

2 Access conflict (AC) - Indicates if write access conflict is detected.

1 Reserved.

0 EDAC enable (EN) - Set to 1 to enable EDAC decoding on readout. Even when zero, EDAC encoding is enabled for writes.

Table 189. 0x04+*m**0x20 - SCRUBDATA*m* - Scrubber Data Register

31	0
DATA	
nr	
rw	

31: 0 DATA - Data written to memory by the scrubber in wash mode. Also the data that is written in case of diagnostic write accesses (SCRUBCFG*m*.WCB=1). In case of diagnostic read (SCRUBCFG*m*.RCB=1) accesses the 32-bit data part of the 39-bit codeword is latched in this register without decoding.

Table 190. 0x08+*m**20 - SCRUBCTRL*m* - Scrubber Control Register

31 14	13 2	1	0
RESERVED	ADDR	PEN	SEN
0	0	0	0
r	rw	rw	rw

31: 14 Reserved

13: 2 ADDR - Scrubber address offset used for all scrubber operations. Incremented automatically every scrub/write cycle during scrub/wash operations. When ADDR is incremented past 0xFFF (corresponding to byte address 0x3FFC) it wraps to 0x000. The ADDR field is also used as address for diagnostic accesses. This register field is only writeable when SEN=0 prior to the write.

1 Scrub access pending (PEN) - Self-clearing. Reads as 1 whenever a scrub operation is in progress. During a wash operation, it remains 1 until the full wash operation completes. During scrubbing it will normally be set during two clock cycles per scrub cycle, but may be extended if there is a write collision.

This field is writeable when SEN=0. Writing a 0 has no effect. Writing a 1 is used to trigger a diagnostic access (checkbit read, checbit write, or write with XOR:ed checkbits) if enabled in SCRUBCFG*m*.

0 Scrubber enable (SEN) - The other fields in this register are only writeable only when SEN=0 prior to the write, and otherwise retain their previous state. This bit can be self-cleared after a wash operation or when an uncorrectable error is detected, if the scrubber is configured to do so in the SCRUBCFG*m* register.

Table 191. 0x0C+*m**0x20 - SCRUBCFG*m* - Scrubber Configuration Register

31 16	15 14	13	12	11	10 4	3	2	1	0
DELAY	RES	DISW	DISE	R	CB	WCB	RCB	XCB	WASH
0	0	0	0	0	0	0	0	0	0
rw	r	rw	rw	r	rw	rw	rw	rw	rw

31: 16 Scrubber delay (DELAY) - Delay in clock cycles between each scrub access. Applies only in scrub mode, not when washing. One address will be scrubbed for every DELAY+1 clock cycles. A complete scrubbing cycle for the full 16 KiB slice therefore takes 4096*(DELAY+1) clock cycles. The minimum possible scrubbing period is 3 clock cycles per address (DELAY=2)

15: 14 Reserved.

13 Disable after wash (DISW) - Disable the scrubber (set SCRUBCTRL*m*.SEN to 0) after the wash operation is complete. See WASH for details.

12 Disable after error (DISE) - Disable the scrubber (set SCRUBCTRL*m*.SEN to 0) when an uncorrectable error is detected by the scrubber.

11 Reserved.

10: 4 Checkbits (CB) - 7-bit checksum. Holds the value to diagnostically write with the next scrubber operation (if WCB or XCB is 1), or that was diagnostically read in the previous scrubber operation (if RCB was 1).

3 Enable diagnostic checksum write mode (WCB) - If the scrubber is disabled (SCRUBCTRL*m*.SEN=0) and WCB=1 then, on the next write to the SCRUBCTRL*m* register that sets the PEN field to 1, a diagnostic write will be made to the address specified in the ADDR field (bits 13:2) of the same register write. The diagnostic write bypasses EDAC encoding. The data-part of the codeword will be set to SCRUBDATA*m* and the checkbit-part of the codeword will be set to SCRUBCFG*m*.CB. At most one of WCB, RCB and XCB should be 1 at the same time. WCB remains set after the operation completse.

2 Enable diagnostic checksum read mode (RCB) - If the scrubber is disabled (SCRUBCTRL*m*.SEN=0) and RCB=1 then, on the next write to SCRUBCTRL*m* that sets the PEN field to 1, a diagnostic read will be made to the address specified in the ADDR field (bits 13:2) of the same register write. The diagnostic read bypasses EDAC decoding. The data-part of the codeword will be latched in the SCRUBDATA*m* register and the checkbit-part of the codeword will be latched in the SCRUBCFG*m*.CB register field. At most one of WCB, RCB and XCB should be 1 at the same time. RCB remains set after the operation completes. The ECNT and MECNT fields of LRAMCFG may increment if EDAC is enabled during the operation.

1 Enable error injection mode (XCB) - If the scrubber is disabled (SCRUBCTRLm.SEN=0) and XCB=1, then, on the next write to SCRUBCTRL*m* that sets the PEN field to 1, a read-modify-write operation will be performed to the address specified in the ADDR field. The data read out will be used to calculate a new checksum, the checksum will be XOR:ed with the value in the CB field of this register and then written back to the memory. The data readout is affected by whether or not EDAC is enabled in the LRAMCFG register. At most one of WCB, RCB and XCB should be 1 at the same time. XCB remains set after the operation completes. The ECNT and MECNT fields of LRAMCFG will not be changed by the the operation.

0 Enable wash mode (WASH) - If WASH=1, then the scrubber will write the data in DATA*m* to every memory address in its 16 KiB slice. The wash operation starts from the address in SCRUBCTRL*m* and continues to the last address (0x3FFC). One address is written per clock cycle, regardless of the value in the DELAY field, so performing a wash takes 4096 system clock cycles, if the start address is 0x0000. WASH will self-clear when the operation completes. If DISW=1 then the scrubber will be disabled (SCRUBCTRL*m*.SEN set to 0) after the wash operation completes. If DISW=0, then the scrubber will instead enter scrubbing mode (with delay DELAY system clock cycles between each access) after the wash operation completes.

22 Fault Tolerant PROM/SRAM Memory Interface

22.1 Overview

The fault tolerant 8-bit memory controller (FTMCTRL0) provides a bridge between external memory and the AHB bus. The memory controller can handle two types of devices: PROM, asynchronous static ram (SRAM) The PROM and SRAM areas can be EDAC-protected using a (39,7) BCH code. The BCH code provides single-error correction and double-error detection for each 32-bit memory word.

The memory controller is configured through three configuration registers accessible via an APB bus interface. The PROM and SRAM external data bus is configured in 8-bit mode, for the application requirements.

External chip-selects are provided for up to two PROM bank and four SRAM banks. External PROM are mapped in the address range from 0x01000000 to 0x01FFFFFF and external SRAM in the address range 0x40000000 to 0x4FFFFFFF.

The fault tolerant 8-bit memory controller configuration registers are located on APB bus in the address range from 0x80000000 to 0x80000FFF. See fault tolerant 8-bit memory controller unit connections in the next drawing. The drawing picture memory locations and functions used for fault tolerant 8-bit memory controller configuration and control.

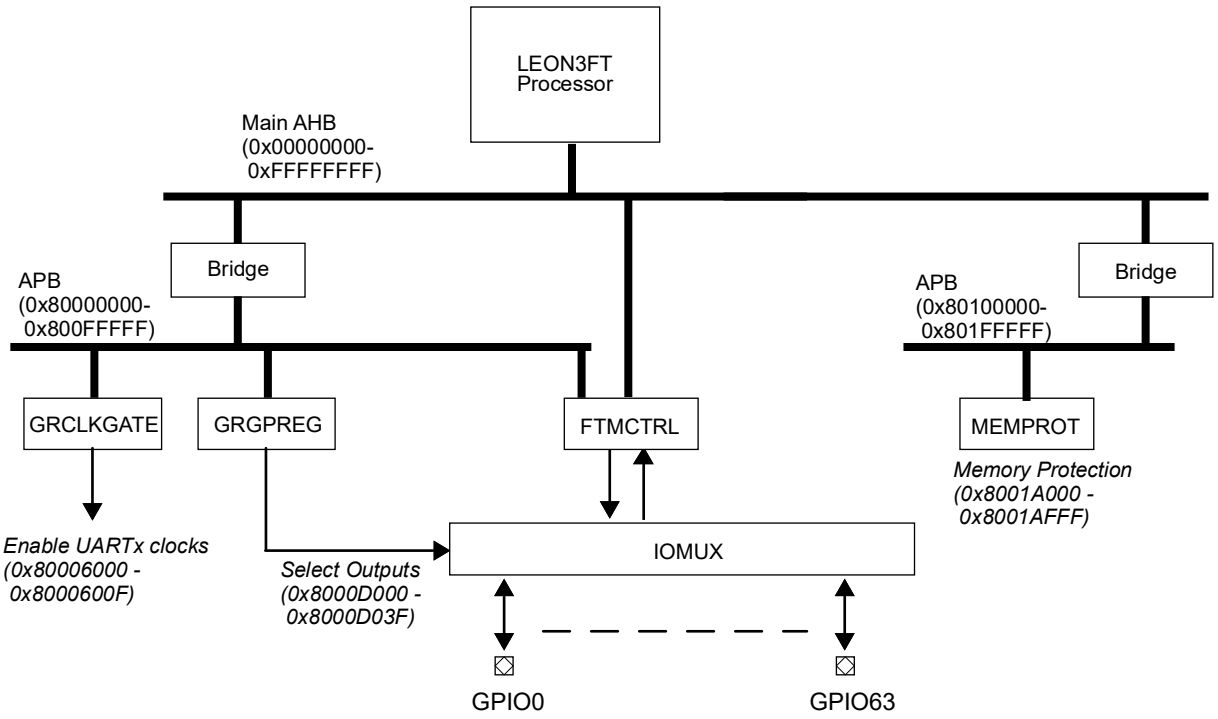


Figure 41. GR716B FTMCTRL bus and pin

The primary clock gating unit **GRCLKGATE** described in section 27 is used to enable/disable the fault tolerant 8-bit memory controller (FTMCTRL0). The unit **GRCLKGATE** can also be used to perform reset of the fault tolerant 8-bit memory controller (FTMCTRL). Software must enable clock and release reset described in section 27 before memory configuration and operations can start.

External IO selection is made in the system IO configuration register (**GRGPREG**) in the address range from 0x8000D000 to 0x8000D03F. See section 7.1 for further information.

The system can be configured to protect and restrict access to the fault tolerant 8-bit memory controller (FTMCTRL0) units in the **MEMPROT** unit. See section 47 for more information.

22.2 PROM access

Two external PROM chip-select signals are provided for the PROM area. The size of the banks can be set in binary steps from 16KiB to 256MiB. If the AHB memory area assigned to the memory controller for PROM accesses is larger than the combined size of the memory banks then the PROM memory area will wrap.

A read access to PROM consists of two data cycles and between 0 and 30 waitstates. The read data (and optional EDAC check-bits) are latched on the rising edge of the clock on the last data cycle. On non-consecutive accesses, a idle cycle is placed between the read cycles to prevent bus contention due to slow turn-off time of PROM devices. Figure 42 shows the basic read cycle waveform (zero waitstate) for non-consecutive PROM reads. Note that the address is undefined in the idle cycle. Figure 43 shows the timing for consecutive cycles (zero waitstate). Waitstates are added by extending the data2 phase. This is shown in figure 44 and applies to both consecutive and non-consecutive cycles. Only an even number of waitstates can be assigned to the PROM area.

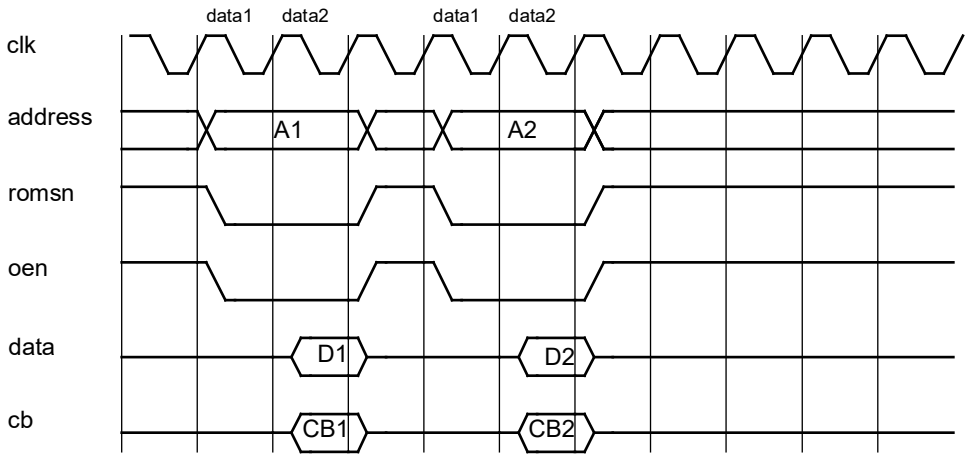


Figure 42. Prom non-consecutive read cycles.

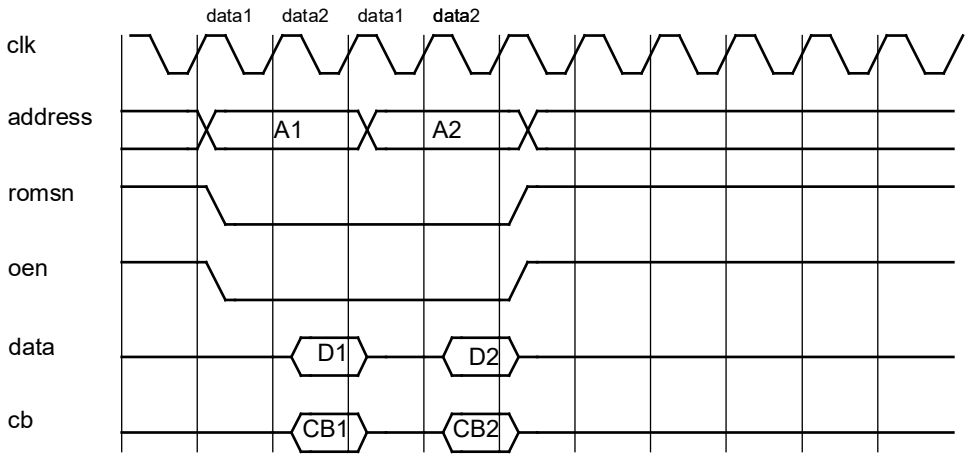


Figure 43. Prom consecutive read cycles.

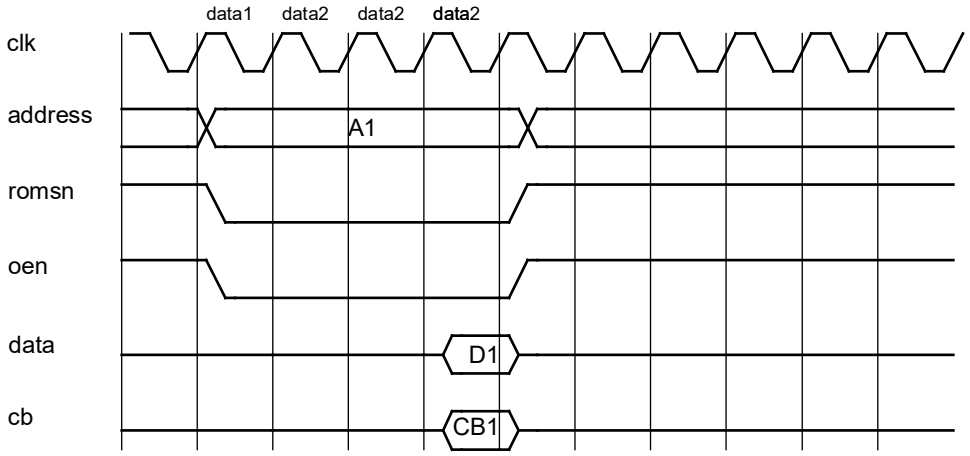


Figure 44. Prom read access with two waitstates.

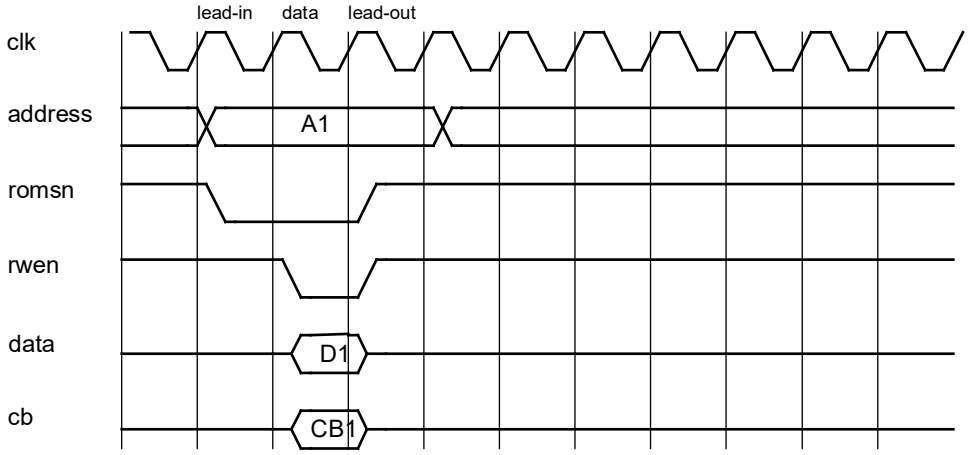


Figure 45. Prom write cycle (0-waitstates)

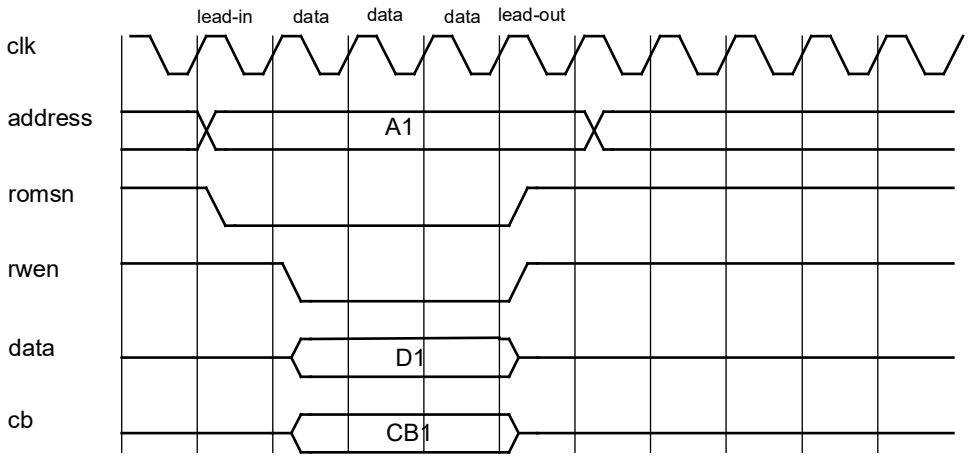


Figure 46. Prom write cycle (2-waitstates)

22.3 SRAM access

The SRAM area is divided on up to four RAM banks. The size of banks is programmed in the RAM bank-size field (MCFG2[12:9]) and can be set in binary steps from 8KiB to 256MiB. A read access to SRAM consists of two data cycles and between zero and three waitstates. The read data (and optional EDAC check-bits) are latched on the rising edge of the clock on the last data cycle. Accesses to RAM bank four can further be stretched by de-asserting BRDYN until the data is available. On non-consecutive accesses, a idle cycle is added after a read cycle to prevent bus contention due to slow turn-off time of memories. Figure 47 shows the basic read cycle waveform (zero waitstate). Waitstates are added in the same way as for PROM in figure 44.

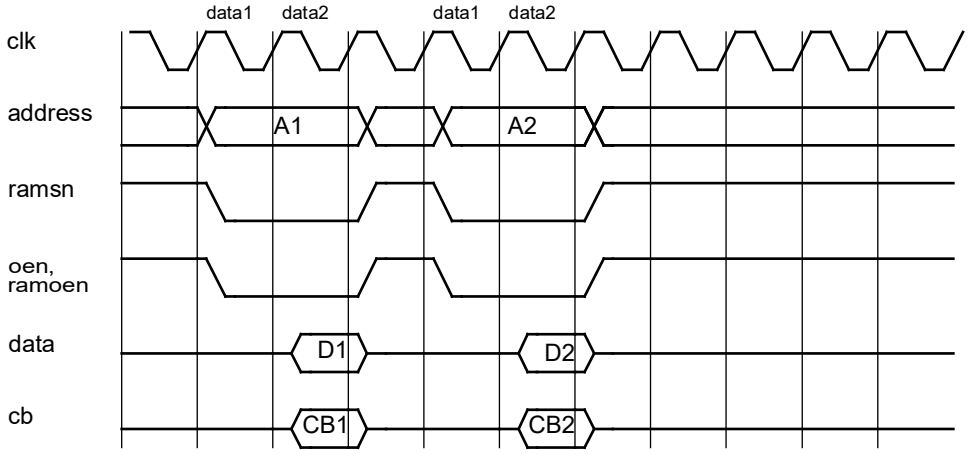


Figure 47. SRAM non-consecutive read cycles.

The SRAM and PROM areas is configured for 8-bit operations. Since reads to memory are always done on 32-bit word basis, read access to 8-bit memory will be transformed in a burst of four read cycles. During writes, only the necessary bytes will be written.

All possible combinations of width, EDAC, and RMW are not supported. The supported combinations are given in table 192, and the behavior of setting an unsupported combination is undefined.

*Table 192.*FTMCTRL supported SRAM and PROM configurations

PROM/SRAM bus width	RWEN resolution (SRAM)	EDAC	RMW bit (SRAM)	Core configuration
8	Bus width	None	0	8-bit support
8	Bus width	BCH	1	8-bit support, EDAC

22.4 Memory EDAC

22.4.1 BCH EDAC

The FTMCTRL is provided with an BCH EDAC that can correct one error and detect two errors in a 32-bit word. For each word, a 7-bit checksum is generated according to the equations below. A correctable error will be handled transparently by the memory controller, but adding one waitstate to the access. If an un-correctable error (double-error) is detected, the current AHB cycle will end with an error response. The EDAC can be used during access to PROM and SRAM areas by setting the corresponding EDAC enable bits in the MCFG3 register. The equations below show how the EDAC check-bits are generated:

```
CB0 = D0 ^ D4 ^ D6 ^ D7 ^ D8 ^ D9 ^ D11 ^ D14 ^ D17 ^ D18 ^ D19 ^ D21 ^ D26 ^ D28 ^ D29 ^ D31
```

```
CB1 = D0 ^ D1 ^ D2 ^ D4 ^ D6 ^ D8 ^ D10 ^ D12 ^ D16 ^ D17 ^ D18 ^ D20 ^ D22 ^ D24 ^ D26 ^ D28
CB2 = D0 ^ D3 ^ D4 ^ D7 ^ D9 ^ D10 ^ D13 ^ D15 ^ D16 ^ D19 ^ D20 ^ D23 ^ D25 ^ D26 ^ D29 ^ D31
CB3 = D0 ^ D1 ^ D5 ^ D6 ^ D7 ^ D11 ^ D12 ^ D13 ^ D16 ^ D17 ^ D21 ^ D22 ^ D23 ^ D27 ^ D28 ^ D29
CB4 = D2 ^ D3 ^ D4 ^ D5 ^ D6 ^ D7 ^ D14 ^ D15 ^ D18 ^ D19 ^ D20 ^ D21 ^ D22 ^ D23 ^ D30 ^ D31
CB5 = D8 ^ D9 ^ D10 ^ D11 ^ D12 ^ D13 ^ D14 ^ D15 ^ D24 ^ D25 ^ D26 ^ D27 ^ D28 ^ D29 ^ D30 ^ D31
CB6 = D0 ^ D1 ^ D2 ^ D3 ^ D4 ^ D5 ^ D6 ^ D7 ^ D24 ^ D25 ^ D26 ^ D27 ^ D28 ^ D29 ^ D30 ^ D31
```

Data is always accessed as words (4 bytes at a time) and the corresponding checkbits are located at the address acquired by inverting the word address (bits 2 to 27) and using it as a byte address. The same chip-select is kept active. A word written as four bytes to addresses 0, 1, 2, 3 will have its checkbits at address 0xFFFFFFF, addresses 4, 5, 6, 7 at 0xFFFFFFE and so on. All the bits up to the maximum bank size will be inverted while the same chip-select is always asserted. This way all the bank sizes can be supported and no memory will be unused (except for a maximum of 4 byte in the gap between the data and checkbit area). A read access will automatically read the four data bytes individually from the nominal addresses and the EDAC checkbit byte from the top part of the bank. A write cycle is performed the same way. Byte or half-word write accesses will result in an automatic read-modify-write access where 4 data bytes and the checkbit byte are firstly read, and then 4 data bytes and the newly calculated checkbit byte are writen back to the memory.

For the ROM the EDAC protection is provided in a similar way as for the SRAM memory described above. The difference is that write accesses are not being handled automatically. Instead, write accesses must only be performed as individual byte accesses by the software, writing one byte at a time, and the corresponding checkbit byte must be calculated and be written to the correct location by the software.

The operation of the EDAC can be tested trough the MCFG3 register. If the WB (write bypass) bit is set, the value in the TCB field will replace the normal checkbits during memory write cycles. If the RB (read bypass) is set, the memory checkbits of the loaded data will be stored in the TCB field during memory read cycles. NOTE: when the EDAC is enabled, the RMW bit in memory configuration register 2 must be set.

22.5 Bus Ready signalling

The BRDYN signal can be used to stretch all types of access cycles to the PROM and the SRAM area. This covers read and write accesses in general, and additionally read-modify-write accesses to the SRAM area. The accesses will always have at least the pre-programmed number of waitstates as defined in memory configuration registers 1 & 2, but will be further stretched until BRDYN is asserted. BRDYN should be asserted in the cycle preceding the last one. If bit 29 in MCFG1 is set, BRDYN can be asserted asynchronously with the system clock. In this case, the read data must be kept stable until the de-assertion of OEN/RAMOEN and BRDYN must be asserted for at least 1.5 clock cycle. It is recommended that BRDYN is asserted until the corresponding chip select signal is de-asserted, to ensure that the access has been properly completed and avoiding the system to stall.

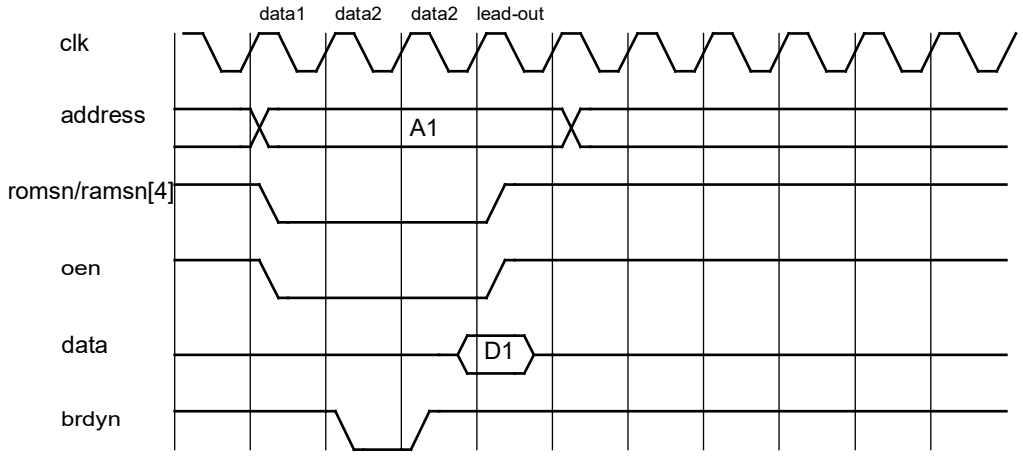


Figure 48. READ cycle with one extra data2 cycle added with BRDYN (synchronous sampling).

Figure 49 shows the use of BRDYN with asynchronous sampling. BRDYN is kept asserted for more than 1.5 clock-cycle. Two synchronization registers are used so it will take at least one additional cycle from when BRDYN is first asserted until it is visible internally. In figure 49 one cycle is added to the data2 phase.

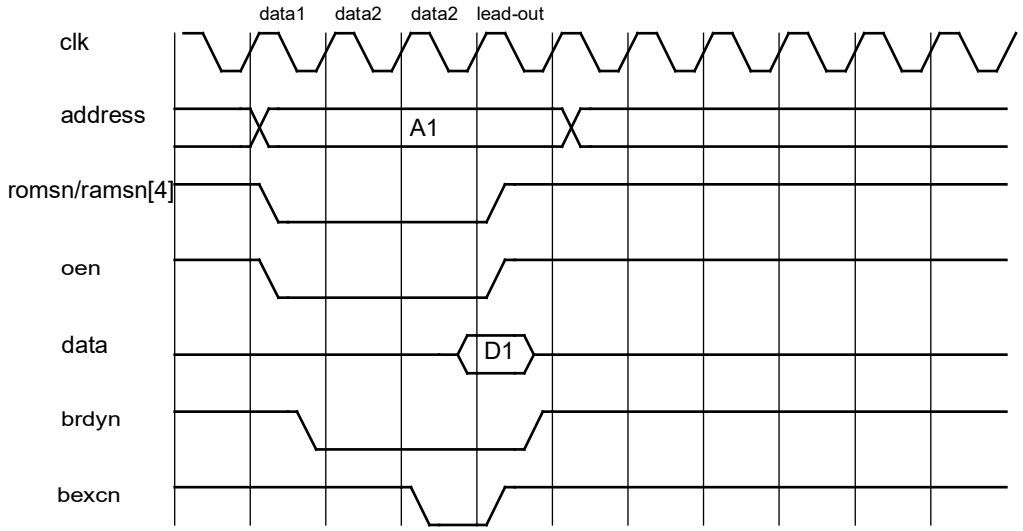


Figure 49. BRDYN (asynchronous) sampling and BEXCN timing.

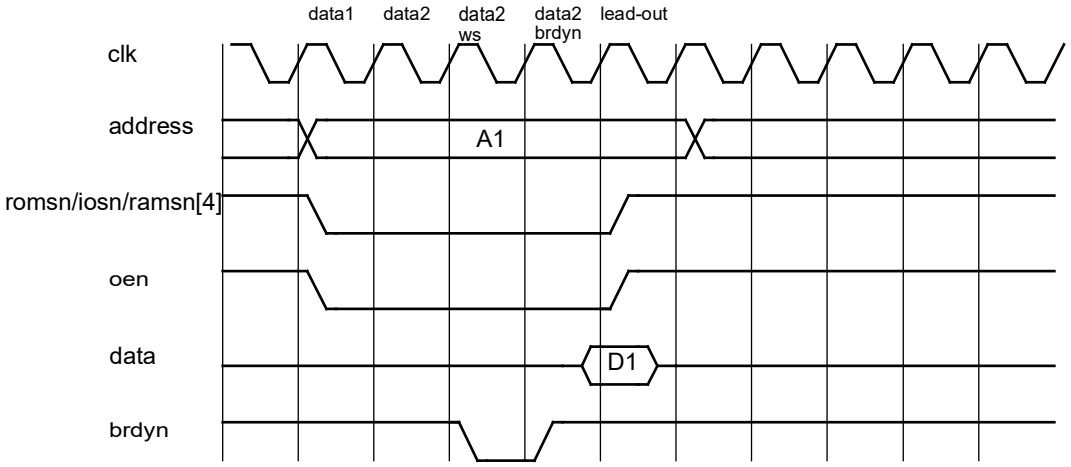


Figure 50. Read cycle with one waitstate (configured) and one BRDYN generated waitstate (synchronous sampling).

If burst accesses and BRDYN signalling are to be used together, special care needs to be taken to make sure BRDYN is raised between the separate accesses of the burst. The controller does not raise the select and OEN signal (in the read case) between accesses during the burst so if BRDYN is kept asserted until the select signal is raised, all remaining accesses in the burst will finish with the configured fixed number of wait states.

22.6 Access errors

An access error can be signalled by asserting the BEXCN signal for read and write accesses. For reads it is sampled together with the read data. For writes it is sampled on the last rising edge before chip select is de-asserted, which is controlled by means of waitstates or bus ready signalling. If the usage of BEXCN is enabled in memory configuration register 1, an error response will be generated on the internal AHB bus. BEXCN can be enabled or disabled through memory configuration register 1, and is active for all areas (PROM and RAM).

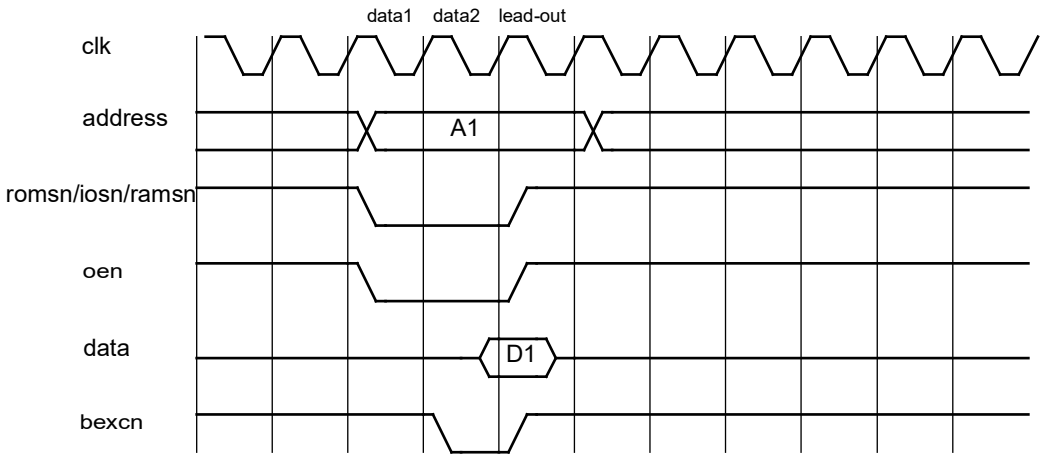


Figure 51. Read cycle with BEXCN.

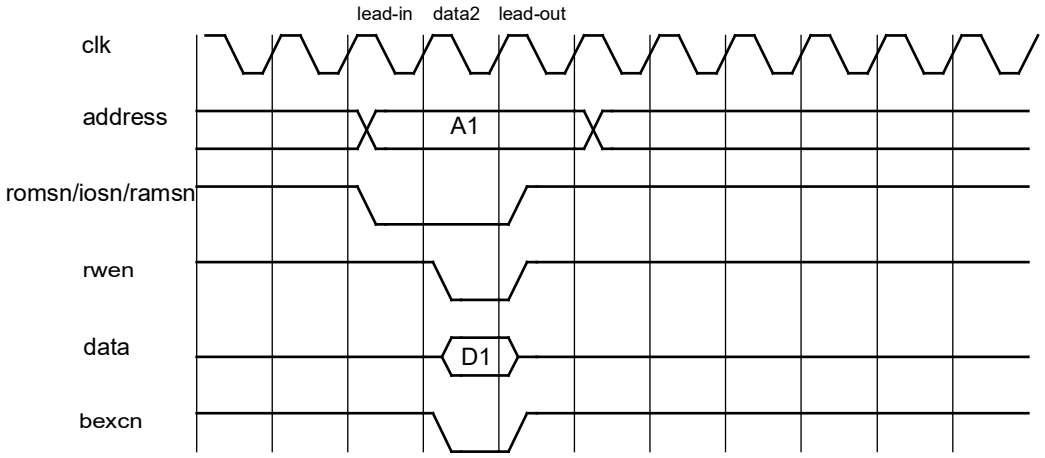


Figure 52. Write cycle with BEXCN. Chip-select (iosn) is not asserted in lead-in cycle for io-accesses.

22.7 Registers

The core is programmed through registers mapped into APB address space.

*Table 193.*FTMCTRL memory controller registers

APB Address offset	Register
0x0	Memory configuration register 1 (MCFG1)
0x4	Memory configuration register 2 (MCFG2)
0x8	Memory configuration register 3 (MCFG3)
0xC	Memory configuration register 4 (MCFG4)
0x10	Memory configuration register 5 (MCFG5)
0x14	Memory configuration register 6 (MCFG6)

22.7.1 Memory configuration register 1 (MCFG1)

Memory configuration register 1 is used to program the timing of rom and IO accesses.

*Table 194.*0x00 - MCFG1 - Memory configuration register 1

31	30	29	28	27	26	25	24	23 20	19	18	17
R	PBRDY	ABRDY	RESERVED		R	BEXCN	R	RESERVED	IOEN	R	ROMBANKSZ
0	0	0	NR		0	0	0	0XF	0	0	0x0
r	rw	rw	rw		rw	rw	r	rw	rw	rw	rw

14	13 12	11	10	9 8	7 4	3 0
ROMANKS7	RESERVED	PWEN	RES	PROM WIDTH	PROM WRITE WS	PROM READ WS
	0	0	0	0	0xF	0xF
rw	r	rw	r	rw	rw	rw

31	RESERVED
30	PROM area bus ready enable (PBRDY) - Enables bus ready (BRDYN) signalling for the PROM area. Reset to ‘0’.
29	Asynchronous bus ready (ABRDY) - Enables asynchronous bus ready.
28 : 27	RESERVED
26	RESERVED
25	Bus error enable (BEXCN) - Enables bus error signalling for all areas. Reset to ‘0’.
24	RESERVED
23 : 20	RESERVED

*Table 194.*0x00 - MCFG1 - Memory configuration register 1

19	I/O enable (IOEN) - Enables accesses to the memory bus I/O area. GR716B doesn't provide any I/O area, i.e. for GR716B this bit field shall always be set to '0'.
18	RESERVED
17: 14	PROM bank size (ROMBANKSZ) - Returns current PROM bank size when read. "0000" is a special case and corresponds to a bank size of 256MiB. All other values give the bank size in binary steps: "0001"=16KiB, "0010"=32KiB, "0011"=64KiB,... , "1111"=256MiB (i.e. 8KIB * 2 ^(ROMBANKSZ)). For value "0000" or "1111" only two chip selects are available. For other values, two chip select signals are available for fixed bank sizes. For other values, four chip select signals are available for programmable bank sizes. Programmable bank sizes can be changed by writing to this register field. The written values correspond to the bank sizes and number of chip-selects as above. Reset to "0000" when programmable.
13:12	RESERVED
11	PROM write enable (PWEN) - Enables write cycles to the PROM area.
10	RESERVED
9 : 8	PROM width (PROM WIDTH) - Sets the data width of the PROM area ("00"=8, "01"=16, "10"=32). For GR716B the data width is locked to 8 bits i.e. for GR716B this bit field shall always be set to "00".
7 : 4	PROM write waitstates (PROM WRITE WS) - Sets the number of wait states for PROM write cycles ("0000"=0, "0001"=2, "0010"=4,..., "1111"=30).
3 : 0	PROM read waitstates (PROM READ WS) - Sets the number of wait states for PROM read cycles ("0000"=0, "0001"=2, "0010"=4,...,"1111"=30). Reset to "1111".

22.7.2 Memory configuration register 2 (MCFG2)

Memory configuration register 2 is used to control the timing of the SRAM.

*Table 195.*0x04 - MLFG2 - Memory configuration register 2

31													16
RESERVED													
15	14	13	12	9	8	7	6	5	4	3	2	1	0
R	R	SI	RAM BANK SIZE		R	RBRDY	RMW	RAM WIDTH		RAM WRITE WS		RAM READ WS	
0	0	0	0x3			0	0	0		3		3	
r	r	rw	rw			rw	rw	rw		rw		rw	

31 : 14	RESERVED
13	SRAM disable (SI) - Disables accesses to SRAM bank if bit 14 (SE) is set to '1'.
12 : 9	RAM bank size (RAM BANK SIZE) - Sets the size of each RAM bank ("0000"=8KiB, "0001"=16KiB, "0010"=32KiB, "0011"= 64KiB,.., "1111"=256MiB)(i.e. (i.e. 8KIB * 2 ^(RAMBANKSZ)).
8	RESERVED
7	RAM bus ready enable (RBRDY) - Enables bus ready signalling for the RAM area.
6	Read-modify-write enable (RMW) - Enables read-modify-write cycles for sub-word writes to 16- bit 32-bit areas with common write strobe (no byte write strobe). Set at reset from external pin.
5 : 4	RAM width (RAM WIDTH) - Sets the data width of the RAM area ("00"=8, "01"=16, "1X"=32). For GR716B the data width is locked to 8 bits i.e. for GR716B this bit field shall always be set to "00".
3 : 2	RAM write waitstates (RAM WRITE WS) - Sets the number of wait states for RAM write cycles ("00"=0, "01"=1, "10"=2, "11"=3).
1 : 0	RAM read waitstates (RAM READ WS) - Sets the number of wait states for RAM read cycles ("00"=0, "01"=1, "10"=2, "11"=3).

22.7.3 Memory configuration register 3 (MCFG3)

MCFG3 contains the control and monitor the memory EDAC.

*Table 196.*0x08 - MCFG3 - Memory configuration register 3

31 28	27	26
RESERVED	ME	RESERVED
0	1	
r	r	

12	11	10	9	8	7 0
RESERVED	WB	RB	RE	PE	TCB
	0	0	*	*	NR
	rw	rw	rw	rw	rw

31 : 28 RESERVED

27 Memory EDAC (ME) - Indicates if memory EDAC is present. (read-only)

26 : 12 RESERVED

11 EDAC diagnostic write bypass (WB) - Enables EDAC write bypass.

10 EDAC diagnostic read bypass (RB) - Enables EDAC read bypass.

9 RAM EDAC enable (RE) - Enable EDAC checking of the RAM area. Set at reset from external pin

8 PROM EDAC enable (PE) - Enable EDAC checking of the PROM area. Set at reset from external pin

7 : 0 Test checkbits (TCB) - This field replaces the normal checkbits during write cycles when WB is set. It is also loaded with the memory checkbits during read cycles when RB is set.

22.7.4 Memory configuration register 4 (MCFG4)

*Table 197.*0x0C - MCFG4 - Memory configuration register 4

31 16
RESERVED

15 0
RESERVED

31 : 16 RESERVED

15 : 0 RESERVED

22.7.5 Memory configuration register 5 (MCFG5)

MCFG5 contains fields to control lead out cycles for the ROM areas.

*Table 198.*0x10 - MCFG5 - Memory configuration register 5

31 30	29 23	22 16
RESERVED	RESERVED	RESERVED

15 14	13 7	6 0
RESERVED	ROMHWS	RESERVED
	0x00	
	rw	

31 : 30 RESERVED

29:23 RESERVED

*Table 198.*0x10 - MCFG5 - Memory configuration register 5

22 : 14	RESERVED
13:7	ROM lead out (ROMHWS) - Lead out cycles added to ROM accesses are ROMHWS(3:0)*$2^{ROMHWS(6:4)}$
6 : 0	RESERVED

22.7.6 Memory configuration register 6 (MCFG6)

MCFG6 contains fields to control lead out cycles for the (S)RAM area.

*Table 199.*0x14 - MCFG6 - Memory configuration register 6

31		16
RESERVED		
0		
r		

15	14	13	7	6	0
RESERVED		RAMHWS		RESERVED	
r		0x00		r	
0		rw		0	

31 : 14	RESERVED
13:7	RAM lead out (RAMHWS) - Lead out cycles added to RAM accesses are RAMHWS(3:0)*$2^{RAMHWS(6:4)}$
6 : 0	RESERVED

23 GR716B Microcontroller Dedicated Memory Interface (NVRAM)

The GR716B microcontroller contains a second memory controller with memory interfaces on dedicated pins, these pins are not IO multiplexed (not pin shared with GPIO interfaces).

On the GR716B-PBGA with 400 pin package contains these memory interfaces on dedicated pins. The memory interfaces can be used to interface with PROM/SRAM/MRAM external memories. The data lines on this interface are 16 bits wide compared to the 8-bit wide memory interface (which is also pin multiplexed) on the CQFP package.

On the GR716B-SIP with 400 pin package the memory interface is connected with an MRAM of size 32 Mbit inside the package.The interface width is 16 bit. These interfaces are not available for external functional use.

These interfaces are not available in the GR716B-CQFP due to unavailability of pins in the CQFP132 pin package.

Table 200. GR716B second memory controller usability

Features	GR716B-PBGA	GR716B-SIP	GR716B-CQFP
Dedicated second memory interface	Yes, available	Connected to MRAM of size 32 Mbit inside the package.	Not available
Interface with external memory	Yes, possible	MRAM available inside the package	Second memory interface not available
8/16 bit interface	Both possible	MRAM interfaced with 16 bit data lines.	Not available
Boot using 16-bit interface without EDAC	Yes, possible	Yes, possible	Not possible
Interface external memory with 8-bit data width	Yes, possible	Not possible	Not possible
Boot using 8-bit memory interface with or without EDAC	Not possible	Not possible	Not possible
All devices has an 8-bit memory controller with memory interface pin multiplexed with the GPIO, this controller is described at section 22. It is possible to boot with EDAC enabled from this controller. Also, all the devices has SPI memory controller which is described at section 46.			

The interface and the configuration (boot options) are described in table 201.

*Table 201.*GR716B dedicated memory interface (from second memory controller)

Memory interfaces	No of pins	Function	Input/output
MEM1_CONFIG	3	Boot settings (see table below)	Input
MEM1_RSTN	1	Reset drive	Output
MEM1_CSN	6	Chip select	Output
MEM1_ADDR	23	Address line	Output
MEM1_OEN	1	Output enable	Output
MEM1_WEN	1	Write enable	Output
MEM1_DATA	16	Data line	Inout
Total	**51**		

The second memory controller can address 128 Mbit of parallel memory. The boot configuration option available for the GR716B (section 3.1) works along with the configuration option provided for the second memory controller. The second memory controller boot descriptions are described below.

*Table 202.*GR716B second memory controller bootstrap pins

Boot options	Functions
MEM1_CONFIG(0)	Boot enable, GR716B boots using the second memory controller with dedicated memory interface (memory interfaces not shared with the GPIO). Drive 'Low' to enable boot, 'High' to disable.
MEM1_CONFIG(1)	Memory enable, enable the second memory controller. Drive 'Low' to enable memory interface, 'High' to disable.
MEM1_CONFIG(2)	Drive 'Low' for 16-bit Memory interface. 'High' is for future use.

The second memory controller is described in the following section of this document, it is similar to the first memory controller but with 16 bit wide data interface.

23.1 Overview

The fault tolerant 8/16bit memory controller (FTMCTRL1) provides a bridge between external memory and the AHB bus. The memory controller can handle two types of devices: PROM, asynchronous static ram (SRAM). The PROM and SRAM areas can be EDAC-protected using a (39,7) BCH code (in 8-bit mode). The BCH code provides single-error correction and double-error detection for each 32-bit memory word.

The memory controller is configured through configuration registers accessible via an APB bus interface. The PROM and SRAM external data bus can be configured in 8/16 bit mode, for the application requirements.

External chip-selects are provided for up to two PROM bank and four SRAM banks. External PROM are mapped in the address range from 0x51000000 to 0x51FFFFFF and external SRAM in the address range 0x50000000 to 0x50FFFFFF.

The fault tolerant 8/16-bit memory controller configuration registers are located on APB bus in the address range from 0x80307000 to 0x80307100. See 8/16-bit memory controller unit connections in the next drawing. The drawing picture memory locations and functions used for memory controller configuration and control.

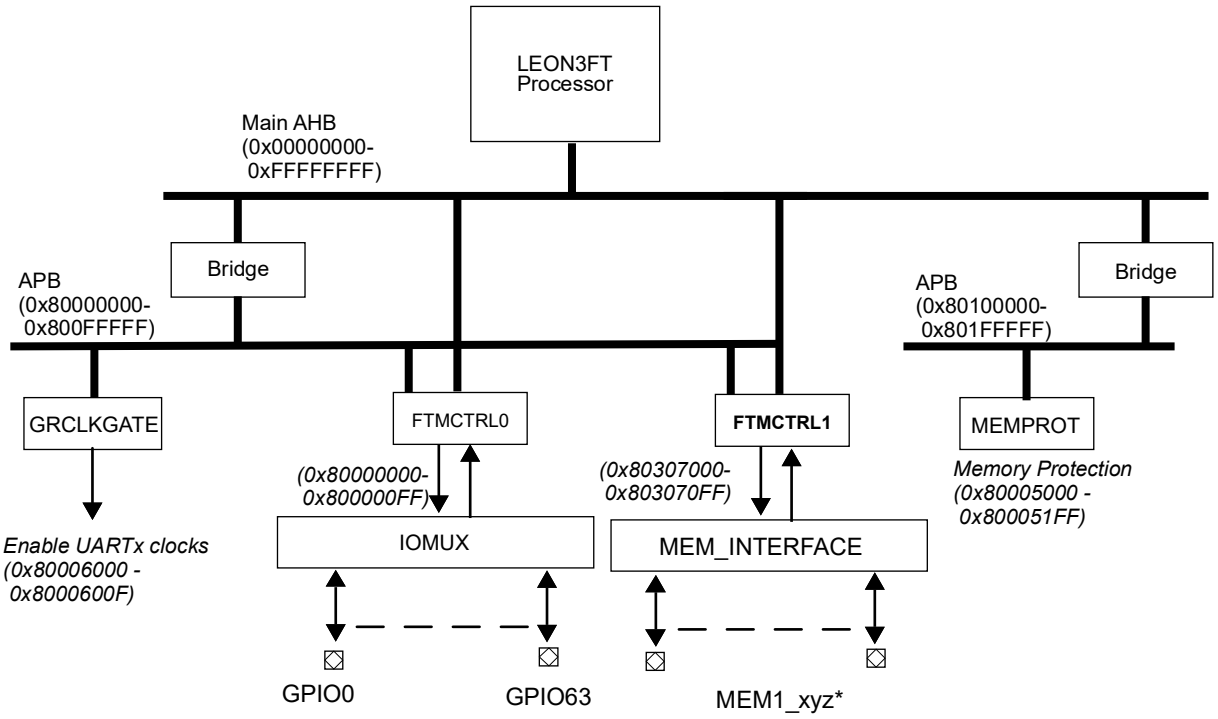


Figure 53. GR716B FTMCTRL bus and pin

The primary clock gating unit **GRCLKGATE** described in section 27 is used to enable/disable the FTMCTRL. The unit **GRCLKGATE** can also be used to perform reset of the FTMCTRL1. Software must enable clock and release reset described in section 27 before memory configuration and operations can start.

The system can be configured to protect and restrict access to the FTMCTRL1 units in the **MEMPROT** unit. See section 47 for more information.

23.2 PROM access

Four external PROM chip-select signals are provided for the PROM area. The size of the banks can be set in binary steps from 16KiB to 16MiB. If the AHB memory area assigned to the memory controller for PROM accesses is larger than the combined size of the memory banks then the PROM memory area will wrap.

A read access to PROM consists of two data cycles and between 0 and 30 waitstates. The read data (and optional EDAC check-bits) are latched on the rising edge of the clock on the last data cycle. On non-consecutive accesses, a idle cycle is placed between the read cycles to prevent bus contention due to slow turn-off time of PROM devices. Figure 54 shows the basic read cycle waveform (zero waitstate) for non-consecutive PROM reads. Note that the address is undefined in the idle cycle. Figure 55 shows the timing for consecutive cycles (zero waitstate). Waitstates are added by extending the data2 phase. This is shown in figure 56 and applies to both consecutive and non-consecutive cycles. Only an even number of waitstates can be assigned to the PROM area.

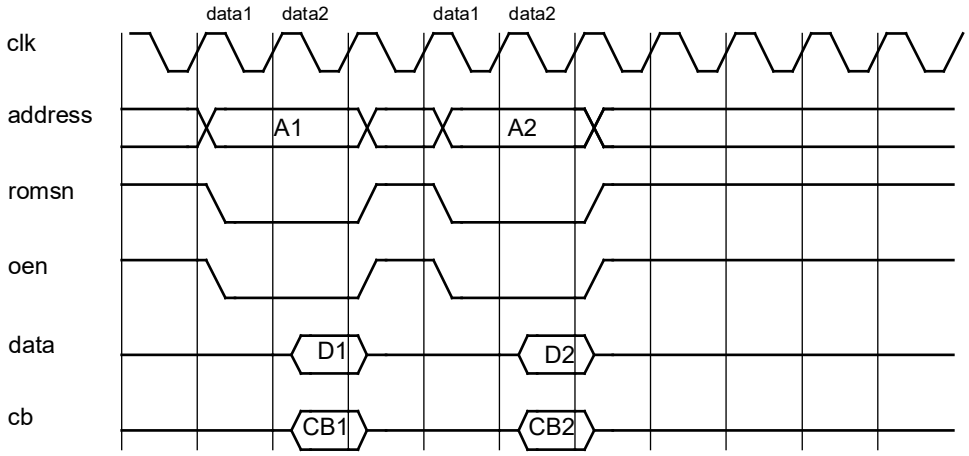


Figure 54. Prom non-consecutive read cycles.

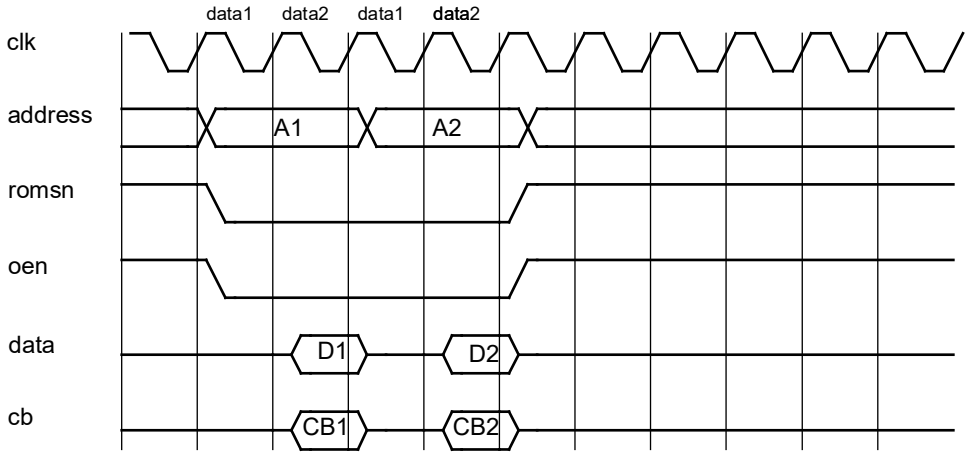


Figure 55. Prom consecutive read cycles.

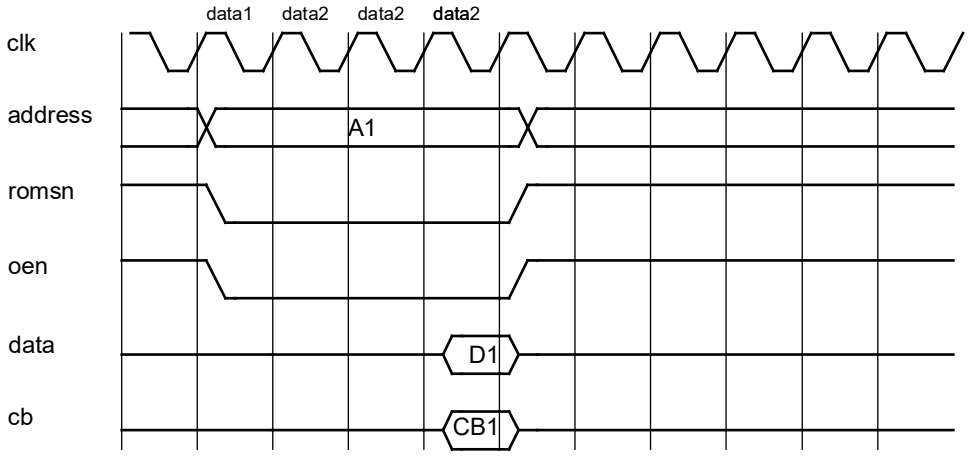


Figure 56. Prom read access with two waitstates.

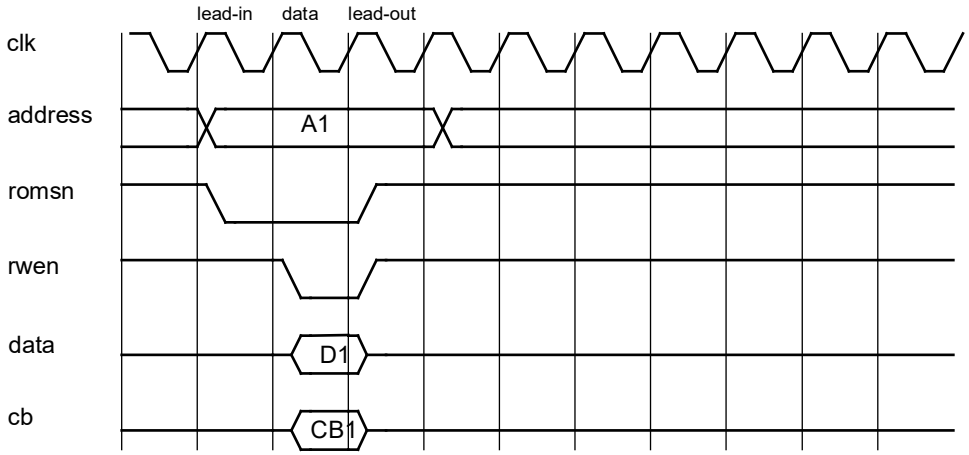


Figure 57. Prom write cycle (0-waitstates)

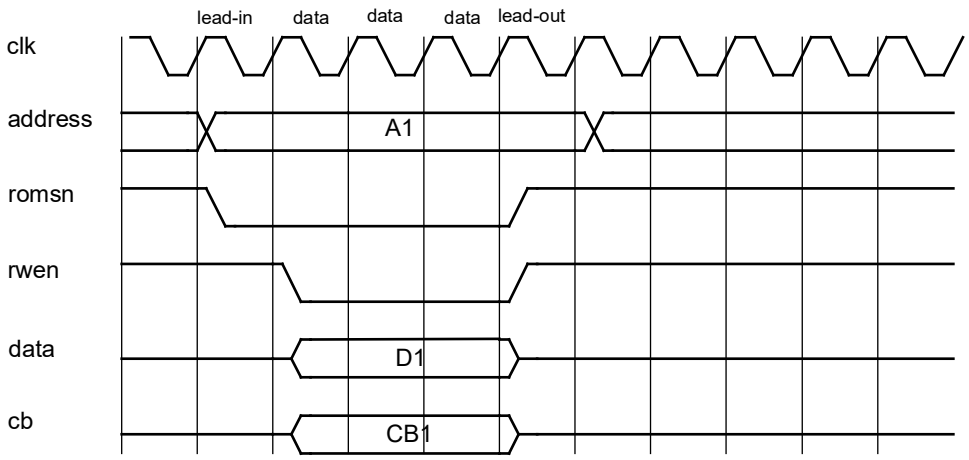


Figure 58. Prom write cycle (2-waitstates)

23.3 SRAM access

The SRAM area is divided on up to four RAM banks. The size of banks is programmed in the RAM bank-size field (MCFG2[12:9]) and can be set in binary steps from 8KiB to 16MiB. A read access to SRAM consists of two data cycles and between zero and three waitstates. The read data (and optional EDAC check-bits) are latched on the rising edge of the clock on the last data cycle. On non-consecutive accesses, a idle cycle is added after a read cycle to prevent bus contention due to slow turn-off

time of memories. Figure 59 shows the basic read cycle waveform (zero waitstate). Waitstates are added in the same way as for PROM in figure 56.

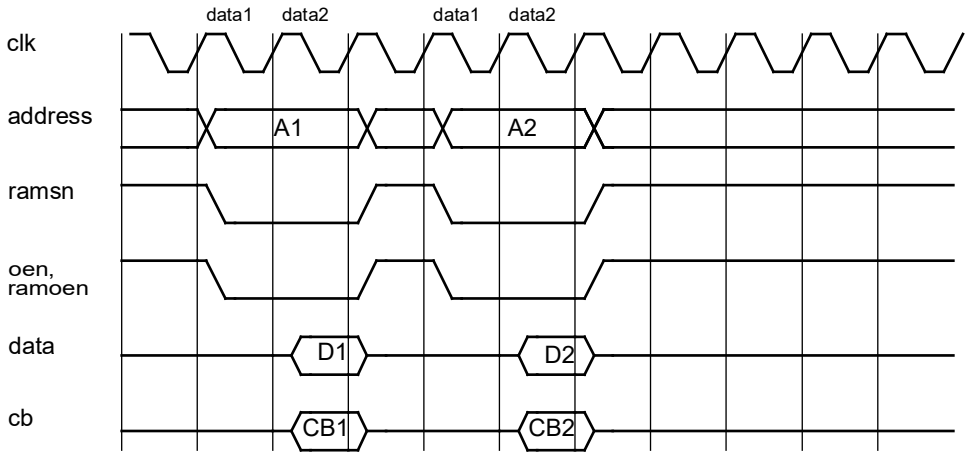


Figure 59. SRAM non-consecutive read cycles.

23.4 8-bit and 16-bit PROM and SRAM access

The SRAM and PROM areas can be configured for 8/16-bit operations by programming the ROM and RAM width fields in the memory configuration registers. Since reads to memory are always done on 32-bit word basis, read access to 8-bit memory will be transformed in a burst of four read cycles while access to 16-bit memory will generate a burst of two 16-bit reads. During writes, only the necessary bytes will be written.

All possible combinations of width, EDAC, and RMW are not supported. The supported combinations are given in table 203, and the behavior of setting an unsupported combination is undefined.

*Table 203.*FTMCTRL1 supported SRAM and PROM configurations

PROM/SRAM bus width	RWEN resolution (SRAM)	EDAC	RMW bit (SRAM)	Core configuration
8	Bus width	None	0	8-bit support
8	Bus width	BCH	1	8-bit support, EDAC
16	Byte	None	0	16-bit support
16	Bus width	None	1	16-bit support

In 8-bit mode, the PROM/SRAM devices should be connected to the MSB byte of the data bus (MEM1_DATA[15:8]). The LSB address bus should be used for addressing (MEM1_ADDR[22:0]). In 16-bit mode, MEM1_DATA[15:0] should be used as data bus, and MEM1_ADDR[22:1] as address bus.

23.5 Memory EDAC

23.5.1 BCH EDAC

The FTMCTRL is provided with an BCH EDAC that can correct one error and detect two errors in a 32-bit word. For each word, a 7-bit checksum is generated according to the equations below. A correctable error will be handled transparently by the memory controller, but adding one waitstate to the access. If an un-correctable error (double-error) is detected, the current AHB cycle will end with an error response. The EDAC can be used during access to PROM and SRAM areas by setting the corre-

sponding EDAC enable bits in the MCFG3 register. The equations below show how the EDAC checkbits are generated:

```
CB0 = D0 ^ D4 ^ D6 ^ D7 ^ D8 ^ D9 ^ D11 ^ D14 ^ D17 ^ D18 ^ D19 ^ D21 ^ D26 ^ D28 ^ D29 ^ D31
CB1 = D0 ^ D1 ^ D2 ^ D4 ^ D6 ^ D8 ^ D10 ^ D12 ^ D16 ^ D17 ^ D18 ^ D20 ^ D22 ^ D24 ^ D26 ^ D28
CB2 = D0 ^ D3 ^ D4 ^ D7 ^ D9 ^ D10 ^ D13 ^ D15 ^ D16 ^ D19 ^ D20 ^ D23 ^ D25 ^ D26 ^ D29 ^ D31
CB3 = D0 ^ D1 ^ D5 ^ D6 ^ D7 ^ D11 ^ D12 ^ D13 ^ D16 ^ D17 ^ D21 ^ D22 ^ D23 ^ D27 ^ D28 ^ D29
CB4 = D2 ^ D3 ^ D4 ^ D5 ^ D6 ^ D7 ^ D14 ^ D15 ^ D18 ^ D19 ^ D20 ^ D21 ^ D22 ^ D23 ^ D30 ^ D31
CB5 = D8 ^ D9 ^ D10 ^ D11 ^ D12 ^ D13 ^ D14 ^ D15 ^ D24 ^ D25 ^ D26 ^ D27 ^ D28 ^ D29 ^ D30 ^ D31
CB6 = D0 ^ D1 ^ D2 ^ D3 ^ D4 ^ D5 ^ D6 ^ D7 ^ D24 ^ D25 ^ D26 ^ D27 ^ D28 ^ D29 ^ D30 ^ D31
```

If the SRAM is configured in 8-bit mode data is always accessed as words (4 bytes at a time) and the corresponding checkbits are located at the address acquired by inverting the word address (bits 2 to 27) and using it as a byte address. The same chip-select is kept active. A word written as four bytes to addresses 0, 1, 2, 3 will have its checkbits at address 0xFFFFFFF, addresses 4, 5, 6, 7 at 0xFFFFFFE and so on. All the bits up to the maximum bank size will be inverted while the same chip-select is always asserted. This way all the bank sizes can be supported and no memory will be unused (except for a maximum of 4 byte in the gap between the data and checkbit area). A read access will automatically read the four data bytes individually from the nominal addresses and the EDAC checkbit byte from the top part of the bank. A write cycle is performed the same way. Byte or half-word write accesses will result in an automatic read-modify-write access where 4 data bytes and the checkbit byte are firstly read, and then 4 data bytes and the newly calculated checkbit byte are writen back to the memory.

If the ROM is configured in 8-bit mode, the EDAC protection is provided in a similar way as for the SRAM memory described above. The difference is that write accesses are not being handled automatically. Instead, write accesses must only be performed as individual byte accesses by the software, writing one byte at a time, and the corresponding checkbit byte must be calculated and be written to the correct location by the software.

NOTE: Refer [FTMCTRL] to implement the EDAC in 8-bit bus mode.

The operation of the EDAC can be tested trough the MCFG3 register. If the WB (write bypass) bit is set, the value in the TCB field will replace the normal checkbits during memory write cycles. If the RB (read bypass) is set, the memory checkbits of the loaded data will be stored in the TCB field during memory read cycles. NOTE: when the EDAC is enabled, the RMW bit in memory configuration register 2 must be set.

23.5.2 EDAC Error reporting

As mentioned above an un-correctable error results in an AHB error response which can be monitored on the bus. Correctable errors however are handled transparently and are not visible on the AHB bus. A sideband signal is provided which is asserted during one clock cycle for each access for which a correctable error is detected. This can be used for providing an external scrubbing mechanism and/or statistics.

23.6 Registers

The core is programmed through registers mapped into APB address space.

*Table 204.*FTMCTRL memory controller registers

APB Address offset	Register
0x0	Memory configuration register 1 (MCFG1)
0x4	Memory configuration register 2 (MCFG2)
0x8	Memory configuration register 3 (MCFG3)
0xC	Memory configuration register 4 (MCFG4)
0x10	Memory configuration register 5 (MCFG5)
0x14	Memory configuration register 6 (MCFG6)

23.6.1 Memory configuration register 1 (MCFG1)

Memory configuration register 1 is used to program the timing of rom and IO accesses.

*Table 205.*0x00 - MCFG1 - Memory configuration register 1

31	30	29	28	27	26	25	24	23 20	19	18	17
R	R	R	RESERVED		R	BEXCN	R	RESERVED	IOEN	R	ROMBANKSZ
0	0	0	NR		0	0	0	0XF	0	0	0x0
r	r	r	rw		rw	rw	r	rw	rw	rw	rw

14	13 12	11	10	9 8	7 4	3 0
ROMANKS7	RESERVED	PWEN	RES	PROM WIDTH	PROM WRITE WS	PROM READ WS
	0	0	0	0	0xF	0xF
rw	r	rw	r	rw	rw	rw

31	RESERVED
30	RESERVED
29	RESERVED
28 : 27	RESERVED
26	RESERVED
25	Bus error enable (BEXCN) - Enables bus error signalling for all areas. Reset to ‘0’.
24	RESERVED
23 : 20	RESERVED
19	I/O enable (IOEN) - Enables accesses to the memory bus I/O area. GR716B doesn’t provide any I/O area, i.e. for GR716B this bit field shall always be set to ’0’.
18	RESERVED
17: 14	PROM bank size (ROMBANKSZ) - Returns current PROM bank size when read. “0000” is a special case and corresponds to a bank size of 16MiB. All other values give the bank size in binary steps: “0001”=16KiB, “0010”=32KiB, “0011”=64KiB,... , “1111”=256MiB (i.e. 8KIB * 2 ^(ROM-BANKSZ)). For value “0000” or “1111” only two chip selects are available. For other values, two chip select signals are available for fixed bank sizes. For other values, four chip select signals are available for programmable bank sizes. Programmable bank sizes can be changed by writing to this register field. The written values correspond to the bank sizes and number of chip-selects as above. Reset to “0000” when programmable.
13:12	RESERVED
11	PROM write enable (PWEN) - Enables write cycles to the PROM area.
10	RESERVED
9 : 8	PROM width (PROM WIDTH) - Sets the data width of the PROM area (“00”=8, “01”=16, “10”=32). For this second memory controller the data width can be 8 or 16 bit wide.

*Table 205.*0x00 - MCFG1 - Memory configuration register 1

7 : 4 PROM write waitstates (PROM WRITE WS) - Sets the number of wait states for PROM write cycles (“0000”=0, “0001”=2, “0010”=4,..., “1111”=30).

3 : 0 PROM read waitstates (PROM READ WS) - Sets the number of wait states for PROM read cycles (“0000”=0, “0001”=2, “0010”=4,...,”1111”=30). Reset to “1111”.

23.6.2 Memory configuration register 2 (MCFG2)

Memory configuration register 2 is used to control the timing of the SRAM.

*Table 206.*0x04 - MLFG2 - Memory configuration register 2

31 16
RESERVED

15	14	13	12 9	8	7	6	5 4	3 2	1 0
R	R	SI	RAM BANK SIZE	R	RBRDY	RMW	RAM WIDTH	RAM WRITE WS	RAM READ WS
0	0	0	0x3		0	0	0	3	3
r	r	rw	rw		rw	rw	rw	rw	rw

31 : 14 RESERVED

13 SRAM disable (SI) - Disables accesses to SRAM bank if bit 14 (SE) is set to ‘1’.

12 : 9 RAM bank size (RAM BANK SIZE) - Sets the size of each RAM bank (“0000”=8KiB, “0001”=16KiB, “0010”=32KiB, “0011”= 64KiB,.., “1111”=256MiB)(i.e. (i.e. 8KIB * 2 ^(RAM-BANKSZ)).

8 RESERVED

7 RAM bus ready enable (RBRDY) - Enables bus ready signalling for the RAM area.

6 Read-modify-write enable (RMW) - Enables read-modify-write cycles for sub-word writes to 16- bit 32-bit areas with common write strobe (no byte write strobe). Set at reset from external pin.

5 : 4 RAM width (RAM WIDTH) - Sets the data width of the RAM area (“00”=8, “01”=16, “1X”=32). For this second memory controller the data width can be 8 or 16 bit wide.

3 : 2 RAM write waitstates (RAM WRITE WS) - Sets the number of wait states for RAM write cycles (“00”=0, “01”=1, “10”=2, “11”=3).

1 : 0 RAM read waitstates (RAM READ WS) - Sets the number of wait states for RAM read cycles (“00”=0, “01”=1, “10”=2, “11”=3).

23.6.3 Memory configuration register 3 (MCFG3)

MCFG3 contains the control and monitor the memory EDAC.

*Table 207.*0x08 - MCFG3 - Memory configuration register 3

31 28	27	26
RESERVED	ME	RESERVED
0	1	
r	r	

12	11	10	9	8	7 0
RESERVED	WB	RB	RE	PE	TCB
	0	0	*	*	NR
	rw	rw	rw	rw	rw

31 : 28 RESERVED

27 Memory EDAC (ME) - Indicates if memory EDAC is present. (read-only)

26 : 12 RESERVED

11 EDAC diagnostic write bypass (WB) - Enables EDAC write bypass.

10 EDAC diagnostic read bypass (RB) - Enables EDAC read bypass.

9 RAM EDAC enable (RE) - Enable EDAC checking of the RAM area. Set at reset from external pin

*Table 207.*0x08 - MCFG3 - Memory configuration register 3

8	PROM EDAC enable (PE) - Enable EDAC checking of the PROM area. Set at reset from external pin
7 : 0	Test checkbits (TCB) - This field replaces the normal checkbits during write cycles when WB is set. It is also loaded with the memory checkbits during read cycles when RB is set.

23.6.4 Memory configuration register 4 (MCFG4)

*Table 208.*0x0C - MCFG4 - Memory configuration register 4

31	16
RESERVED	

15	0
RESERVED	

31 : 16	RESERVED
15 : 0	RESERVED

23.6.5 Memory configuration register 5 (MCFG5)

MCFG5 contains fields to control lead out cycles for the ROM areas.

*Table 209.*0x10 - MCFG5 - Memory configuration register 5

31 30	29 23	22 16
RESERVED	RESERVED	RESERVED

15 14	13 7	6 0
RESERVED	ROMHWS	RESERVED
	0x00	
	rw	

31 : 30	RESERVED
29:23	RESERVED
22 : 14	RESERVED
13:7	ROM lead out (ROMHWS) - Lead out cycles added to ROM accesses are ROMHWS(3:0)*$2^{ROMHWS(6:4)}$
6 : 0	RESERVED

23.6.6 Memory configuration register 6 (MCFG6)

MCFG6 contains fields to control lead out cycles for the (S)RAM area.

*Table 210.*0x14 - MCFG6 - Memory configuration register 6

31	16
RESERVED	
0	
r	

15 14	13 7	6 0
RESERVED	RAMHWS	RESERVED
r	0x00	r
0	rw	0

*Table 210.*0x14 - MCFG6 - Memory configuration register 6

31 : 14	RESERVED
13:7	RAM lead out (RAMHWS) - Lead out cycles added to RAM accesses are RAMHWS(3:0)*$2^{RAMHWS(6:4)}$
6 : 0	RESERVED

24 MIL-STD-1553B / AS15531 Interface

The GR716B microcontroller comprises a MIL-STD-1553B / AS15531 Interface (GR1553B) unit. The MIL-STD-1553B / AS15531 Interface (GR1553B) unit controls its own external pins and has a unique AMBA address described in chapter 2.10.

The MIL-STD-1553B / AS15531 Interface (GR1553B) unit is located on APB bus in the address range from 0x80101000 to 0x80101FFF. See GR1553B unit connections in the next drawing. The drawing picture memory locations and functions used for GR1553B configuration and control.

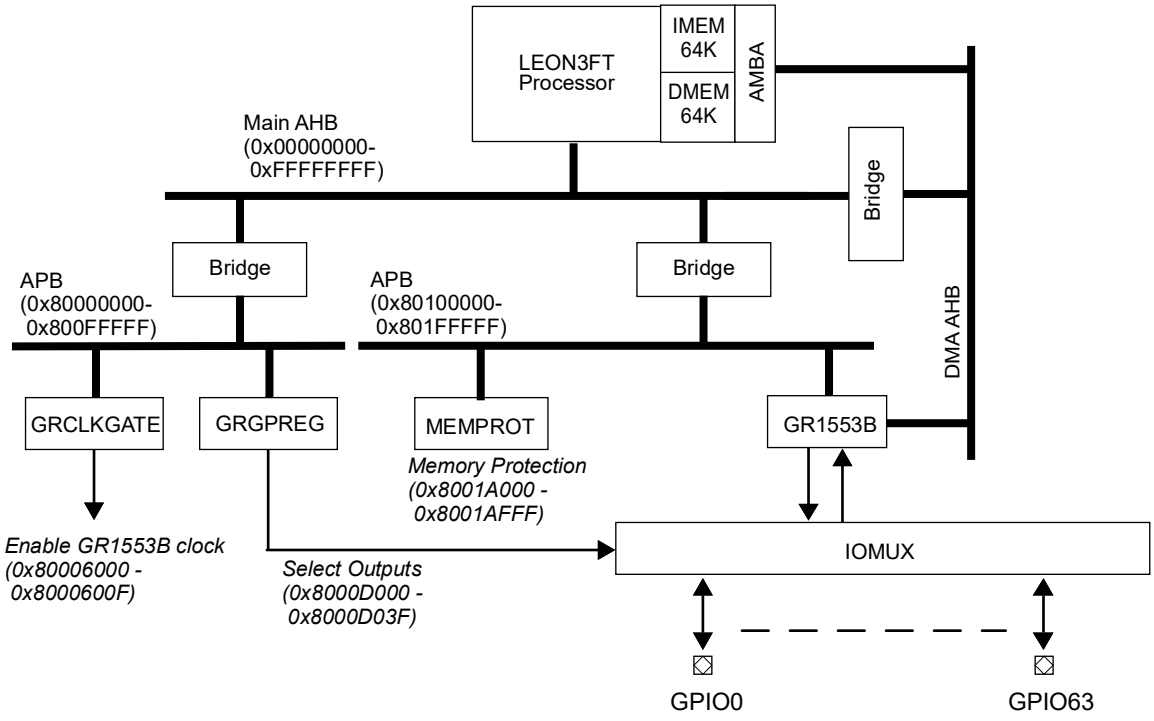


Figure 60. GR716B GR1553B bus and pin connection

The primary clock gating unit **GRCLKGATE** described in section 27 is used to enable/disable the GR1553B unit. The unit **GRCLKGATE** can also be used to perform reset of the GR1553B unit. Software must enable clock and release reset described in section 27 before GR1553B configuration and transmission can start.

External IO selection per GR1553B unit is made in the system IO configuration register (**GRGPREG**) in the address range from 0x8000D000 to 0x8000D03F. See section 7.1 for further information.

The **GR1553B** unit controls its own external pins and has a unique AMBA address described in chapter 2.10. Configuration and status registers are described in section 24.7.

The system can be configured to protect and restrict access to GR1553B in the **MEMPROT** unit. See section 47 for more information.

24.1 Overview

This interface core connects the AMBA AHB/APB bus to a single- or dual redundant MIL-STD-1553B bus, and can act as either Bus Controller, Remote Terminal or Bus Monitor.

MIL-STD-1553B (and derived standard SAE AS15531) is a bus standard for transferring data between up to 32 devices over a shared (typically dual-redundant) differential wire. The bus is designed for predictable real-time behavior and fault-tolerance. The raw bus data rate is fixed at 1 Mbit/s, giving a maximum of around 770 kbit/s payload data rate.

One of the terminals on the bus is the Bus Controller (BC), which controls all traffic on the bus. The other terminals are Remote Terminals (RTs), which act on commands issued by the bus controller. Each RT is assigned a unique address between 0-30. In addition, the bus may have passive Bus Monitors (BM:s) connected.

There are 5 possible data transfer types on the MIL-STD-1553 bus:

- BC-to-RT transfer (“receive”)
- RT-to-BC transfer (“transmit”)
- RT-to-RT transfer
- Broadcast BC-to-RTs
- Broadcast RT-to-RTs

Each transfer can contain 1-32 data words of 16 bits each.

The bus controller can also send “mode codes” to the RTs to perform administrative tasks such as time synchronization, and reading out terminal status.

24.2 Electrical interface

The core is connected to the MIL-STD-1553B bus wire through single or dual transceivers, isolation transformers and transformer or stub couplers as shown in figure 61. If single-redundancy is used, the unused bus receive P/N signals should be tied both-high or both-low. The transmitter enables are typically inverted and therefore called transmitter inibit (txinh).

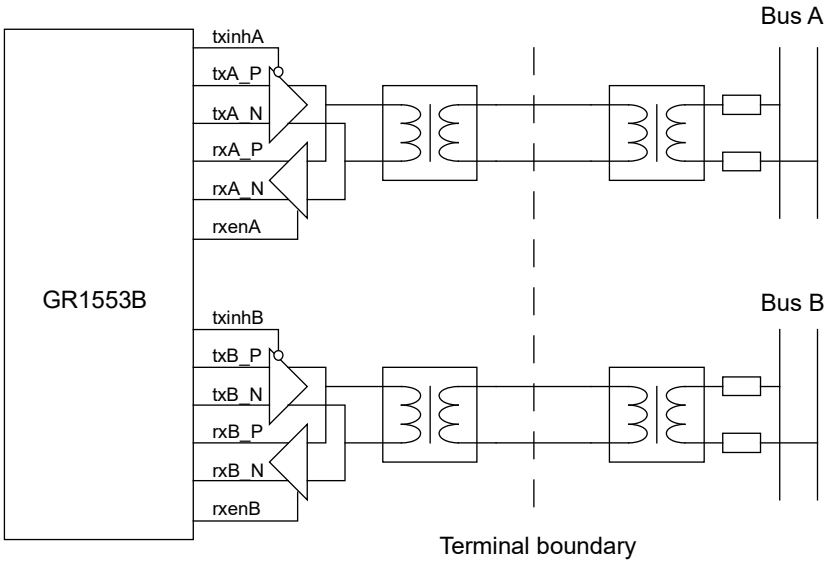


Figure 61. Interface between core and MIL-STD-1553B bus (dual-redundant, transformer coupled)

24.3 Operation

24.3.1 Operating modes

The core contains three separate control units for the Bus Controller, Remote Terminal and Bus Monitor handling, with a shared 1553 codec.

The operating mode of the core is controlled by starting and stopping of the BC/RT/BM units via register writes. At start-up, none of the parts are enabled, and the core is completely passive on both the 1553 and AMBA bus.

The BC and RT parts of the core can not be active on the 1553 bus at the same time. While the BC is running or suspended, only the BC (and possibly BM) has access to the 1553 bus, and the RT can only receive and respond to commands when both the BC schedules are completely stopped (not running or even suspended).

The Bus Monitor, however, is only listening on the codec receivers and can therefore operate regardless of the enabled/disabled state of the other two parts.

24.3.2 Register interface

The core is configured and controlled through control registers accessed over the APB bus. Each of the BC,RT,BM parts has a separate set of registers, plus there is a small set of shared registers.

Some of the control register fields for the BC and RT are protected using a ‘key’, a field in the same register that has to be written with a certain value for the write to take effect. The purpose of the keys are to give RT/BM designers a way to ensure that the software can not interfere with the bus traffic by enabling the BC or changing the RT address. If the software is built without knowledge of the key to a certain register, it is very unlikely that it will accidentally perform a write with the correct key to that control register.

24.3.3 Interrupting

The core has one interrupt output, which can be generated from several different source events. Which events should cause an interrupt can be controlled through the IRQ Enable Mask register.

24.3.4 MIL-STD-1553 Codec

The core’s internal codec receives and transmits data words on the 1553 bus, and generates and checks sync patterns and parity.

Loop-back checking logic checks that each transmitted word is also seen on the receive inputs. If the transmitted word is not echoed back, the transmitter stops and signals an error condition, which is then reported back to the user.

24.4 Bus Controller Operation

24.4.1 Overview

When operating as Bus Controller, the core acts as master on the MIL-STD-1553 bus, initiates and performs transfers.

This mode works based on a scheduled transfer list concept. The software sets up in memory a sequence of transfer descriptors and branches, data buffers for sent and received data, and an IRQ pointer ring buffer. When the schedule is started (through a BC action register write), the core processes the list, performs the transfers one after another and writes resulting status into the transfer list and incoming data into the corresponding buffers.

24.4.2 Timing control

In each transfer descriptor in the schedule is a “slot time” field. If the scheduled transfer finishes sooner than its slot time, the core will pause the remaining time before scheduling the next command. This allows the user to accurately control the message timing during a communication frame.

If the transfer uses more than its slot time, the overshooting time will be subtracted from the following command’s time slot. The following command may in turn borrow time from the following command and so on. The core can keep track of up to one second of borrowed time, and will not insert pauses again until the balance is positive, except for intermessage gaps and pauses that the standard requires.

If you wish to execute the schedule as fast as possible you can set all slot times in the schedule to zero. If you want to group a number of transfers you can move all the slot time to the last transfer.

The schedule can be stopped or suspended by writing into the BC action register. When suspended, the schedule’s time will still be accounted, so that the schedule timing will still be correct when the schedule is resumed. When stopped, on the other hand, the schedule’s timers will be reset.

When the extsync bit is set in the schedule’s next transfer descriptor, the core will wait for a positive edge on the external sync input before starting the command. The schedule timer and the time slot balance will then be reset and the command is started. If the sync pulse arrives before the transfer is reached, it is stored so the command will begin immediately. The trigger memory is cleared when stopping (but not when suspending) the schedule. Also, the trigger can be set/cleared by software through the BC action register.

24.4.3 Bus selection

Each transfer descriptor has a bus selection bit that allows you to control on which one of the two redundant buses (‘0’ for bus A, ‘1’ for bus B) the transfer will occur.

Another way to control the bus usage is through the per-RT bus swap register, which has one register bit for each RT address. The bus swap register is an optional feature, software can check the BCFEAT read-only register field to see if it is available.

Writing a ‘1’ to a bit in the per-RT Bus Swap register inverts the meaning of the bus selection bit for all transfers to the corresponding RT, so ‘0’ now means bus ‘B’ and ‘1’ means bus ‘A’. This allows you to switch all transfers to one or a set of RT:s over to the other bus with a single register write and without having to modify any descriptors.

The hardware determines which bus to use by taking the exclusive-or of the bus swap register bit and the bus selection bit. Normally it only makes sense to use one of these two methods for each RT, either the bus selection bit is always zero and the swap register is used, or the swap register bit is always zero and the bus selection bit is used.

If the bus swap register is used for bus selection, the store-bus descriptor bit can be enabled to automatically update the register depending on transfer outcome. If the transfer succeeded on bus A, the bus swap register bit is set to ‘0’, if it succeeds on bus B, the swap register bit is set to ‘1’. If the transfer fails, the bus swap register is set to the opposite value.

24.4.4 Secondary transfer list

The core can be set up with a secondary “asynchronous” transfer list with the same format as the ordinary schedule. This transfer list can be commanded to start at any time during the ordinary schedule. While the core is waiting for a scheduled command’s slot time to finish, it will check if the next asynchronous transfer’s slot time is lower than the remaining sleep time. In that case, the asynchronous command will be scheduled.

If the asynchronous command doesn’t finish in time, time will be borrowed from the next command in the ordinary schedule. In order to not disturb the ordinary schedule, the slot time for the asynchronous messages must therefore be set to pessimistic values.

The exclusive bit in the transfer descriptor can be set if one does not want an asynchronous command scheduled during the sleep time following the transfer.

Asynchronous messages will not be scheduled while the schedule is waiting for a sync pulse or the schedule is suspended and the current slot time has expired, since it is then not known when the next scheduled command will start.

24.4.5 Interrupt generation

Each command in the transfer schedule can be set to generate an interrupt after certain transfers have completed, with or without error. Invalid command descriptors always generate interrupts and stop the schedule. Before a transfer-triggered interrupt is generated, the address to the corresponding descriptor is written into the BC transfer-triggered IRQ ring buffer and the BC Transfer-triggered IRQ Ring Position Register is incremented.

A separate error interrupt signals DMA errors. If a DMA error occurs when reading/writing descriptors, the executing schedule will be suspended. DMA errors in data buffers will cause the corresponding transfer to fail with an error code (see table 214).

Whether any of these interrupt events actually cause an interrupt request on the AMBA bus is controlled by the IRQ Mask Register setting.

24.4.6 Transfer list format

The BC:s transfer list is an array of transfer descriptors mixed with branches as shown in table 211. Each entry has to be aligned to start on a 128-bit (16-byte) boundary. The two unused words in the branch case are free to be used by software to store arbitrary data.

Table 211. GR1553B transfer descriptor format

Offset	Value for transfer descriptor	DMA R/W	Value for branch	DMA R/W
0x00	Transfer descriptor word 0 (see table 212)	R	Condition word (see table 216)	R
0x04	Transfer descriptor word 1 (see table 213)	R	Jump address, 128-bit aligned	R
0x08	Data buffer pointer, 16-bit aligned. For write buffers, if bit 0 is set the received data is discarded and the pointer is ignored. This can be used for RT-to-RT transfers where the BC is not interested in the data transferred.	R	Unused	-
0x0C	Result word, written by core (see table 214)	W	Unused	-

The transfer descriptor words are structured as shown in tables 212-214 below.

*Table 212.*GR1553B BC transfer descriptor word 0 (offset 0x00)

31	30	29	28	27	26	25	24 23	22 20	19	18	17 16	15 0
0	WTRIG	EXCL	IRQE	IRQN	SUSE	SUSN	RETMD	NRET	STBUS	GAP	RESERVED	STIME

31	Must be 0 to identify as descriptor
30	Wait for external trigger (WTRIG)
29	Exclusive time slot (EXCL) - Do not schedule asynchronous messages
28	IRQ after transfer on Error (IRQE)
27	IRQ normally (IRQN) - Always interrupts after transfer
26	Suspend on Error (SUSE) - Suspends the schedule (or stops the async transfer list) on error
25	Suspend normally (SUSN) - Always suspends after transfer
24 : 23	Retry mode (RETMD). 00 - Retry on same bus only. 01 - Retry alternating on both buses 10: Retry first on same bus, then on alternating bus. 11 - Reserved, do not use
22 : 20	Number of retries (NRET) - Number of automatic retries per bus The total number of tries (including the first attempt) is NRET+1 for RETMD=00, 2 x (NRET+1) for RETMD=01/10
19	Store bus (STBUS) - If the transfer succeeds and this bit is set, store the bus on which the transfer succeeded (0 for bus A, 1 for bus B) into the per-RT bus swap register. If the transfer fails and this bit is set, store the opposite bus instead. (only if the per-RT bus mask is supported in the core) See section 24.4.3 for more information.
18	Extended intermessage gap (GAP) - If set, adds an additional amount of gap time, corresponding to the RTTO field, after the transfer
17 : 16	Reserved - Set to 0 for forward compatibility
15 : 0	Slot time (STIME) - Allocated time in 4 microsecond units, remaining time after transfer will insert delay

*Table 213.*GR1553B BC transfer descriptor word 1 (offset 0x04)

31	30	29 26	25 21	20 16	15 11	10	9 5	4 0
DUM	BUS	RTTO	RTAD2	RTSA2	RTAD1	TR	RTSA1	WCMC

31	Dummy transfer (DUM) - If set to ‘1’ no bus traffic is generated and transfer “succeeds” immediately For dummy transfers, the EXCL,IRQN,SUSN,STBUS,GAP,STIME settings are still in effect, other bits and the data buffer pointer are ignored.	
30	Bus selection (BUS) - Bus to use for transfer, 0 - Bus A, 1 - Bus B	
29:26	RT Timeout (RTTO) - Extra RT status word timeout above nominal in units of 4 us (0000 -14 us, 1111 -74 us). Note: This extra time is also used as extra intermessage gap time if the GAP bit is set.	
25:21	Second RT Address for RT-to-RT transfer (RTAD2)	See table 215 for details on how to setup RTAD1,RTSA1,RTAD2,RTSA2,WCMC,TR for different transfer types. Note that bits 15:0 correspond to the (first) command word on the 1553 bus
20:16	Second RT Subaddress for RT-to-RT transfer (RTSA2)	
15:11	RT Address (RTAD1)	
10	Transmit/receive (TR)	
9:5	RT Subaddress (RTSA1)	
4:0	Word count/Mode code (WCMC)	

*Table 214.*GR1553B transfer descriptor result word (offset 0x0C)

31	30 24	23 16	15 8	7 4	3	2 0
0	Reserved	RT2ST	RTST	RETCNT	RES	TFRST

31	Always written as 0
30:24	Reserved - Mask away on read for forward compatibility
23:16	RT 2 Status Bits (RT2ST) - Status bits from receiving RT in RT-to-RT transfer, otherwise 0 Same bit pattern as for RTST below
15:8	RT Status Bits (RTST) - Status bits from RT (transmitting RT in RT-to-RT transfer) 15 - Message error, 14 - Instrumentation bit or reserved bit set, 13 - Service request, 12 - Broadcast command received, 11 - Busy bit, 10 - Subsystem flag, 9 - Dynamic bus control acceptance, 8 - Terminal flag
7:4	Retry count (RETCNT) - Number of retries performed
3	Reserved - Mask away on read for forward compatibility
2:0	Transfer status (TFRST) - Outcome of last try 000 - Success (or dummy bit was set) 001 - RT did not respond (transmitting RT in RT-to-RT transfer) 010 - Receiving RT of RT-to-RT transfer did not respond 011 - A responding RT:s status word had message error, busy, instrumentation or reserved bit set (*) 100 - Protocol error (improperly timed data words, decoder error, wrong number of data words) 101 - The transfer descriptor was invalid 110 - Data buffer DMA timeout or error response 111 - Transfer aborted due to loop back check failure

* Error code 011 is issued only when the number of data words match the success case, otherwise code 100 is used. Error code 011 can be issued for a correctly executed “transmit last command” or “transmit last status word” mode code since these commands do not reset the status word.

*Table 215.*GR1553B BC Transfer configuration bits for different transfer types

Transfer type	**RTAD1 (15:11)**	**RTSA1 (9:5)**	**RTAD2 (25:21)**	**RTSA2 (20:16)**	**WCMC (4:0)**	**TR (10)**	**Data buffer direction**
Data, BC-to-RT	RT address (0-30)	RT subaddr (1-30)	Don’t care	0	Word count (0 for 32)	0	Read (2-64 bytes)
Data, RT-to-BC	RT address (0-30)	RT subaddr (1-30)	Don’t care	0	Word count (0 for 32)	1	Write (2-64 bytes)
Data, RT-to-RT	Recv-RT addr (0-30)	Recv-RT subad. (1-30)	Xmit-RT addr (0-30)	Xmit-RT subad. (1-30)	Word count (0 for 32)	0	Write (2-64 bytes)
Mode, no data	RT address (0-30)	0 or 31 (*)	Don’t care	Don’t care	Mode code (0-8)	1	Unused
Mode, RT-to-BC	RT address (0-30)	0 or 31 (*)	Don’t care	Don’t care	Mode code (16/18/19)	1	Write (2 bytes)
Mode, BC-to-RT	RT address (0-30)	0 or 31 (*)	Don’t care	Don’t care	Mode code (17/20/21)	0	Read (2 bytes)
Broadcast Data, BC-to-RTs	31	RTs subaddr (1-30)	Don’t care	0	Word count (0 for 32)	0	Read (2-64 bytes)
Broadcast Data, RT-to-RTs	31	Recv-RTs subad. (1-30)	Xmit-RT addr (0-30)	Xmit-RT subad. (1-30)	Word count (0 for 32)	0	Write (2-64 bytes)
Broadcast Mode, no data	31	0 or 31 (*)	Don’t care	Don’t care	Mode code (1, 3-8)	1	Unused
Broadcast Mode, BC-to-RT	31	0 or 31 (*)	Don’t care	Don’t care	Mode code (17/20/21)	0	Read (2 bytes)

(*) The standard allows using either of subaddress 0 or 31 for mode commands.

The branch condition word is formed as shown in table 216.

*Table 216.*GR1553B branch condition word (offset 0x00)

31	30 27	26	25	24	23 16	15 8	7 0
1	Reserved (0)	IRQC	ACT	MODE	RT2CC	RTCC	STCC

31	Must be 1 to identify as branch
30 : 27	Reserved - Set to 0
26	Interrupt if condition met (IRQC)
25	Action (ACT) - What to do if condition is met, 0 - Suspend schedule, 1 - Jump
24	Logic mode (MODE): 0 = Or mode (any bit set in RT2CC, RTCC is set in RT2ST,RTST, or result is in STCC mask) 1 - And mode (all bits set in RT2CC,RTCC are set in RT2ST,RTST and result is in STCC mask)
23:16	RT 2 Condition Code (RT2CC) - Mask with bits corresponding to RT2ST in result word of last transfer
15:8	RT Condition Code (RTCC) - Mask with bits corresponding to RTST in result word of last transfer
7:0	Status Condition Code (STCC) - Mask with bits corresponding to status value of last transfer

Note that you can get a constant true condition by setting MODE=0 and STCC=0xFF, and a constant false condition by setting STCC=0x00. 0x800000FF can thus be used as an end-of-list marker.

24.5 Remote Terminal Operation

24.5.1 Overview

When operating as Remote Terminal, the core acts as a slave on the MIL-STD-1553B bus. It listens for requests to its own RT address (or broadcast transfers), checks whether they are configured as legal and, if legal, performs the corresponding transfer or, if illegal, sets the message error flag in the status word. Legality is controlled by the subaddress control word for data transfers and by the mode code control register for mode codes.

To start the RT, set up the subaddress table and log ring buffer, and then write the address and RT enable bit is into the RT Config Register.

24.5.2 Data transfer handling

The Remote Terminal mode uses a three-level structure to handle data transfer DMA. The top level is a subaddress table, where each subaddress has a subaddress control word, and pointers to a transmit descriptor and a receive descriptor. Each descriptor in turn contains a descriptor control/status word, pointer to a data buffer, and a pointer to a next descriptor, forming a linked list or ring of descriptors. Data buffers can reside anywhere in memory with 16-bit alignment.

When the RT receives a data transfer request, it checks in the subaddress table that the request is legal. If it is legal, the transfer is then performed with DMA to or from the corresponding data buffer. After a data transfer, the descriptor's control/status word is updated with success or failure status and the subaddress table pointer is changed to point to the next descriptor.

If logging is enabled, a log entry will be written into a log ring buffer area. A transfer-triggered IRQ may also be enabled. To identify which transfer caused the interrupt, the RT Event Log IRQ Position points to the corresponding log entry. For that reason, logging must be enabled in order to enable interrupts.

If a request is legal but can not be fulfilled, either because there is no valid descriptor ready or because the data can not be accessed within the required response time, the core will signal a RT table access error interrupt and not respond to the request. Optionally, the terminal flag status bit can be automatically set on these error conditions.

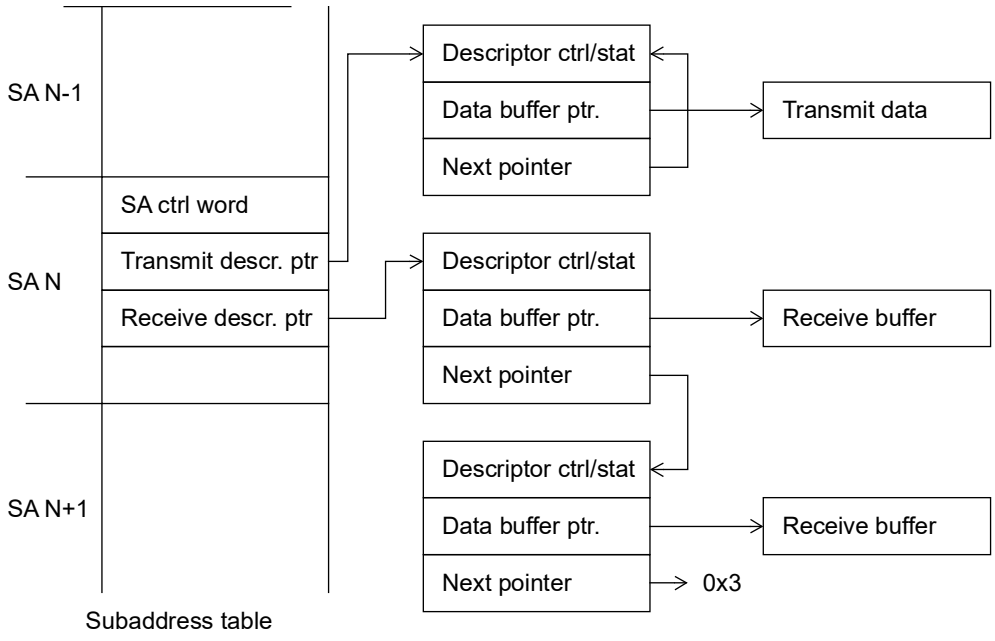


Figure 62. RT subaddress data structure example diagram

24.5.3 Mode Codes

Which of the MIL-STD-1553B mode codes that are legal and should be logged and interrupted are controlled by the RT Mode Code Control register. As for data transfers, to enable interrupts you must also enable logging. Inhibit mode codes are controlled by the same fields as their non-inhibit counterpart and mode codes that can be broadcast have two separate fields to control the broadcast and non-broadcast variants.

The different mode codes and the corresponding action taken by the RT are tabulated below. Some mode codes do not have a built-in action, so they will need to be implemented in software if desired. The relation between each mode code to the fields in the RT Mode Code control register is also shown.

*Table 217.*RT Mode Codes

Mode code		Description	Built-in action, if mode code is enabled	Can log/ IRQ	Enabled after reset	Ctrl. reg bits
0	00000	Dynamic bus control	If the DBCA bit is set in the RT Bus Status register, a Dynamic Bus Control Acceptance response is sent.	Yes	No	17:16
1	00001	Synchronize	The time field in the RT sync register is updated. The output rtsync is pulsed high one AMBA cycle.	Yes	Yes	3:0
2	00010	Transmit status word	Transmits the RT:s status word Enabled always, can not be logged or disabled.	No	Yes	-
3	00011	Initiate self test	No built-in action	Yes	No	21:18
4	00100	Transmitter shutdown	The RT will stop responding to commands on the other bus (not the bus on which this command was given).	Yes	Yes	11:8
5	00101	Override transmitter shutdown	Removes the effect of an earlier transmitter shutdown mode code received on the same bus	Yes	Yes	11:8
6	00110	Inhibit terminal flag	Masks the terminal flag of the sent RT status words	Yes	No	25:22
7	00111	Override inhibit terminal flag	Removes the effect of an earlier inhibit terminal flag mode code.	Yes	No	25:22
8	01000	Reset remote terminal	The fail-safe timers, transmitter shutdown and inhibit terminal flag inhibit status are reset. The Terminal Flag and Service Request bits in the RT Bus Status register are cleared. The extreset output is pulsed high one AMBA cycle.	Yes	No	29:26
16	10000	Transmit vector word	Responds with vector word from RT Status Words Register	Yes	No	13:12
17	10001	Synchronize with data word	The time and data fields in the RT sync register are updated. The rtsync output is pulsed high one AMBA cycle	Yes	Yes	7:4
18	10010	Transmit last command	Transmits the last command sent to the RT. Enabled always, can not be logged or disabled.	No	Yes	-
19	10011	Transmit BIT word	Responds with BIT word from RT Status Words Register	Yes	No	15:14
20	10100	Selected transmitter shutdown	No built-in action	No	No	-
21	10101	Override selected transmitter shutdown	No built-in action	No	No	-

24.5.4 Event Log

The event log is a ring of 32-bit entries, each entry having the format given in table 218. Note that for data transfers, bits 23-0 in the event log are identical to bits 23-0 in the descriptor status word.

*Table 218.*GR1553B RT Event Log entry format

31	30 29	28 24	23 10	9	8 3	2 0
IRQSR	TYPE	SAMC	TIMEL	BC	SZ	TRES

31	IRQ Source (IRQSRC) - Set to '1' if this transfer caused an interrupt
30 : 29	Transfer type (TYPE) - 00 - Transmit data, 01 - Receive data, 10 - Mode code
28 : 24	Subaddress / Mode code (SAMC) - If TYPE=00/01 this is the transfer subaddress, If TYPE=10, this is the mode code
23 : 10	TIMEL - Low 14 bits of time tag counter.
9	Broadcast (BC) - Set to 1 if request was to the broadcast address
8 : 3	Transfer size (SZ) - Count in 16-bit words (0-32)
2 : 0	Transfer result (TRES) 000 = Success 001 = Superseded (canceled because a new command was given on the other bus) 010 = DMA error or memory timeout occurred 011 = Protocol error (improperly timed data words or decoder error) 100 = The busy bit or message error bit was set in the transmitted status word and no data was sent 101 = Transfer aborted due to loop back checker error

24.5.5 Subaddress table format

*Table 219.*GR1553B RT Subaddress table entry for subaddress number N, 0<N<31

Offset	Value	DMA R/W
0x10*N + 0x00	Subaddress N control word (table 220)	R
0x10*N + 0x04	Transmit descriptor pointer, 16-byte aligned (0x3 to indicate invalid pointer)	R/W
0x10*N + 0x08	Receive descriptor pointer, 16-byte aligned (0x3 to indicate invalid pointer)	R/W
0x10*N + 0x0C	Unused	-

Note: The table entries for mode code subaddresses 0 and 31 are never accessed by the core.

*Table 220.*GR1553B RT Subaddress table control word (offset 0x00)

31 19	18	17	16	15	14	13	12 8	7	6	5	4 0
0 (reserved)	WRAP	IGNDV	BCRXE	RXEN	RXLOG	RXIRQ	RXSZ	TXEN	TXLOG	TXIRQ	TXSZ

31 : 19	Reserved - set to 0 for forward compatibility
18	Auto-wraparound enable (WRAP) - Enables a test mode for this subaddress, where transmit transfers send back the last received data. This is done by copying the finished transfer's descriptor pointer to the transmit descriptor pointer address after each successful transfer. Note: If WRAP=1, you should not set TXSZ > RXSZ as this might cause reading beyond buffer end
17	Ignore data valid bit (IGNDV) - If this is '1' then receive transfers will proceed (and overwrite the buffer) if the receive descriptor has the data valid bit set, instead of not responding to the request. This can be used for descriptor rings where you don't care if the oldest data is overwritten.
16	Broadcast receive enable (BCRXEN) - Allow broadcast receive transfers to this subaddress
15	Receive enable (RXEN) - Allow receive transfers to this subaddress
14	Log receive transfers (RXLOG) - Log all receive transfers in event log ring (only used if RXEN=1)
13	Interrupt on receive transfers (RXIRQ) - Each receive transfer will cause an interrupt (only if also RXEN,RXLOG=1)
12 : 8	Maximum legal receive size (RXSZ) to this subaddress - in16-bit words, 0 means 32
7	Transmit enable (TXEN) - Allow transmit transfers from this subaddress
6	Log transmit transfers (TXLOG) - Log all transmit transfers in event log ring (only if also TXEN=1)
5	Interrupt on transmit transfers (TXIRQ) - Each transmit transfer will cause an interrupt (only if TXEN,TXLOG=1)
4 : 0	Maximum legal transmit size (TXSZ) from this subaddress - in 16-bit words, 0 means 32

*Table 221.*GR1553B RT Descriptor format

Offset	Value	DMA R/W
0x00	Control and status word, see table 222	R/W
0x04	Data buffer pointer, 16-bit aligned	R
0x08	Pointer to next descriptor, 16-byte aligned or 0x0000003 to indicate end of list	R

*Table 222.*GR1553B RT Descriptor control/status word (offset 0x00)

31	30	29 26	25 10	9	8 3	2 0
DV	IRQEN	Reserved (0)	TIME	BC	SZ	TRES

31	Data valid (DV) - Should be set to 0 by software before and set to 1 by hardware after transfer. If DV=1 in the current receive descriptor before the receive transfer begins then a descriptor table error will be triggered. You can override this by setting the IGNDV bit in the subaddress table.
30	IRQ Enable override (IRQEN) - Log and IRQ after transfer regardless of SA control word settings Can be used for getting an interrupt when nearing the end of a descriptor list.
29 : 26	Reserved - Write 0 and mask out on read for forward compatibility
25 : 10	Transmission time tag (TTIME) - Set by the core to the value of the RT timer when the transfer finished.
9	Broadcast (BC) - Set by the core if the transfer was a broadcast transfer
8 : 3	Transfer size (SZ) - Count in 16-bit words (0-32)
2 : 0	Transfer result (TRES) 000 = Success 001 = Superseded (canceled because a new command was given on the other bus) 010 = DMA error or memory timeout occurred 011 = Protocol error (improperly timed data words or decoder error) 100 = The busy bit or message error bit was set in the transmitted status word and no data was sent 101 = Transfer aborted due to loop back checker error

24.6 Bus Monitor Operation

24.6.1 Overview

The Bus Monitor (BM) can be enabled by itself, or in parallel to the BC or RT. The BM acts as a passive logging device, writing received data with time stamps to a ring buffer.

24.6.2 Filtering

The Bus Monitor can also support filtering. This is an optional feature, software can check for this by testing whether the BM filter registers are writable.

Transfers can be filtered per RT address and per subaddress or mode code, and the filter conditions are logically AND:ed. If all bits of the three filter registers and bits 2-3 of the control register are set to '1', the BM core will log all words that are received on the bus.

In order to filter on subaddress/mode code, the BM has logic to track 1553 words belonging to the same message. All 10 message types are supported. If an unexpected word appears, the filter logic will restart. Data words not appearing to belong to any message can be logged by setting a bit in the control register.

The filter logic can be manually restarted by setting the BM enable bit low and then back to high. This feature is mainly to improve testability of the BM itself.

24.6.3 No-response handling

In the MIL-STD-1553B protocol, a command word for a mode code using indicator 0 or a regular transfer to subaddress 8 has the same structure as a legal status word. Therefore ambiguity can arise when the subaddress or mode code filters are used, an RT is not responding on a subaddress, and the BC then commands the same RT again on subaddress 8 or mode code indicator 0 on the same bus. This can lead to the second command word being interpreted as a status word and filtered out.

The BM can use the instrumentation bit and reserved bits to disambiguate, which means that this case will never occur when subaddresses 1-7, 9-30 and mode code indicator 31 are used. Also, this case does not occur when the subaddress/mode code filters are unused and only the RT address filter is used.

24.6.4 Log entry format

Each log entry is two 32-bit words.

Table 223. GR1553B BM Log entry word 0 (offset 0x00)

31	30 24	23 0
1	Reserved	TIME

31	Always written as 1
30 : 24	Reserved - Mask out on read for forward compatibility
23 : 0	Time tag (TIME)

Table 224. GR1553B BM Log entry word 1 (offset 0x04)

31	30 20	19	18 17	16	15 0
0	Reserved	BUS	WST	WTP	WD

31	Always written as 0
30 : 20	Reserved - Mask out on read for forward compatibility
19	Receive data bus (BUS) - 0:A, 1:B
18 : 17	Word status (WST) - 00=word OK, 01=Manchester error, 10=Parity error
16	Word type (WTP) - 0:Data, 1:Command/status
15 : 0	Word data (WD)

24.7 Registers

The core is programmed through registers mapped into APB address space. Reserved register fields should be written as zeroes and masked out on read.

*Table 225.*MIL-STD-1553B interface registers

APB address offset	Register	R/W	Reset value
0x00	IRQ Register	RW (write '1' to clear)	0x00000000
0x04	IRQ Enable	RW	0x00000000
0x08...0x0F	(Reserved)		
0x10	Hardware config register	R (constant)	0x00000000*
0x14...0x3F	(Reserved)		
0x40...0x7F	BC Register area (see table 226)		
0x80...0xBF	RT Register area (see table 227)		
0xC0...0xFF	BM Register area (see table 228)		

(*) May differ depending on core configuration

*Table 226.*MIL-STD-1553B interface BC-specific registers

APB address offset	Register	R/W	Reset value
0x40	BC Status and Config register	RW	0xf0000000*
0x44	BC Action register	W	
0x48	BC Transfer list next pointer	RW	0x00000000
0x4C	BC Asynchronous list next pointer	RW	0x00000000
0x50	BC Timer register	R	0x00000000
0x54	BC Timer wake-up register	RW	0x00000000
0x58	BC Transfer-triggered IRQ ring position	RW	0x00000000
0x5C	BC Per-RT bus swap register	RW	0x00000000
0x60...0x67	(Reserved)		
0x68	BC Transfer list current slot pointer	R	0x00000000
0x6C	BC Asynchronous list current slot pointer	R	0x00000000
0x70...0x7F	(Reserved)		

(*) May differ depending on core configuration

*Table 227.*MIL-STD-1553B interface RT-specific registers

APB address offset	Register	R/W	Reset value
0x80	RT Status register	R	0x80000000
0x84	RT Config register	RW	0x0000e03e***
0x88	RT Bus status bits register	RW	0x00000000
0x8C	RT Status words register	RW	0x00000000
0x90	RT Sync register	R	0x00000000
0x94	RT Subaddress table base address	RW	0x00000000
0x98	RT Mode code control register	RW	0x00000555
0x9C...0xA3	(Reserved)		
0xA4	RT Time tag control register	RW	0x00000000
0xA8	(Reserved)		
0xAC	RT Event log size mask	RW	0xfffffffc
0xB0	RT Event log position	RW	0x00000000
0xB4	RT Event log interrupt position	R	0x00000000
0xB8.. 0xBF	(Reserved)		

(***) Reset value is affected by the external RTADDR/RTPAR input signals

*Table 228.*MIL-STD-1553B interface BM-specific registers

APB address offset	Register	R/W	Reset value
0xC0	BM Status register	R	0x80000000
0xC4	BM Control register	RW	0x00000000
0xC8	BM RT Address filter register	RW	0xffffffff
0xCC	BM RT Subaddress filter register	RW	0xffffffff
0xD0	BM RT Mode code filter register	RW	0xffffffff
0xD4	BM Log buffer start	RW	0x00000000
0xD8	BM Log buffer end	RW	0x00000007
0xDC	BM Log buffer position	RW	0x00000000
0xE0	BM Time tag control register	RW	0x00000000
0xE4...0xFF	(Reserved)		

24.7.1 IRQ Register

*Table 229.*0x00 - IRQ - GR1553B IRQ Register

31 18	17	16	15 11	10	9	8	7 3	2	1	0
RESERVED	BMTOF	BMD	RESERVED	RTTE	RTD	RTEV	RESERVED	BCWK	BCD	BCEV
0	0	0	0	0	0	0	0	0	0	0
r	wc	wc	r	wc	wc	wc	r	wc	wc	wc

Bits read '1' if interrupt occurred, write back '1' to acknowledge

- 17 BM Timer overflow (BMTOF)
- 16 BM DMA Error (BMD)
- 10 RT Table access error (RTTE)
- 9 RT DMA Error (RTD)
- 8 RT transfer-triggered event interrupt (RTEV)
- 2 BC Wake-up timer interrupt (BCWK)
- 1 BC DMA Error (BCD)
- 0 BC Transfer-triggered event interrupt (BCEV)

24.7.2 IRQ Enable Register

*Table 230.*0x04 - IRQE - GR1553B IRQ Enable Register

31 18	17	16	15 11	10	9	8	7 3	2	1	0
RESERVED	BMTOE	BMDE	RESERVED	RTTEE	RTDE	RTEVE	RESERVED	BCWKE	BCDE	BCEVE
0	0	0	0	0	0	0	0	0	0	0
r	rw	rw	r	rw	rw	rw	r	rw	rw	rw

- 17 BM Timer overflow interrupt enable (BMTOE)
- 16 BM DMA error interrupt enable (BMDE)
- 10 RT Table access error interrupt enable (RTTEE)
- 9 RT DMA error interrupt enable (RTDE)
- 8 RT Transfer-triggered event interrupt enable (RTEVE)
- 2 BC Wake up timer interrupt (BCWKE)
- 1 BC DMA Error Enable (BCDE)
- 0 BC Transfer-triggered event interrupt (BCEVE)

24.7.3 Hardware Configuration Register

*Table 231.*GR1553B Hardware Configuration Register

31	30 12	11	10 9	8	7 0
MOD	RESERVED	XKEYS	ENDIAN	SCLK	CCFREQ
0	0	0	0	0	0
r	r	r	r	r	r

Note: This register reads 0x0000 for the standard configuration of the core

- 31 Modified (MOD) - Reserved to indicate that the core has been modified / customized in an unspecified manner
- 11 Set if safety keys are enabled for the BM Control Register and for all RT Control Register fields.
- 10 : 9 AHB Endianness - 00=Big-endian, 01=Little-endian, 10/11=Reserved
- 8 Same clock (SCLK) - Reserved for future versions to indicate that the core has been modified to run with a single clock
- 7 : 0 Codec clock frequency (CCFREQ) - Reserved for future versions of the core to indicate that the core runs at a different codec clock frequency. Frequency value in MHz, a value of 0 means 20 MHz.

24.7.4 BC Status and Config Register

*Table 232.*0x40 - BCSL - GR1553B BC Status and Config Register

31	30 28	27 17	16	15 11	10	9 8	7 3	2 0
BCSUP	BCFEAT	RESERVED	BCCHK	ASADL	R	ASST	SCADL	SCST
*	*	0	0	0	0	0	0	0
r	r	r	rw	r	r	r	r	r

31 BC Supported (BCSUP) - Reads '1' if core supports BC mode

30 : 28 BC Features (BCFEAT) - Bit field describing supported optional features ('1'=supported):

- 30 BC Schedule timer supported
- 29 BC Schedule time wake-up interrupt supported
- 28 BC per-RT bus swap register and STBUS descriptor bit supported

16 Check broadcasts (BCCHK) - Writable bit, if set to '1' enables waiting and checking for (unexpected) responses to all broadcasts.

15 : 11 Asynchronous list address low bits (ASADL) - Bit 8-4 of currently executing (if ASST=01) or next asynchronous command descriptor address

9 : 8 Asynchronous list state (ASST) - 00=Stopped, 01=Executing command, 10=Waiting for time slot

7 : 3 Schedule address low bits (SCADL) - Bit 8-4 of currently executing (if SCST=001) or next schedule descriptor address

2 : 0 Schedule state (SCST) - 000=Stopped, 001=Executing command, 010=Waiting for time slot, 011=Suspended, 100=Waiting for external trigger

24.7.5 BC Action Register

*Table 233.*0x44 - BCA - GR1553B BC Action Register

31 16	15 10	9	8	7 5	4	3	2	1	0
BCKEY	RESERVED	ASSTP	ASSRT	RESERVED	CLRT	SETT	SCSTP	SCSUS	SCSRT
-	-	-	-	-	-	-	-	-	-
w	-	w	w	-	w	w	w	w	w

31 : 16 Safety code (BCKEY) - Must be 0x1552 when writing, otherwise register write is ignored

9 Asynchronous list stop (ASSTP) - Write '1' to stop asynchronous list (after current transfer, if executing)

8 Asynchronous list start (ASSRT) - Write '1' to start asynchronous list

4 Clear external trigger (CLRT) - Write '1' to clear trigger memory

3 Set external trigger (SETT) - Write '1' to force the trigger memory to set

2 Schedule stop (SCSTP) - Write '1' to stop schedule (after current transfer, if executing)

1 Schedule suspend (SCSUS) - Write '1' to suspend schedule (after current transfer, if executing)

0 Schedule start (SCSRT) - Write '1' to start schedule

24.7.6 BC Transfer List Next Pointer Register

*Table 234.*0x48 - BCTNP - GR1553B BC Transfer list next pointer register

31 0
SCHEDULE TRANSFER LIST POINTER
0
rw

31 : 0 Read: Currently executing (if SCST=001) or next transfer to be executed in regular schedule.
Write: Change address. If running, this will cause a jump after the current transfer has finished.

24.7.7 BC Asynchronous List Next Pointer Register

*Table 235.*0x4C - BCANP - GR1553B BC Asynchronous list next pointer register

31	0
ASYNCHRONOUS LIST POINTER	
0	
rw	

31 :0 Read: Currently executing (if ASST=01) or next transfer to be executed in asynchronous schedule.
Write: Change address. If running, this will cause a jump after the current transfer has finished.

24.7.8 BC Timer Register

*Table 236.*0x50 - BCT - GR1553B BC Timer register

31 24	23 0
RESERVED	SCHEDULE TIME (SCTM)
0	0
r	r

23 : 0 Elapsed “transfer list” time in microseconds (read-only)
Set to zero when schedule is stopped or on external sync.

Note: This register is an optional feature, see BC Status and Config Register, bit 30

24.7.9 BC Timer Wake-up Register

*Table 237.*0x54 - BCTW - GR1553B BC Timer Wake-up register

31	30 24	23 0
WKEN	RESERVED	WAKE-UP TIME (WKTM)
0	0	0
rw	r	rw

31 Wake-up timer enable (WKEN) - If set, an interrupt will be triggered when WKTM=SCTM

23 : 0 Wake-up time (WKTM).

Note: This register is an optional feature, see BC Status and Config Register, bit 29

24.7.10 BC Transfer-triggered IRQ Ring Position Register

*Table 238.*0x58 - BCRD - GR1553B BC Transfer-triggered IRQ ring position register

31	0
BC IRQ SOURCE POINTER RING POSITION	
0	
rw	

31 : 0 The current write pointer into the transfer-tirggered IRQ descriptor pointer ring.
Bits 1:0 are constant zero (4-byte aligned)
The ring wraps at the 64-byte boundary, so bits 31:6 are only changed by user

24.7.11 BC per-RT Bus Swap Register

*Table 239.*0x5C - BCBS - GR1553B BC per-RT Bus swap register

31	0
BC PER-RT BUS SWAP	
0	
rw	

31 : 0	The bus selection value will be logically exclusive-or:ed with the bit in this mask corresponding to the addressed RT (the receiving RT for RT-to-RT transfers). This register gets updated by the core if the STBUS descriptor bit is used. For more information on how to use this feature, see section 24.4.3.

Note: This register is an optional feature, see BC Status and Config Register, bit 28

24.7.12 BC Transfer List Current Slot Pointer

*Table 240.*0x68 - BCTCP - GR1553B BC Transfer list current slot pointer

31	0
BC TRANSFER SLOT POINTER	
0	
r	

31 : 0	Points to the transfer descriptor corresponding to the current time slot (read-only, only valid while transfer list is running). Bits 3:0 are constant zero (128-bit/16-byte aligned)

24.7.13 BC Asynchronous List Current Slot Pointer

*Table 241.*0x6C - BCACP - GR1553B BC Asynchronous list current slot pointer

31	0
BC TRANSFER SLOT POINTER	
0	
r	

31 : 0	Points to the transfer descriptor corresponding to the current asynchronous schedule time slot (read-only, only valid while asynchronous list is running). Bits 3:0 are constant zero (128-bit/16-byte aligned)

24.7.14 RT Status Register

*Table 242.*0x80 - RTS - GR1553B RT Status register (read-only)

31	30	4	3	2	1	0
RTSUP	RESERVED		ACT	SHDA	SHDB	RUN

31	RT Supported (RTSUP) - Reads ‘1’ if core supports RT mode
3	RT Active (ACT) - ‘1’ if RT is currently processing a transfer
2	Bus A shutdown (SHDA) - Reads ‘1’ if bus A has been shut down by the BC (using the transmitter shutdown mode command on bus B)
1	Bus B shutdown (SHDB) - Reads ‘1’ if bus B has been shut down by the BC (using the transmitter shutdown mode command on bus A)
0	RT Running (RUN) - ‘1’ if the RT is listening to commands.

24.7.15 RT Config Register

*Table 243.*0x84 - RTC - GR1553B RT Config register

31 16	15	14	13	12 7	6	5 1	0
RTKEY	SYS	SYDS	BRS	RESERVED	RTEIS	RTADDR	RTEN
0	1	1	1	0	*	*	0
w	rw	rw	rw	r	r	rw	rw

31 : 16	Safety code (RTKEY) - Must be written as 0x1553 when changing the RT address, otherwise the address field is unaffected by the write. When reading the register, this field reads 0x0000. If extra safety keys are enabled (see Hardware Config Register), the lower half of the key is used to also protect the other fields in this register.
15	Sync signal enable (SYS) - Set to '1' to pulse the rtsync output when a synchronize mode code (without data) has been received
14	Sync with data signal enable (SYDS) - Set to '1' to pulse the rtsync output when a synchronize with data word mode code has been received
13	Bus reset signal enable (BRS) - Set to '1' to pulse the busreset output when a reset remote terminal mode code has been received.
6	Reads '1' if current address was set through external inputs. After setting the address from software this field is set to '0'
5 : 1	RT Address (RTADDR) - This RT:s address (0-30)
0	RT Enable (RTEN) - Set to '1' to enable listening for requests

24.7.16 RT Bus Status Register

*Table 244.*0x88 - RTBS - GR1553B RT Bus status register

31 9	8	7 5	4	3	2	1	0
RESERVED	TFDE	RESERVED	SREQ	BUSY	SSF	DBCA	TFLG
0	0	0	0	0	0	0	0
r	rw	rw	rw	rw	rw	rw	rw

8	Set Terminal flag automatically on DMA and descriptor table errors (TFDE)
4 : 0	These bits will be sent in the RT:s status responses over the 1553 bus.
4	Service request (SREQ)
3	Busy bit (BUSY) Note: If the busy bit is set, the RT will respond with only the status word and the transfer "fails"
2	Subsystem Flag (SSF)
1	Dynamic Bus Control Acceptance (DBCA) Note: This bit is only sent in response to the Dynamic Bus Control mode code
0	Terminal Flag (TFLG) The BC can mask this flag using the "inhibit terminal flag" mode command, if legal

24.7.17 RT Status Words Register

*Table 245.*0x8C - RTSW - GR1553B RT Status words register

31 16	15 0
BIT WORD (BITW)	VECTOR WORD (VECW)
0	0
rw	rw

31 : 16	BIT Word - Transmitted in response to the "Transmit BIT Word" mode command, if legal
15 : 0	Vector word - Transmitted in response to the "Transmit vector word" mode command, if legal.

24.7.18 RT Sync Register

*Table 246.*0x90 - RTSY - GR1553B RT Sync register

31 16	15 0
SYNC TIME (SYTM)	SYNC DATA (SYD)
0	0
r	r

31 : 16	The value of the RT timer at the last sync or sync with data word mode command, if legal.
15 : 0	The data received with the last synchronize with data word mode command, if legal

24.7.19 Sub Address Table Base Address Register

*Table 247.*0x94 - RTSTBA - GR1553B RT Sub address table base address register

31 9	8 0
SUBADDRESS TABLE BASE (SATB)	RESERVED
0	0
rw	r

31 : 9	Base address, bits 31-9 for subaddress table
8 : 0	Always read ‘0’, writing has no effect

24.7.20 RT Mode Code Control Register

*Table 248.*0x98 - RTMCC - GR1553B RT Mode code control register

31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16
RESERVED	RRTB	RRT	ITFB	ITF	ISTB	IST	DBC
0	0	0	0	0	0	0	0
r	rw	rw	rw	rw	rw	rw	rw

15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
TBW	TVW	TSB	TS	SDB	SD	SB	S
0	0	1	1	1	1	1	1
rw	rw	rw	rw	rw	rw	rw	rw

	For each mode code: “00” - Illegal, “01” - Legal, “10” - Legal, log enabled, “11” - Legal, log and interrupt
29 : 28	Reset remote terminal broadcast (RRTB)
27 : 26	Reset remote terminal (RRT)
25 : 24	Inhibit & override inhibit terminal flag bit broadcast (ITFB)
23 : 22	Inhibit & override inhibit terminal flag (ITF)
21 : 20	Initiate self test broadcast (ISTB)
19 : 18	Initiate self test (IST)
17 : 16	Dynamic bus control (DBC)
15 : 14	Transmit BIT word (TBW)
13 : 12	Transmit vector word (TVW)
11 : 10	Transmitter shutdown & override transmitter shutdown broadcast (TSB)
9 : 8	Transmitter shutdown & override transmitter shutdown (TS)
7 : 6	Synchronize with data word broadcast (SDB)
5 : 4	Synchronize with data word (SD)
3 : 2	Synchronize broadcast (SB)
1 : 0	Synchronize (S)

24.7.21 RT Time Tag Control Register

Table 249. 0xA4 - RTTTC - GR1553B RT Time tag control register

31 16	15 0
TIME RESOLUTION (TRES)	TIME TAG VALUE (TVAL)
0	0
rw	rw

31 : 16 Time tag resolution (TRES) - Time unit of RT:s time tag counter in microseconds, minus 1

15 : 0 Time tag value (TVAL) - Current value of running time tag counter

24.7.22 RT Event Log Mask Register

Table 250. 0xAC - RTELM - GR1553B RT Event Log mask register

31 21	20 2	1 0
RESERVED	EVENT LOG SIZE MASK	RES
	0xFFFFFFC	
r	rw	r

31 : 0 Mask determining size and alignment of the RT event log ring buffer. All bits “above” the size should be set to ‘1’, all bits below should be set to ‘0’

24.7.23 RT Event Log Position Register

Table 251. 0xB0 - RTELP - GR1553B RT Event Log position register

31 0
EVENT LOG WRITE POINTER
0
rw

31 : 0 Address to first unused/oldest entry of event log buffer, 32-bit aligned

24.7.24 RT Event Log Interrupt Position Register

Table 252. 0xB4 - RTELIP - GR1553B RT Event Log interrupt position register

31 0
EVENT LOG IRQ POINTER
0
r

31 : 0 Address to event log entry corresponding to interrupt, 32-bit aligned
The register is set for the first interrupt and not set again until the interrupt has been acknowledged.

24.7.25 BM Status Register

Table 253. 0xC0 - BMS - GR1553B BM Status register

31	30	29 0
BMSUP	KEYEN	RESERVED
*	*	0
r	r	r

31 BM Supported (BMSUP) - Reads ‘1’ if BM support is in the core.

30 Key Enabled (KEYEN) - Reads ‘1’ if the BM validates the BMKEY field when the control register is written.

24.7.26 BM Control Register

*Table 254.*0xC4 - BMC - GR1553B BM Control register

31 16	15 6	5	4	3	2	1	0
BMKEY	RESERVED	WRSTP	EXST	IMCL	UDWL	MANL	BMEN
0	0	0	0	0	0	0	0
rw	r	rw	rw	rw	rw	rw	rw

31 : 16	Safety key - If extra safety keys are enabled (see KEYEN), this field must be 0x1543 for a write to be accepted. Is 0x0000 when read.
5	Wrap stop (WRSTP) - If set to ‘1’, BMEN will be set to ‘0’ and stop the BM when the BM log position wraps around from buffer end to buffer start
4	External sync start (EXST) - If set to ‘1’,BMEN will be set to ‘1’ and the BM is started when an external BC sync pulse is received
3	Invalid mode code log (IMCL) - Set to ‘1’ to log invalid or reserved mode codes.
2	Unexpected data word logging (UDWL) - Set to ‘1’ to log data words not seeming to be part of any command
1	Manchester/parity error logging (MANL) - Set to ‘1’ to log bit decoding errors
0	BM Enable (BMEN) - Must be set to ‘1’ to enable any BM logging

24.7.27 BMRT Address Filter Register

*Table 255.*0xC8 - BMRTAF - GR1553B BM RT Address filter register

31 0
ADDRESS FILTER MASK
0xFFFFFFFF
rw

31	Enables logging of broadcast transfers
30 : 0	Each bit position set to ‘1’ enables logging of transfers with the corresponding RT address

24.7.28 BMRT Sub address Filter Register

*Table 256.*0xCC - BMRTSF - GR1553B BM RT Sub address filter register

31 0
SUBADDRESS FILTER MASK
0xFFFFFFFF
rw

31	Enables logging of mode commands on sub address 31
30 : 1	Each bit position set to ‘1’ enables logging of transfers with the corresponding RT sub address
0	Enables logging of mode commands on sub address 0

24.7.29 BMRT Mode Code Filter Register

*Table 257.*0xCC - BMRTMC - GR1553B BM RT Mode code filter register

31	19	18	17	16
RESERVED		STSB	STS	TLC
0x1ttt		1	1	1
r		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSW	RRTB	RRT	ITFB	ITF	ISTB	IST	DBC	TBW	TVW	TSB	TS	SDB	SD	SB	S
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Each bit set to '1' enables logging of a mode code:

18	Selected transmitter shutdown broadcast & override selected transmitter shutdown broadcast (STSB)
17	Selected transmitter shutdown & override selected transmitter shutdown (STS)
16	Transmit last command (TLC)
15	Transmit status word (TSW)
14	Reset remote terminal broadcast (RRTB)
13	Reset remote terminal (RRT)
12	Inhibit & override inhibit terminal flag bit broadcast (ITFB)
11	Inhibit & override inhibit terminal flag (ITF)
10	Initiate self test broadcast (ISTB)
9	Initiate self test (IST)
8	Dynamic bus control (DBC)
7	Transmit BIT word (TBW)
6	Transmit vector word (TVW)
5	Transmitter shutdown & override transmitter shutdown broadcast (TSB)
4	Transmitter shutdown & override transmitter shutdown (TS)
3	Synchronize with data word broadcast (SDB)
2	Synchronize with data word (SD)
1	Synchronize broadcast (SB)
0	Synchronize (S)

24.7.30 BMLog Buffer Start

*Table 258.*0xD4 - BMLBS - GR1553B BM Log buffer start

31	0
BM LOG BUFFER START	
0	
rw	

31 : 0	Pointer to the lowest address of the BM log buffer (8-byte aligned) Due to alignment, bits 2:0 are always 0.

24.7.31 BMLog Buffer End

*Table 259.*0xD8 - BMLBE - GR1553B BM Log buffer end

31 22	21 3	2 0
-	BM LOG BUFFER END	-
	0x0000007	
r	rw	r

31 : 0	Pointer to the highest address of the BM log buffer Only bits 21:3 are settable, i.e. the buffer can not cross a 4 MB boundary Bits 31:22 read the same as the buffer start address.Due to alignment, bits 2:0 are always equal to 1

24.7.32 BMLog Buffer Position

*Table 260.*0xDC - BMLBP - GR1553B BM Log buffer position

31 22	21 3	2 0
-	BM LOG BUFFER POSITION	-
	0x00000000	
r	rw	r

31 : 0 Pointer to the next position that will be written to in the BM log buffer

Only bits 21:3 are settable, i.e. the buffer can not cross a 4 MB boundary Bits 31:22 read the same as the buffer start address.Due to alignment, bits 2:0 are always equal to 0

24.7.33 BM Time Tag Control Register

*Table 261.*0xE0 - BMTTC - GR1553B BM Time tag control register

31 24	23 0
TIME TAG RESOLUTION	TIME TAG VALUE
0	0
rw	rw

31 : 24 Time tag resolution (TRES) - Time unit of BM:s time tag counter in microseconds, minus 1

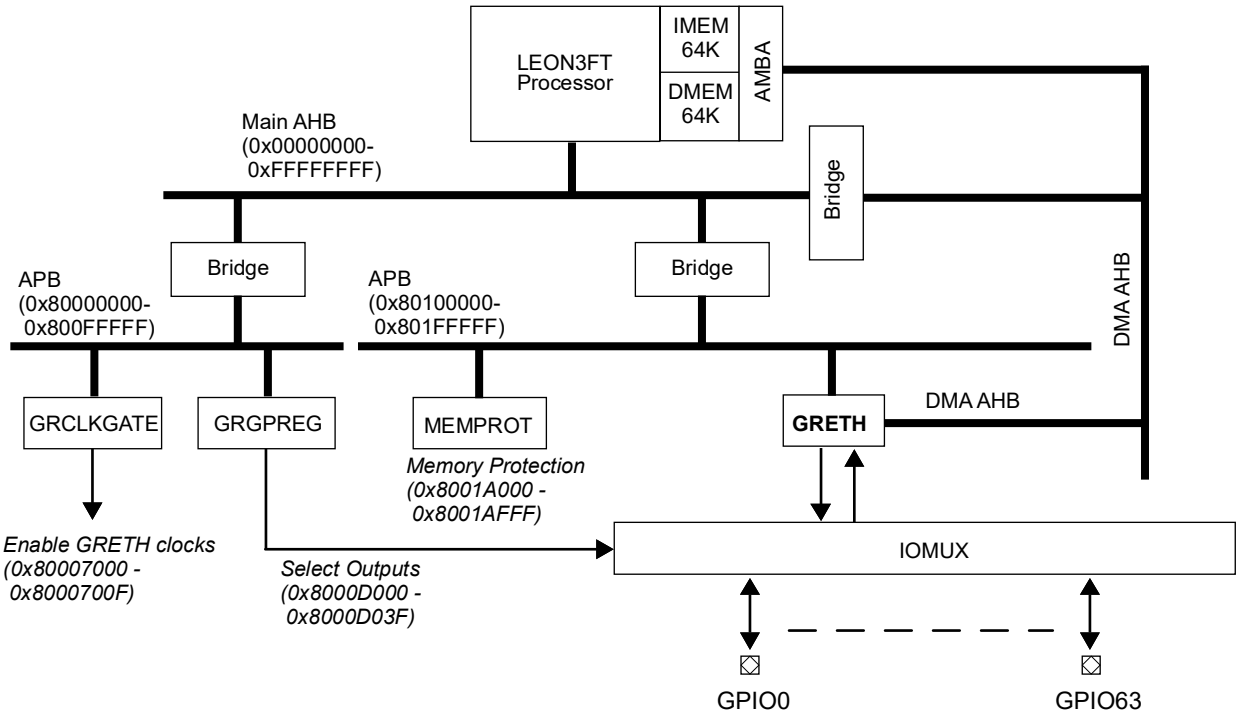
23 : 0 Time tag value (TVAL) - Current value of running time tag counter

25 Ethernet Media Access Controller (MAC)

The GR716B microcontroller contains one Ethernet controller unit. The controller unit controls its own external pins and has a unique AMBA address described in chapter 2.10. The GRETH configuration and status registers are described in section 25.7.

The Ethernet controller units are located on APB bus in the address range from 0x80109000 to 0x80109FFF. See GRETH0 unit connections in the next drawing. The drawing picture provides memory locations and functions used for GRETH configuration and control.

Figure 63. GR716B GRETH bus and pin connection



The primary clock gating unit **GRCLKGATE** described in section 27 is used to enable/disable the GRETH unit. The unit **GRCLKGATE** can also be used to perform reset of the GRETH unit. Software must enable clock and release reset described in section 27 before GRETH configuration and transmission can start.

External IO selection per GRETH unit is made in the system IO configuration register (**GRGPREG**) in the address range from 0x8000D000 to 0x8000D03F. See section 7.1 for further information.

System can be configured to protect and restrict access to individual GRETH units in the **MEMPROT** unit. See section 47 for more information.

25.1 Overview

Frontgrade Gaisler's Ethernet Media Access Controller (GRETH) provides an interface between an AMBA-AHB bus and an Ethernet network. It supports 10/100 Mbit speed in both full- and half-duplex. The AMBA interface consists of an APB interface for configuration and control and an AHB master interface which handles the dataflow. The dataflow is handled through DMA channels. There is one DMA engine for the transmitter and one for the receiver. Both share the same AHB master interface. The ethernet interface supports the MII interfaces which should be connected to an external PHY. The GRETH also provides access to the MII Management interface which is used to configure the PHY.

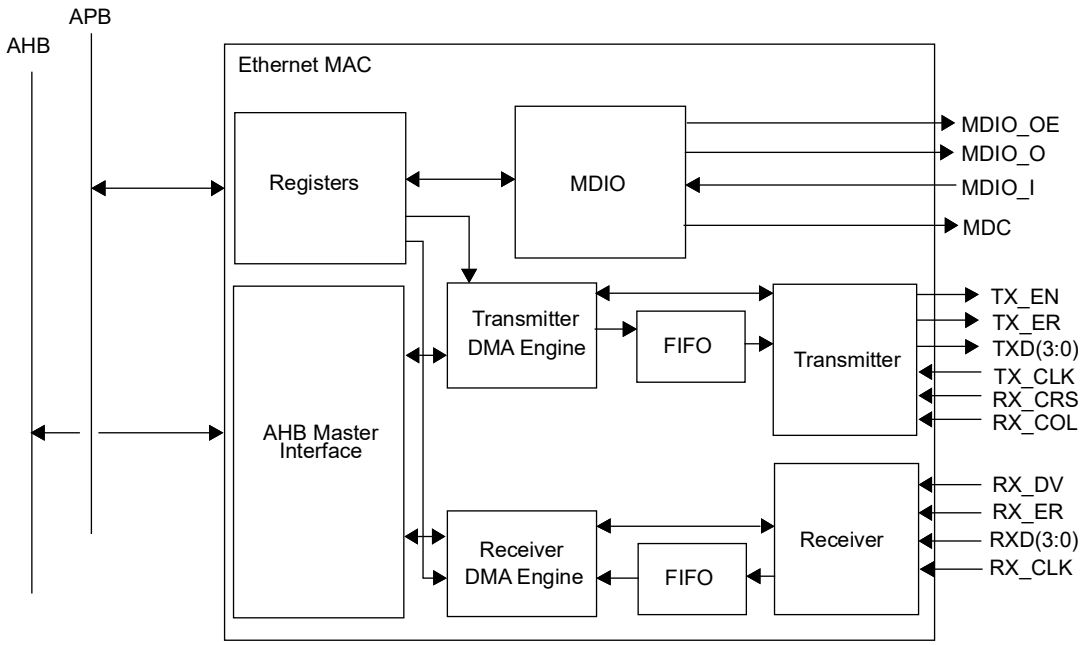


Figure 64. Block diagram of the internal structure of the GRETH.

25.2 Operation

25.2.1 System overview

The GRETH consists of 3 functional units: The DMA channels, MDIO interface.

The main functionality consists of the DMA channels which are used to transfer data between an AHB bus and an Ethernet network. There is one transmitter DMA channel and one Receiver DMA channel. The operation of the DMA channels is controlled through registers accessible through the APB interface.

The MDIO interface is used for accessing configuration and status registers in one or more PHYs connected to the MAC. The operation of this interface is also controlled through the APB interface.

The Media Independent Interface (MII) is used for communicating with the PHY. There is an Ethernet transmitter which sends all data from the AHB domain on the Ethernet using the MII interface. Correspondingly, there is an Ethernet receiver which stores all data from the Ethernet on the AHB bus. Both of these interfaces use FIFOs when transferring the data streams.

25.2.2 Protocol support

The GRETH is implemented according to IEEE standard 802.3-2002 and IEEE standard 802.3Q-2003. There is no support for the optional control sublayer. This means that packets with type 0x8808 (the only currently defined ctrl packets) are discarded.

25.2.3 Clocking

GRETH has three clock domains: The AHB clock, Ethernet receiver clock and the Ethernet transmitter clock. The ethernet transmitter and receiver clocks are generated by the external ethernet PHY, and are inputs to the core through the MII interface. The three clock domains are unrelated to each other and all signals crossing the clock regions are fully synchronized inside the core.

Both full-duplex and half-duplex operating modes are supported and both can be run in either 10 or 100 Mbit. The minimum AHB clock for 10 Mbit operation is 2.5 MHz, while 18 MHz is needed for 100 Mbit. Using a lower AHB clock than specified will lead to excessive packet loss.

25.3 Tx DMA interface

The transmitter DMA interface is used for transmitting data on an Ethernet network. The transmission is done using descriptors located in memory.

25.3.1 Setting up a descriptor.

A single descriptor is shown in table 262 and 263. The number of bytes to be sent should be set in the length field and the address field should point to the data. The address must be word-aligned. If the interrupt enable (IE) bit is set, an interrupt will be generated when the packet has been sent (this requires that the transmitter interrupt bit in the control register is also set). The interrupt will be generated regardless of whether the packet was transmitted successfully or not. The Wrap (WR) bit is also a control bit that should be set before transmission and it will be explained later in this section.

*Table 262.*GRETH transmit descriptor word 0 (address offset 0x0)

31 16	15	14	13	12	11	10 0
RESERVED	AL	UE	IE	WR	EN	LENGTH

31: 16	RESERVED
15	Attempt Limit Error (AL) - The packet was not transmitted because the maximum number of attempts was reached.
14	Underrun Error (UE) - The packet was incorrectly transmitted due to a FIFO underrun error.
13	Interrupt Enable (IE) - Enable Interrupts. An interrupt will be generated when the packet from this descriptor has been sent provided that the transmitter interrupt enable bit in the control register is set. The interrupt is generated regardless if the packet was transmitted successfully or if it terminated with an error.
12	Wrap (WR) - Set to one to make the descriptor pointer wrap to zero after this descriptor has been used. If this bit is not set the pointer will increment by 8. The pointer automatically wraps to zero when the 1 kB boundary of the descriptor table is reached.
11	Enable (EN) - Set to one to enable the descriptor. Should always be set last of all the descriptor fields.
10: 0	LENGTH - The number of bytes to be transmitted.

*Table 263.*GRETH transmit descriptor word 1 (address offset 0x4)

31 2	1 0
ADDRESS	RES

31: 2	Address (ADDRESS) - Pointer to the buffer area from where the packet data will be loaded.
1: 0	RESERVED

To enable a descriptor the enable (EN) bit should be set and after this is done, the descriptor should not be touched until the enable bit has been cleared by the GRETH.

25.3.2 Starting transmissions

Enabling a descriptor is not enough to start a transmission. A pointer to the memory area holding the descriptors must first be set in the GRETH. This is done in the transmitter descriptor pointer register. The address must be aligned to a 1 kB boundary. Bits 31 to 10 hold the base address of descriptor area while bits 9 to 3 form a pointer to an individual descriptor.The first descriptor should be located at the base address and when it has been used by the GRETH the pointer field is incremented by 8 to point at the next descriptor. The pointer will automatically wrap back to zero when the next 1 kB boundary has been reached (the descriptor at address offset 0x3F8 has been used). The WR bit in the descriptors can be set to make the pointer wrap back to zero before the 1 kB boundary.

The pointer field has also been made writable for maximum flexibility but care should be taken when writing to the descriptor pointer register. It should never be touched when a transmission is active.

The final step to activate the transmission is to set the transmit enable bit in the control register. This tells the GRETH that there are more active descriptors in the descriptor table. This bit should always be set when new descriptors are enabled, even if transmissions are already active. The descriptors must always be enabled before the transmit enable bit is set.

25.3.3 Descriptor handling after transmission

When a transmission of a packet has finished, status is written to the first word in the corresponding descriptor. The Underrun Error bit is set if the FIFO became empty before the packet was completely transmitted while the Attempt Limit Error bit is set if more collisions occurred than allowed. The packet was successfully transmitted only if both of these bits are zero. The other bits in the first descriptor word are set to zero after transmission while the second word is left untouched.

The enable bit should be used as the indicator when a descriptor can be used again, which is when it has been cleared by the GRETH. There are three bits in the GRETH status register that hold transmission status. The Transmitter Error (TE) bit is set each time an transmission ended with an error (when at least one of the two status bits in the transmit descriptor has been set). The Transmitter Interrupt (TI) is set each time a transmission ended successfully.

The transmitter AHB error (TA) bit is set when an AHB error was encountered either when reading a descriptor or when reading packet data. Any active transmissions were aborted and the transmitter was disabled. The transmitter can be activated again by setting the transmit enable register.

25.3.4 Setting up the data for transmission

The data to be transmitted should be placed beginning at the address pointed by the descriptor address field. The GRETH does not add the Ethernet address and type fields so they must also be stored in the data buffer. The 4 B Ethernet CRC is automatically appended at the end of each packet. Each descriptor will be sent as a single Ethernet packet.

25.4 Rx DMA interface

The receiver DMA interface is used for receiving data from an Ethernet network. The reception is done using descriptors located in memory.

25.4.1 Setting up descriptors

A single descriptor is shown in table 264 and 265. The address field should point to a word-aligned buffer where the received data should be stored. If the interrupt enable (IE) bit is set, an interrupt will be generated when a packet has been received to this buffer (this requires that the receiver interrupt bit in the control register is also set). The interrupt will be generated regardless of whether the packet was received successfully or not. The Wrap (WR) bit is also a control bit that should be set before the descriptor is enabled and it will be explained later in this section.

*Table 264.*GRETH receive descriptor word 0 (address offset 0x0)

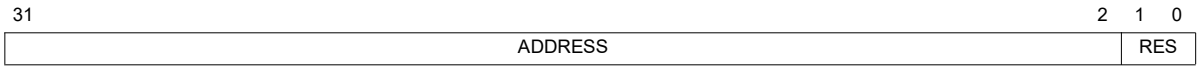
31 27	26	25 19	18	17	16	15	14	13	12	11	10 0
RESERVED	MC	RESERVED	LE	OE	CE	FT	AE	IE	WR	EN	LENGTH

31: 27	RESERVED
26	Multicast address (MC) - The destination address of the packet was a multicast address (not broadcast).
25: 19	RESERVED
18	Length error (LE) - The length/type field of the packet did not match the actual number of received bytes.

Table 264. GRETH receive descriptor word 0 (address offset 0x0)

17	Overrun error (OE) - The frame was incorrectly received due to a FIFO overrun.
16	CRC error (CE) - A CRC error was detected in this frame.
15	Frame too long (FT) - A frame larger than the maximum size was received. The excessive part was truncated.
14	Alignment error (AE) - An odd number of nibbles were received.
13	Interrupt Enable (IE) - Enable Interrupts. An interrupt will be generated when a packet has been received to this descriptor provided that the receiver interrupt enable bit in the control register is set. The interrupt is generated regardless if the packet was received successfully or if it terminated with an error.
12	Wrap (WR) - Set to one to make the descriptor pointer wrap to zero after this descriptor has been used. If this bit is not set the pointer will increment by 8. The pointer automatically wraps to zero when the 1 kB boundary of the descriptor table is reached.
11	Enable (EN) - Set to one to enable the descriptor. Should always be set last of all the descriptor fields.
10: 0	LENGTH - The number of bytes received to this descriptor.

Table 265. GRETH receive descriptor word 1 (address offset 0x4)



31: 2	Address (ADDRESS) - Pointer to the buffer area from where the packet data will be loaded.
1: 0	RESERVED

25.4.2 Starting reception

Enabling a descriptor is not enough to start reception. A pointer to the memory area holding the descriptors must first be set in the GRETH. This is done in the receiver descriptor pointer register. The address must be aligned to a 1 kB boundary. Bits 31 to 10 hold the base address of descriptor area while bits 9 to 3 form a pointer to an individual descriptor. The first descriptor should be located at the base address and when it has been used by the GRETH the pointer field is incremented by 8 to point at the next descriptor. The pointer will automatically wrap back to zero when the next 1 kB boundary has been reached (the descriptor at address offset 0x3F8 has been used). The WR bit in the descriptors can be set to make the pointer wrap back to zero before the 1 kB boundary.

The pointer field has also been made writable for maximum flexibility but care should be taken when writing to the descriptor pointer register. It should never be touched when reception is active.

The final step to activate reception is to set the receiver enable bit in the control register. This will make the GRETH read the first descriptor and wait for an incoming packet.

25.4.3 Descriptor handling after reception

The GRETH indicates a completed reception by clearing the descriptor enable bit. The other control bits (WR, IE) are also cleared. The number of received bytes is shown in the length field. The parts of the Ethernet frame stored are the destination address, source address, type and data fields. Bits 17-14 in the first descriptor word are status bits indicating different receive errors. All four bits are zero after a reception without errors. The status bits are described in table 264.

Packets arriving that are smaller than the minimum Ethernet size of 64 B are not considered as a reception and are discarded. The current receive descriptor will be left untouched an used for the first packet arriving with an accepted size. The TS bit in the status register is set each time this event occurs.

If a packet is received with an address not accepted by the MAC, the IA status register bit will be set.

Packets larger than maximum size cause the FT bit in the receive descriptor to be set. The length field is not guaranteed to hold the correct value of received bytes. The counting stops after the word containing the last byte up to the maximum size limit has been written to memory.

The address word of the descriptor is never touched by the GRETH.

25.4.4 Reception with AHB errors

If an AHB error occurs during a descriptor read or data store, the Receiver AHB Error (RA) bit in the status register will be set and the receiver is disabled. The current reception is aborted. The receiver can be enabled again by setting the Receive Enable bit in the control register.

25.5 MDIO Interface

The MDIO interface provides access to PHY configuration and status registers through a two-wire interface which is included in the MII interface. The GRETH provided full support for the MDIO interface.

The MDIO interface can be used to access from 1 to 32 PHY containing 1 to 32 16-bit registers. A read transfer i set up by writing the PHY and register addresses to the MDIO Control register and setting the read bit. This caused the Busy bit to be set and the operation is finished when the Busy bit is cleared. If the operation was successful the Linkfail bit is zero and the data field contains the read data. An unsuccessful operation is indicated by the Linkfail bit being set. The data field is undefined in this case.

A write operation is started by writing the 16-bit data, PHY address and register address to the MDIO Control register and setting the write bit. The operation is finished when the busy bit is cleared and it was successful if the Linkfail bit is zero.

25.5.1 PHY interrupts

The core also supports status change interrupts from the PHY. A level sensitive interrupt signal can be connected on the mdint input. The PHY status change bit in the status register is set each time an event is detected in this signal. If the PHY status interrupt enable bit is set at the time of the event the core will also generate an interrupt on the AHB bus.

25.6 Media Independent Interfaces

There are several interfaces defined between the MAC sublayer and the Physical layer. The GRETH supports two of them: The Media Independent Interface (MII).

The MII was defined in the 802.3 standard and is most commonly supported. The ethernet interface have been implemented according to this specification. It uses 16 signals.

To support lower speed where the operation and clock frequency of the core and phy remains unchanged i.e. running at 10Mb/s when the controller is configured for 100Mb/s speed enable signals should be created to mimic the desired bit rate.

When operating at 10Mb/s, every byte of the MAC frame is repeated 10 clock periods to achieve the correct bit rate. The core does not take care of this operation and enable signals with toggling frequency of the correct bit rate needs to be created.

25.7 Registers

The core is programmed through registers mapped into APB address space.

*Table 266.*GRETH registers

APB address offset	Register
0x0	Control register
0x4	Status/Interrupt-source register
0x8	MAC Address MSB
0xC	MAC Address LSB
0x10	MDIO Control/Status
0x14	Transmit descriptor pointer
0x18	Receiver descriptor pointer
0x1C	Not Used
0x20	Not used
0x24	Not used

25.7.1 Control Register

*Table 267.*0x00 - CTRL - GRETH control register

31	30 27	26	25	24 15	14	13	12	11	10	9 8	7	6	5	4	3	2	1	0
EA	RESERVED	MA	MC	RESERVED					PI	RES		RS	PM	FD	RI	TI	RE	TE
0	0	1	0	0					0	0		0	0	0	0	0	0	0
r	r	r	r	r					rw	r		wc	rw	rw	rw	rw	rw	rw

31	EDCL available (A) - Set to 0 to indicate that the core does not implement EDCL. Read only.
30: 27	RESERVED
27	RESERVED
26	MDIO interrupts available (MA) - Set to one to indicate that the core supports MDIO interrupts. Read only.
25	Multicast available (MC) - Set to zero to indicate that the core does not support multicast address reception. Read only.
24: 11	RESERVED
10	PHY status change interrupt enable (PI) - Enables interrupts for detected PHY status changes.
9: 7	RESERVED
6	Reset (RS) - A one written to this bit resets the GRETH core. Self clearing. No other accesses should be done .to the slave interface other than polling this bit until it is cleared.
5	Promiscuous mode (PM) - If set, the GRETH operates in promiscuous mode which means it will receive all packets regardless of the destination address. Reset value: ‘0’.
4	Full duplex (FD) - If set, the GRETH operates in full-duplex mode otherwise it operates in half-duplex. Reset value: ‘0’.
3	Receiver interrupt (RI) - Enable Receiver Interrupts. An interrupt will be generated each time a packet is received when this bit is set. The interrupt is generated regardless if the packet was received successfully or if it terminated with an error. Reset value: ‘0’.
2	Transmitter interrupt (TI) - Enable Transmitter Interrupts. An interrupt will be generated each time a packet is transmitted when this bit is set. The interrupt is generated regardless if the packet was transmitted successfully or if it terminated with an error. Reset value: ‘0’.
1	Receive enable (RE) - Should be written with a one each time new descriptors are enabled. As long as this bit is one the GRETH will read new descriptors and as soon as it encounters a disabled descriptor it will stop until RE is set again. This bit should be written with a one after the new descriptors have been enabled. Reset value: ‘0’.
0	Transmit enable (TE) - Should be written with a one each time new descriptors are enabled. As long as this bit is one the GRETH will read new descriptors and as soon as it encounters a disabled descriptor it will stop until TE is set again. This bit should be written with a one after the new descriptors have been enabled. Reset value: ‘0’.

25.7.2 Status Register

*Table 268.*0x04 - STAT - GRETH status register

31 9	8	7	6	5	4	3	2	1	0
RESERVED	PS	IA	TS	TA	RA	TI	RI	TE	RE
	0	0	0	NR	NR	NR	NR	NR	NR
	wc	wc	wc	wc	wc	wc	wc	wc	wc

8 PHY status changes (PS) - Set each time a PHY status change is detected.

7 Invalid address (IA) - A packet with an address not accepted by the MAC was received. Cleared when written with a one. Reset value: ‘0’.

6 Too small (TS) - A packet smaller than the minimum size was received. Cleared when written with a one. Reset value: ‘0’.

5 Transmitter AHB error (TA) - An AHB error was encountered in transmitter DMA engine. Cleared when written with a one. Not Reset.

4 Receiver AHB error (RA) - An AHB error was encountered in receiver DMA engine. Cleared when written with a one. Not Reset.

3 Transmitter interrupt (TI) - A packet was transmitted without errors. Cleared when written with a one. Not Reset.

2 Receiver interrupt (RI) - A packet was received without errors. Cleared when written with a one. Not Reset.

1 Transmitter error (TE) - A packet was transmitted which terminated with an error. Cleared when written with a one. Not Reset.

0 Receiver error (RE) - A packet has been received which terminated with an error. Cleared when written with a one. Not Reset.

25.7.3 MAC Address MSB

*Table 269.*0x08 - MACMSB - GRETH MAC address MSB.

31 16	15 0
RESERVED	Bit 47 downto 32 of the MAC address
	NR
	rw

31: 16 RESERVED

15: 0 The two most significant bytes of the MAC Address. Not Reset.

25.7.4 MAC Address LSB

*Table 270.*0x0C - MACLSB - GRETH MAC address LSB.

31 0
Bit 31 downto 0 of the MAC address

31: 0 The four least significant bytes of the MAC Address. Not Reset.

25.7.5 MDIO ctrl/status Register

*Table 271.*0x10 - MDIO - GRETH MDIO ctrl/status register.

31 16	15 11	10 6	5 4	3	2	1	0
DATA	PHYADDR	REGADDR	RES	BU	LF	RD	WR
0	*	0		0	1	0	0
rw	rw	rw		r	r	rw	rw

31: 16 Data (DATA) - Contains data read during a read operation and data that is transmitted is taken from this field. Reset value: 0x0000.

15: 11 PHY address (PHYADDR) - This field contains the address of the PHY that should be accessed during a write or read operation. Reset value: “00000”.

10: 6 Register address (REGADDR) - This field contains the address of the register that should be accessed during a write or read operation. Reset value: “00000”.

5:4 RESERVED

3 Busy (BU) - When an operation is performed this bit is set to one. As soon as the operation is finished and the management link is idle this bit is cleared. Reset value: ‘0’.

2 Linkfail (LF) - When an operation completes (BUSY = 0) this bit is set if a functional management link was not detected. Reset value: ‘1’.

1 Read (RD) - Start a read operation on the management interface. Data is stored in the data field. Reset value: ‘0’.

0 Write (WR) - Start a write operation on the management interface. Data is taken from the Data field. Reset value: ‘0’.

25.7.6 Transmitter Descriptor Table Base Address Register

*Table 272.*0x14 - TXBASE - GRETH transmitter descriptor table base address register.

31 10	9 3	2 0
BASEADDR	DESCPNT	RES
NR	0	0
rw	rw	r

31: 10 Transmitter descriptor table base address (BASEADDR) - Base address to the transmitter descriptor table.Not Reset.

9: 3 Descriptor pointer (DESCPNT) - Pointer to individual descriptors. Automatically incremented by the Ethernet MAC.

2: 0 RESERVED

25.7.7 Receiver Descriptor Table Base Address Register

*Table 273.*0x18 - RXBASE - GRETH receiver descriptor table base address register.

31 10	9 3	2 0
BASEADDR	DESCPNT	RES
NR	0	0
rw	rw	r

31: 10 Receiver descriptor table base address (BASEADDR) - Base address to the receiver descriptor table.Not Reset.

9: 3 Descriptor pointer (DESCPNT) - Pointer to individual descriptors. Automatically incremented by the Ethernet MAC.

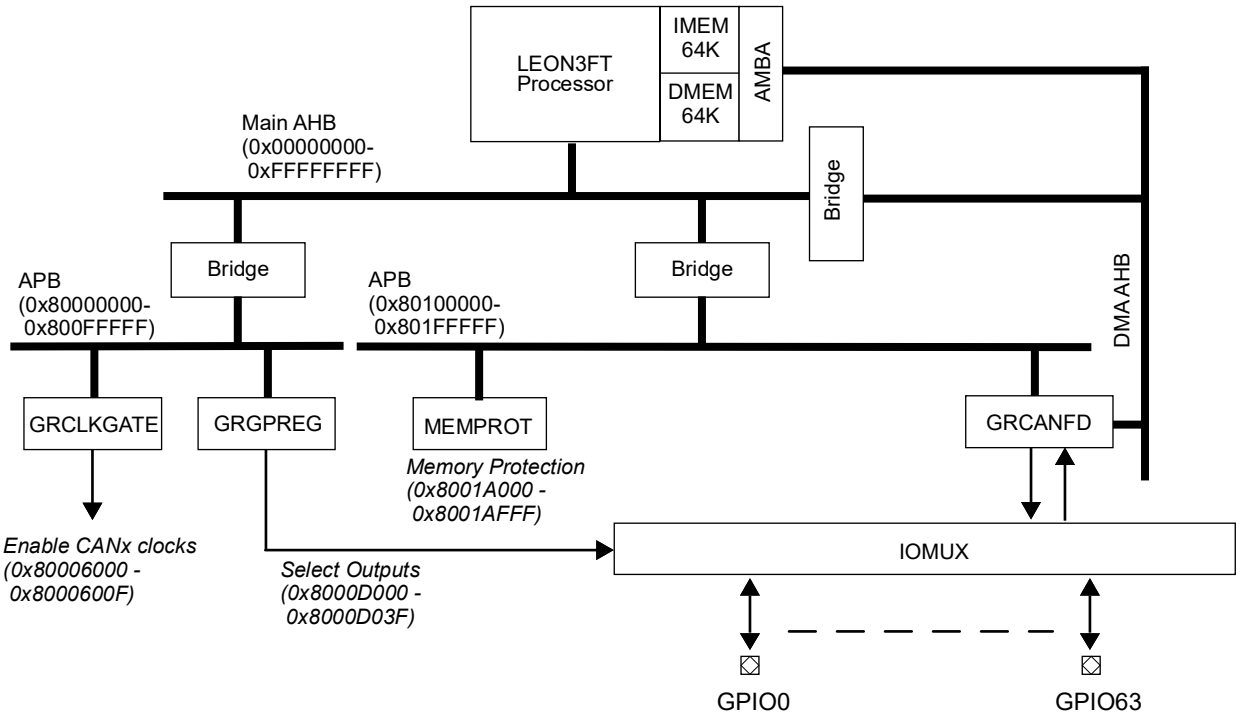
2: 0 RESERVED

26 CAN Flexible Data-Rate Controller with CANOpen support

The GR716B microcontroller comprise a CAN-FD controller (GRCANFD) units.

The CAN-FD controller (GRCANFD) units are located on APB bus in the address range from 0x80102000 to 0x80102FFF . See GRCANFD units connections in the next drawing. The drawing picture memory locations and functions used for GRCANFD configuration and control.

Figure 65. GR716B GRCANFD bus and pin connection



The primary clock gating unit **GRCLKGATE** described in section 27 is used to enable/disable individual GRCANFD units. The unit **GRCLKGATE** can also be used to perform reset of individual GRCANFD units. Software must enable clock and release reset described in section 27 before GRCANFD configuration and transmission can start.

External IO selection per GRCANFD unit is made in the system IO configuration register (**GRGPREG**) in the address range from 0x8000D000 to 0x8000D03F. See section 7.1 for further information.

Each **GRACNFDx** unit controls its own external pins and has a unique AMBA address described in chapter 2.10. Configuration and status registers are described in section 26.11.

System can be configured to protect and restrict access to individual GRCANFD units in the **MEMPROT** unit. See section 47 for more information.

26.1 Overview

GRCANFD implements a CAN-FD controller with a top DMA layer handling the configuration of the controller and the communication between the internal CAN-FD controller and a memory external to the IP.

The internal codec is the CAN-FD controller implementing the MAC and PL sub-layers of the protocol: transmission and reception of frames, error detection and signaling, frame acknowledgment, bit resynchronization, etc. This functionality is compliant with the ISO standard 11898-1:2015 (2nd edition).

Additionally, GRCANFD implements the CANOpen Minimal Set Protocol as per the ECSS-E-ST-50-15C specification, section 9. The controller acts as a slave node (i.e. cannot operate as a CANOpen master). This functionality allows an external CAN node to access the AMBA address space of the device by means of CANOpen PDOs, similarly to RMAP in SpaceWire. The functionality can be configured upon start-up to enable automatic operation in simple applications without need of software intervention. The user can switch between the standard and the CANOpen mode in run-time via the APB interface.

The following block diagram depicts the main components of GRCANFD.

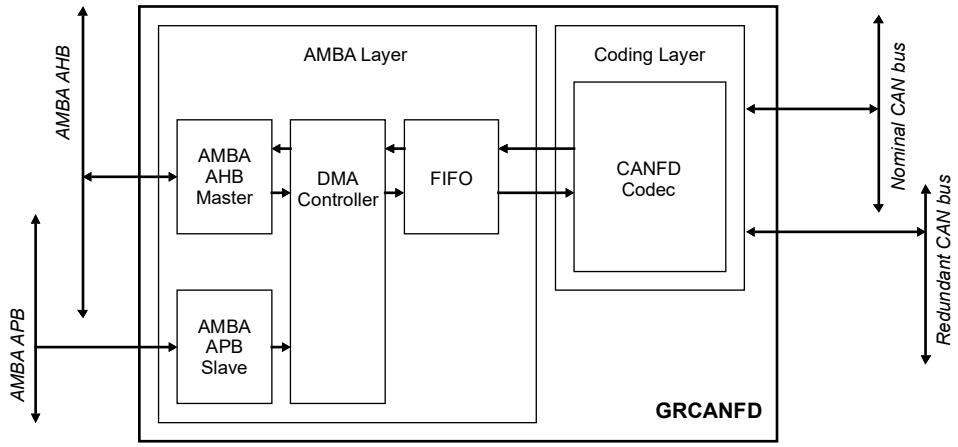


Figure 66. Block diagram

The top AMBA layer contains the memory-mapped registers for the configuration of the IP, accessible through the APB interface. It also controls the global functionality of the controller by fetching frames to be transmitted from the external memory (TX channel) and by storing received frames into the external memory (RX channel) in standard mode, and by processing any input CANOpen commands and providing the reply when applicable in CANOpen mode.

This layer also includes 2 internal SRAMs for the transmit and receive channels. These buffers are used to store the data bytes of a frame, whereas the control bits are stored in registers. The size of each buffer is 64 to 512. Note that these buffers are only used in standard mode, no additional memory is required in CANOpen mode.

The external memory where the frames are stored in standard mode is commonly referred to as the circular buffers, and it may be located on-chip or external to the chip. The controller will write and read from this memory through the master interface. There is a separate buffer for the TX and for the RX channels. Although the default topology is circular, straight buffers are also supported. The content of the circular buffers is handled by pointers.

In standard mode, the frames are temporarily buffered in the internal SRAMs. For transmission, the DMA engine fetches frames from the TX circular buffer and the CAN-FD codec transmits them as soon as the SRAM contains a full frame. Likewise, frames received by the codec are temporarily stored in the internal SRAM, and written to the RX circular buffer as soon as the reception is complete. The communication with the circular buffer takes place via the bus master interface.

In CANOpen mode, GRCANFD receives and acknowledges any incoming frame according to the ISO standard, regardless of its format. The frame is then decoded to identify if it is a PDO, SYNC or Heartbeat command, otherwise it is discarded. The CANOpen master node can therefore issue read and write commands, as well as to trigger interrupts in the slave via the CAN interface.

The controller may have up to 3 separate interrupt lines. The first one is the common line, which comprises interrupts for reception and transmission of frames, update on the status of the controller and events on the CAN bus. The other two interrupt lines are used for SYNC messages (for TX and RX channel, respectively) in standard mode. All interrupts are maskable by accessing the APB registers.

GRCANFD implements bus redundancy: its interface includes two RX and two TX lines. The pair of lines to be used can be selected via the AMBA APB interface. The transmission and reception of frames can only occur on one bus at a time. An additional signal is included to enable or disable the output of the external CAN-FD transceiver.

26.1.1 Function

The core implements the following functions:

- CAN-FD core functionality
- TX channel: frame fetching, buffering and transmission
- RX channel: frame reception, filtering, buffering and storage
- CANOpen interpreter supporting PDOs 1-4, SYNC and Heartbeat commands
- SYNC message detection for both TX and RX channels
- Status and monitoring of the IP
- Interrupt generation
- Redundancy selection

26.2 CAN Interface

The external interface towards the CAN bus features two redundant pairs of transmit output and receive input (i.e. 0 and 1).

The active pair (i.e. 0 or 1) is selectable by means of a configuration register bit. Note that all reception and transmission is made over the active pair.

For each pair, there is one enable output (i.e. 0 and 1), each being individually programmable. The enable outputs can be used for enabling an external physical driver. Note that both pairs can be enabled simultaneously. The polarity for the enable/inhibit inputs on physical interface drivers may differ, thus the meaning of the enable output is undefined.

Redundancy is implemented by means of Selective Bus Access. Note that the active pair selection above provides means to meet this requirement.

26.3 Protocol compliance

The CAN-FD controller included in GRCANFD compliant with the ISO standard for CAN with the FD extension: ISO 11898-1:2015 (2nd edition). The minimal CANOpen implementation is compatible with the ECSS-E-ST-50-15C specification.

26.4 Status and monitoring

The GRCANFD register interface provides status and monitoring data, including:

- Operational mode (standard or CANOpen)
- Ongoing transmission
- Bus-off state
- Error-passive mode
- Overrun during reception
- Transmitter error counter (8 bits)
- Receiver error counter (8 bits)

Some of the previous indicators are also written to the circular buffer every time a frame is received and stored in standard mode. This is described later in 26.5.2. Note that this only applies to the RX channel.

26.5 CAN and CAN-FD frames

26.5.1 Frame formats

As defined in the CAN-FD standard, GRCANFD supports the following formats for the transmission and reception of data frames:

- Classical Base Frame Format (CBFF)
- Classical Extended Frame Format (CEFF)
- FD Base Frame Format (FBFF)
- FD Extended Frame Format (FEFF)

Remote frames are compatible with the classical CAN formats (CBFF and CEFF), but not with the new FD formats (FBFF and FEFF), as per the standard. Error Frames (EF) and Overload Frames (OL) are fully supported too.

The following parameters define the format of a frame. A brief description and additional considerations when working with the controller are included. For a complete overview on the formats and their corresponding fields, please refer to the ISO standard for CAN-FD.

- ID: identifier of the frame. It consists of two parts: the base ID (11 bits) and the extended ID (18 bits, optional). Frames with Base format (CBFF and FBFF) only use the base ID, whereas frames with Extended format (CEFF and FEFF) use both.
- IDE: this parameter selects between Base (0b) and Extended Format (1b). It determines if the extended ID shall be appended to the base ID when building a frame, as described above. This parameter does not depend on whether it is an FD frame or not, so the value of other bits such as FDF or BRS is irrelevant.
- RTR: this parameter selects between Data (0b) and Remote frames (1b). Remote frames are only compatible with classical CAN frames. This means that it must be avoided to set both RTR and FDF to 1b when describing a frame, as the behavior is undefined.
- FDF: this parameter determines if the frame has classical (0b: CBFF and CEFF) or FD format (1b: FBFF and FEFF). The FD format allows to transmit larger payloads of up to 64 bytes, and features a more robust CRC algorithm: CRC-17 or CRC-21, depending on the data length. Optionally, it also enables the possibility of switching to a higher bit-rate. This depends on the configuration of the BRS bit, as described below. When setting FDF to 1b, no Remote frames shall be requested (i.e. RTR should be set to 0b).
- BRS: this bit determines whether the bit-rate shall be switched for the data phase of a frame. It is only meaningful for FD frames (FDF set to 1b). Therefore, the bit is ignored for classical CAN frames (FDF to 0b).
- DLC: it selects the data length of the frame. Its encoding varies depending on whether FD format is used or not. For classical frames, the maximum payload is 8 bytes, whereas FD formats may contain up to 64 bytes of data.

It is important to remark that it is optional to enable the BRS bit for FD frames. FD frames with no bit-rate switching still present the advantage of enabling larger data payloads than in classical CAN frames with more robust CRC algorithms, although the full frame would be transmitted at a constant, nominal bit-rate.

26.5.2 Frame memory mapping (standard mode only)

A descriptor is the minimum unit used to represent a frame. It consists of four 32-bit words. The number of descriptors needed to represent a frame varies from 1 to 5. For classical CAN frames (CBFF and CEFF formats) only one descriptor is needed. For CAN-FD frames (FBFF and FEFF formats) the number of descriptors depends on the data length of the frame: whereas frames with up to 8 bytes only require 1 descriptor, as with classical CAN frames, frames with 64 bytes of data payload require 5 descriptors.

The first descriptor in a frame always includes the control bits describing the frame, as well as information regarding the status of the bus and the IP. This is applicable to both classical CAN and CAN-FD frames. If a frame requires more than one descriptor, the successive descriptors (from 2 up to 5) do not replicate the control and status bits, but only include data bytes. Therefore, each of these descriptors may contain up to 16 bytes of data. Descriptors belonging to the same frame shall always appear consecutively in the circular buffer, taking into account wrap-around conditions.

Each CAN descriptor is aligned to 4 words address boundaries, i.e. the 4 least significant byte address bits are zero for the first word in a CAN descriptor. Note that this frame representation and memory mapping is backwards compatible with GRCAN, which only supports classical CAN format.

Table 274 describes the memory mapping for the first descriptor of a frame, including both control and status bits (first half of the descriptor) and data bytes (second half).

*Table 274.*CAN message representation in memory.

AHB addr	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x0	IDE	RTR	-	bID											eID	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	eID															
0x4	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	DLC				-	FDF	BRS	-	TxErrCntr							
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RxErrCntr								-	-	-	-	BMErr	OR	Off	Pass
0x8	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Byte 0 (first transmitted)								Byte 1							
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Byte 2								Byte 3							
0xC	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Byte 4								Byte 5							
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Byte 6								Byte 7							

Values: Levels according to CAN standard: 1b is recessive, 0b is dominant

Legend: Naming and numbering according to CAN standard

IDE Identifier Extension: 1b for Extended Format (Base + Extended ID), 0b for Standard Format (Base ID)

RTR Remote Transmission Request: 1b for Remote Frame (only classical CAN), 0b for Data Frame

bID Base Identifier
eID Extended Identifier
DLC Data Length Code, according to CAN standard:

0000b	0 bytes
0001b	1 byte
0010b	2 bytes
0011b	3 bytes
0100b	4 bytes
0101b	5 bytes
0110b	6 bytes
0111b	7 bytes
1000b	8 bytes
1001b	12 bytes (only CAN-FD)
1010b	16 bytes (only CAN-FD)
1011b	20 bytes (only CAN-FD)
1100b	24 bytes (only CAN-FD)
1101b	32 bytes (only CAN-FD)
1110b	48 bytes (only CAN-FD)
1111b	64 bytes (only CAN-FD)

FDF Flexible-Data Rate Frame — 1b for FD Format, 0b for Classic Format
BRS Switch Data Bit Rate — 1b for bit-rate switch (only CAN-FD frames), 0b for constant bit-rate

TxErrCntr Transmission Error Counter
RxErrCntr Reception Error Counter
BMErr Bus Master Error during previous accesses to the bus when 1b
OR Reception Overrun when 1b
OFF Bus-Off mode when 1b
PASS Error-Passive mode when 1b
Byte 00 to 07 Transmit/Receive data, Byte 00 first Byte 07 last

Status bits (TxErrCntr, RxErrCntr, BMErr, OR, OFF, PASS) only apply to the RX channel. The controller will write these bits to the circular buffer together with the received frame. For transmission, these bits are not used, therefore they can be simply set to 0b when setting up a frame.

Frames with DLC > 1000b will only result in more than 8 bytes of data for FD frames (FDF set to 1b). For classical CAN frames, the maximum data size is limited to 8 bytes, even if DLC is higher than 1000b. Note also that GRCANFD will ignore the BRS bit if FDF is set to 0b (classical CAN format), so that the Nominal Bit Rate would be used for the entire frame.

If an FD frame contains more than 8 data bytes, it requires additional descriptors to be fully represented in the circular buffer. These descriptors only contain data bytes, and its memory representation is the continuation of the second half of the first descriptor:

- Descriptor #2: data bytes 8 to 23.
- Descriptor #3: data bytes 24 to 39.
- Descriptor #4: data bytes 40 to 55.
- Descriptor #5: data bytes 56 to 63 (second half of the descriptor is unused).

For more information regarding the format of the frame and the content of a descriptor, please refer to 26.5.1 and the CAN-FD ISO standard.

26.6 Operational modes

GRCANFD supports two modes of operation: standard and CANOpen. In standard mode, the controller splits the frame in descriptors, as explained in 26.5.2, and the transmission and reception is controlled by hardware pointers handling the TX and RX circular buffers. In CANOpen mode the

controller becomes a CANOpen interpreter, decoding any incoming frame and reacting accordingly. This mode interacts with neither the circular buffers nor the internal SRAM memories.

The operational mode shall be selected, i.e. GRCANFD cannot operate in standard mode and CANOpen mode simultaneously. This section describes how to set the default mode after reset and how to switch between modes.

Note that the operational mode only affects the top DMA layer of the IP. The internal CAN-FD controller shall handle all frames in any case, regardless of its format.

26.6.1 Default configuration

The interface is by default set to standard mode after reset expect for when remote access via CANOpen is enabled via bootstraps. In remote CANOpen mode CAN timing parameters needs to be configured so that communication can be established automatically without software intervention.The parameters that can be configured are:

- Internal CAN-FD controller enable.
- Active CAN bus selection and transceivers output enable.
- CAN bit time parameters (only for classical bit-rate).
- CANOpen mode enable.
- CANOpen node ID.

For more information about CANOpen bootstrap configuration see section 3.1.

26.6.2 Switching between standard and CANOpen mode

In order to switch between standard and CANOpen mode, several conditions shall be met before the controller can effectively switch to the requested mode. This prevents the controller from entering any unexpected state if the mode is switched during an ongoing operation.

To switch from standard to CANOpen mode, the following conditions shall be met:

- CANOpen mode enabled: set the Enable bit in the CANOpen Control Register to 1b.
- Transmitter disabled: set the Enable bit in the Transmit Channel Control Register to 0b.
- Receiver disabled: set the Enable bit in the Receive Channel Control Register to 0b.

Any ongoing operation shall finalize (i.e. all FSMs related to the TX and RX channels shall be in idle state) before the request to change to CANOpen mode is granted.

To switch from CANOpen to standard mode, the following conditions shall be met:

- CANOpen mode disabled: set the Enable bit in the CANOpen Control Register to 0b.

Any ongoing operation shall finalize (i.e. the CANOpen FSM shall be in idle state) before the request to change to standard mode is granted.

The status of the controller can be monitored at any time by sampling the Mode bit in the Status Register.

26.7 Standard Mode - Transmission

The circular buffer for the transmit channel is defined by the following parameters:

- base address
- buffer size
- write pointer
- read pointer

The transmit channel can be enabled or disabled.

26.7.1 Circular buffer

The transmit channel operates on a circular buffer located in memory external to GRCANFD. The circular buffer can also be used as a straight buffer. The buffer memory is accessed via the AMBA AHB bus master interface.

The size of the buffer is defined by the SIZE field in the Transmission Channel Size Register, specifying the number of CAN descriptors * 4 that fit in the buffer.

E.g. CanTxSIZE.SIZE =2 means 8 CAN descriptors fit in the buffer.

Note however that it is not possible to fill the buffer completely, leaving at least one descriptor position in the buffer empty. This is to simplify wrap-around condition checking.

E.g. CanTxSIZE.SIZE =2 means that 7 CAN descriptors fit in the buffer at any given time.

26.7.2 Write and read pointers

The write pointer (WRITE field in the Transmission Channel Write Register) indicates the position +1 of the last CAN descriptor written to the buffer. The write pointer operates on number of CAN descriptors, not on absolute or relative addresses.

The read pointer (READ field in the Transmission Channel Read Register) indicates the position +1 of the last CAN descriptor read from the buffer. The read pointer operates on number of CAN descriptors, not on absolute or relative addresses.

The difference between the write and the read pointers is the number of CAN descriptors available in the buffer for transmission. The difference is calculated using the buffer size, specified by the CanTxSIZE.SIZE field, taking wrap around effects of the circular buffer into account.

Examples:

- There are 2 CAN descriptors available for transmit when CanTxSIZE.SIZE=2, CanTxWR.WRITE=2 and CanTxRD.READ=0.
- There are 2 CAN descriptors available for transmit when CanTxSIZE.SIZE=2, CanTxWR.WRITE =0 and CanTxRD.READ =6.
- There are 2 CAN descriptors available for transmit when CanTxSIZE.SIZE=2, CanTxWR.WRITE =1 and CanTxRD.READ =7.
- There are 2 CAN descriptors available for transmit when CanTxSIZE.SIZE=2, CanTxWR.WRITE =5 and CanTxRD.READ =3.

When a frame has been successfully transmitted, the read pointer (CanTxRD.READ) is automatically updated, taking wrap-around effects of the circular buffer into account. If a frame consists of more than one descriptor, the pointer is not incremented one by one, but it is updated to the next descriptor to be read from the buffer.

Whenever the write pointer CanTxWR.WRITE and read pointer CanTxRD.READ are equal, there are no CAN descriptors available for transmission.

26.7.3 Location

The location of the circular buffer is defined by a base address in Transmission Channel Address Register, which is an absolute address. The location of a circular buffer is aligned on a 1kbyte address boundary.

26.7.4 Transmission procedure

When the channel is enabled (ENABLE bit in the Transmission Channel Control Register equal to 1b), as soon as there is a difference between the write and read pointer, GRCANFD will start fetching the first descriptor of a frame, whose position is indicated by the read pointer. If a frame consists of more than one descriptor, GRCANFD will continue fetching them until the full frame has been read.

GRCANFD will decode the DLC field, available in the first descriptor, in order to know how many descriptors shall be fetched in total.

The one frame can be buffered by the core prior to transmission. The data bytes of a frame are stored in the internal TX SRAM, whereas the control bits are stored in registers. Note that the core will fetch one frame regardless of their data length. The core will stop fetching frames if there are no more frames initialized in the circular buffer.

Note that the TX channel should not be enabled if a potential difference between the write and read pointers could be created, to avoid the descriptor fetching to start prematurely. The TX write pointer shall not be updated until all the descriptors forming the frame are ready to be fetched from the circular buffer.

The read pointer (CanTxRD.READ) is automatically updated after a successful transmission, taking into account wrap around effects of the circular buffer. The content of the local SRAM may then be overwritten, so a new frame would be fetched as soon as the write and read pointers differ.

If the single-shot mode is enabled for the TX channel (CanTxCTRL.SINGLE=1), any frame for which the arbitration is lost, or failed for some other reason such as ACK missing, will lead to the TX channel being automatically disabled (CanTxCTRL.ENABLE=0). The frame will not be put up for re-arbitration and the local SRAM will be emptied, in order not to block future transmissions.

Interrupts are provided to aid the user during transmission, as described in detail later in this section. The main interrupts are the Tx, TxEmpty and TxIrq which are issued respectively after the successful transmission of a frame, when all frames in the circular buffer have been transmitted and when a predefined number of descriptors have been transmitted. The TxLoss interrupt is asserted whenever a transmission does not complete: examples of this are loss of arbitration or communication errors. The TxSync interrupt is issued when a frame matching the SYNC pattern is successfully transmitted. Additional interrupts are provided to signal error conditions on the CAN bus and the AMBA bus.

26.7.5 AMBA error

An error response occurring on the AMBA bus while a frame descriptor is being fetched will result in a BmRdErr interrupt. Consequently the bit BMErr of the Status Register is set to 1b, and can only be cleared by reading that register.

If the CanCONF.ABORT bit is set to 0b, GRCANFD will automatically try to fetch the same frame descriptor again.

If the CanCONF.ABORT bit is set to 1b, the TX channel will be disabled (ENABLE bit in the Transmission Channel Control register is cleared automatically to 0b). The read pointer can be used to determine which frame caused the bus master error (but not the specific descriptor, if the frame contains more than one). If the error occurs while the other frame in the local SRAM is being transmitted, the transmission is not interrupted, in order not to disrupt the CAN bus. If the transmission is acknowledged, the pointers are updated accordingly, even if the channel is already disabled.

26.7.6 Enable and disable

When the TX channel is disabled (ENABLE bit in the Transmission Channel Control Register cleared to 0b) during an ongoing transmission, the internal CAN-FD codec will not abort the transmission, but attempt to finish it until the frame is acknowledged, the arbitration is lost or any error is detected. If the frame is sent successfully, the read pointer (CanTxRD.READ) is automatically incremented. Any associated interrupt will be generated.

The progress of any ongoing access can be observed via the ONGOING bit in the Transmission Channel Control Register. The ONGOING bit must be 0b before the channel can be re-configured safely (i.e. changing address, size or read pointer). It is also possible to monitor the Tx and TxLoss interrupts described hereafter.

GRCANFD includes a status bit in the TX channel control register called DisACK. This bit is used to indicate that the TX channel disable request has been acknowledged, and it will take effect as soon as the ongoing transmission finishes or fails to complete. The TX channel is not completely disabled until both ENABLE and DisACK are both 0b.

The channel can be re-enabled again without the need to re-configure the address, size and pointers.

Priority inversion is handled by disabling the transmitting channel, i.e. setting CanTxCTRL.ENABLE=0b as described above, and observing the progress, i.e. reading via the CanTxCTRL.ONGOING and CanTxCTRL.DisACK bits as described above. When the transmit channel is disabled, it can be re-configured and a higher priority frames can be transmitted. Note that the single shot mode does not require the channel to be disabled, but the progress should still be observed as above.

26.7.7 Interrupts

During transmission several interrupts can be generated:

- TxLoss: Frame transmission interrupted due to lost of arbitration, ACK not detected by the transmitter or errors during the transmission.
- TxErrCntr: Increment of the transmitter error counter.
- TxSync: Frame matching the SYNC filter transmitted.
- Tx: Successful transmission of a frame.
- TxEmpty: Successful transmission of all the frames in the circular buffer.
- TxIrq: Successful transmission of a predefined number of frame descriptors. One of the descriptors matches the position programmed by the Transmit Channel Interrupt Register.
- BmRdErr: Bus error while fetching a descriptor via the bus master interface.
- Off: Bus-off condition.
- Pass: Error-passive condition.

The Tx, TxEmpty and TxIrq interrupts are only generated as the result of a successful frame transmission, only once the CanTxRD.READ pointer has been incremented.

26.8 Standard Mode - Reception

The receive channel is defined by the following parameters:

- base address
- buffer size
- write pointer
- read pointer

The receive channel can be enabled or disabled.

26.8.1 Circular buffer

The reception channel operates on a circular buffer located in memory external to GRCANFD. The circular buffer can also be used as a straight buffer. The buffer memory is accessed via the bus master interface.

The size of the buffer is defined by the SIZE field in the Reception Channel Size Register, specifying the number of CAN descriptors * 4 that fit in the buffer.

E.g. CanRxSIZE.SIZE =2 means 8 CAN descriptors fit in the buffer.

Note however that it is not possible to fill the buffer completely, leaving at least one descriptor position in the buffer empty. This is to simplify wrap-around condition checking.

E.g. CanRxSIZE.SIZE =2 means that 7 CAN descriptors fit in the buffer at any given time.

26.8.2 Write and read pointers

The write pointer (WRITE field in the Reception Channel Write Register) indicates the position +1 of the last CAN descriptor written to the buffer. The write pointer operates on number of CAN descriptors, not on absolute or relative addresses.

The read pointer (READ field in the Reception Channel Read Register) indicates the position +1 of the last CAN descriptor read from the buffer. The read pointer operates on number of CAN descriptors, not on absolute or relative addresses.

The difference between the write and the read pointers is the number of CAN descriptors available in the buffer for read-out after having received a set of frames. The difference is calculated using the buffer size, specified by the CanRxSIZE.SIZE field, taking wrap around effects of the circular buffer into account.

Examples:

- There are 2 CAN descriptors available for read-out when CanRxSIZE.SIZE=2, CanRxWR.WRITE=2 and CanRxRD.READ=0.
- There are 2 CAN descriptors available for read-out when CanRxSIZE.SIZE=2, CanRxWR.WRITE =0 and CanRxRD.READ=6.
- There are 2 CAN descriptors available for read-out when CanRxSIZE.SIZE=2, CanRxWR.WRITE =1 and CanRxRD.READ=7.
- There are 2 CAN descriptors available for read-out when CanRxSIZE.SIZE=2, CanRxWR.WRITE =5 and CanRxRD.READ=3.

When a frame has been successfully received and all its descriptors have been stored into the circular buffer, the write pointer (CanRxWR.WRITE) is automatically updated, taking wrap-around effects of the circular buffer into account. If a frame consists of more than one descriptor, the pointer is not incremented one by one, but it is updated to the next position to be written to the buffer.

Whenever the read pointer CanRxRD.READ equals (CanRxWR.WRITE+1) modulo (CanRxSIZE.SIZE*4), there is no space available for storing another descriptor. This is signaled by an interrupt (RxFull). If a frame consists of more than one descriptor, GRCANFD will attempt to store each descriptor as long as there is space in the buffer. If there is no space for the next descriptor, the interrupt will be asserted, and GRCANFD will wait until the read pointer is updated to continue storing the frame.

26.8.3 Reception procedure

When the channel is enabled (ENABLE bit in the Reception Channel Control Register equal to 1b), GRCANFD will evaluate whether any incoming frame shall be stored into the local SRAM. To store a frame, the frame shall match the acceptance filter, which is programmable through the Acceptance Code and Acceptance Mask registers, and there shall be space available in the RX SRAM.

Once a complete frame is available, GRCANFD will split it into descriptors, as many as needed to represent the full frame. Then the controller will attempt to write each descriptor to the circular buffer, as long as the write and read pointers differ. This process will continue until the full frame is written to the buffer, which is signaled by the Rx interrupt and the write pointer being updated (CanRxWR.WRITE).

By having a local SRAM with capacity for multiple CAN-FD frames, GRCANFD minimizes the risk of missing frames due to conflicts accessing the AH bus or due to the RX buffer being full. Once a

frame is written to the circular buffer and the write pointer is updated, the content of the SRAM can be replaced with a new frame.

Note that the channel should not be enabled until the write and read pointers are configured, to avoid the message reception to start prematurely.

The core supports generation of Overload Frames when the internal buffer is full in order to request other nodes to delay the next transmission. The core will transmit Overload Frames if the Overload Frames bit is set in the Receive Channel Control Register and the buffer is full. Up to 2 consecutive Overload Frames will be issued, according to the CAN-FD standard.

If a new frame is received before there being space available in the SRAM, the frame cannot be internally stored and the Overrun condition is detected and signaled via the corresponding interrupt (OR) and the Overrun bit in the Status Register. The OR interrupt is raised only upon the reception of the first frame which causes Overrun. If more frames are received, the OR interrupt is not raised until the Overrun condition is resolved.

Interrupts are provided to aid the user during reception, as described in detail later in this section. The main interrupts are Rx and RxIrq which are issued on the successful reception of a frame and when a predefined number of frame descriptors have been received successfully. The RxMiss interrupt is asserted whenever a frame has been received but does not match any filters (i.e. neither the acceptance nor the SYNC filters) or when a frame is received by the codec but the RX channel is disabled and consequently the frame is not stored in the circular buffer.

The RxFull interrupt is raised when there is no space in the circular buffer for storing the next descriptor. This does not only occur when a complete frame has been received and stored, but also when an individual descriptor has been stored and there is no space in the RX buffer for the following one belonging to the same frame (only applicable to FD frames). In this case, the RX Write Pointer would not represent the position currently being written, as it is only updated after all the descriptors of the frame have been stored.

The RxSync interrupt is issued when a frame matching the SYNC filter has been successfully received. Note that the frame does not need to match the acceptance filter. The interrupt is generated as soon as the CAN-FD controller finishes receiving the frame. Additional interrupts are provided to signal error conditions on the CAN bus and AMBA bus.

Regardless of the status of the RX channel, the internal CAN-FD codec will check for errors and acknowledge any incoming frames. The codec shall also evaluate frames with any identifier, even if they do not match the Acceptance or SYNC filters. The only way to prevent the internal codec from checking and acknowledging any incoming frames is to disable it via the Control Register (ENABLE bit).

26.8.4 AMBA error

An error response occurring on the AMBA bus while a frame descriptor is being stored will result in a BmWrErr interrupt. Consequently the bit BMErr of the Status Register is set to 1b, and it can only be cleared when that register is read.

If the CanCONF.ABORT bit is set to 0b, GRCANFD will retry to write the same descriptor to the circular buffer.

If the CanCONF.ABORT bit is set to 1b, the RX channel will be disabled (CanRxCTRL.ENABLE is cleared automatically to 0b). The write pointer can be used to determine which frame caused the bus master error (but not the specific descriptor, if the frame contains more than one). Any ongoing reception is aborted, and the local SRAM is emptied. The descriptor causing the error is not stored and the write pointer is therefore not updated.

26.8.5 Enable and disable

When the RX channel is disabled (ENABLE bit in the Reception Channel Control Register cleared to 0b) while a frame is being written to the circular buffer, the core will still complete this operation before effectively switching off the channel, thus updating the write pointer (CanRxWR.WRITE) and generating the corresponding interrupts.

When the Receive Channel is disabled, the content of the internal SRAM is emptied once any pending operations are finalized, so that the buffer is fully available once the channel is enabled again.

The progress of an ongoing reception can be observed via the ONGOING bit in the Reception Channel Control Register. The ONGOING bit must be 0b before the channel can be re-configured safely (i.e. changing address, size or write pointer).

GRCANFD includes a status bit in the RX channel control register called DisACK. This bit is used to indicate that the RX channel disable request has been acknowledged, and will take effect as soon as the ongoing reception finishes and the frame is stored into the circular buffer, or until it fails to complete. The RX channel is not completely disabled until both ENABLE and DisACK are both 0b.

The channel can be re-enabled again without the need to re-configure the address, size and pointers.

26.8.6 Interrupts

During reception several interrupts can be generated:

- RxMiss: Frame filtered away during reception
- RxErrCntr: Receive error counter incremented.
- RxSync: Frame matching the SYNC filter received.
- Rx: Successful reception of a frame and storage into the circular buffer.
- RxFull: RX buffer full; no space for the next frame descriptor.
- RxIrq: Successful reception of a predefined number of frame descriptors. One of the descriptors of the frame matches the position programmed by the Receive Channel Interrupt Register.
- BmWrErr: Bus error while storing a frame descriptor via the bus master interface.
- OR: Overrun during reception.
- OFF: Bus-off condition.
- PASS: Error-passive condition.

The Rx and RxIrq interrupts are only generated as the result of a successful frame reception, after the CanRxWR.WRITE pointer has been incremented. RxFull may be generated when a descriptor is stored, regardless of whether it is the last one within a frame or not.

The OR interrupt is generated when a frame is received but the internal SRAM already contains the maximum number of frames yet to be stored into the circular buffer. This may be due to a conflict accessing the bus via the bus master interface, or to the RX circular buffer being full. The assertion of this interrupt implies that one or more frames have already been missed. Once this occurs, the Overrun bit of the Status Register is set to 1b, and will only be cleared by reading the mentioned register.

26.9 CANOpen mode

26.9.1 General

GRCANFD features a minimal implementation of the CANOpen standard. This allows an external CAN node to access the internal address space of the device via specific messages defined in CANOpen.

The controller contains an FSM to handle CANOpen commands. GRCANFD will not transmit any frame spontaneously, it shall be triggered by the master node of the bus. Once a CAN frame is received, the controller will check its format and decode it. The frame shall be a classical CAN data frame, i.e. the bits RTR, FDF and BRS shall be all 0b. Frames not matching this criteria are silently discarded.

A separate bus master interface is used for all AMBA accesses performed in CANOpen mode.

When decoding the frame, the controller checks the function code (bits 11-8 of the base ID) and the destination node ID (bits 7-0 of the base ID) of the command. If the command is supported by the IP, the corresponding process is triggered, otherwise the command is silently discarded. Supported commands include PDOs 1-4, SYNC and Heartbeat messages. A detailed description of each command and the usage of their data bytes can be found later in this section.

The PDOs 1 and 3 are mapped to write commands, in which the user shall indicate the target AMBA address and the data to write. The controller supports write accesses of 1-4 bytes, depending on the DLC field of the frame. Once the internal write access to the AMBA bus is finalized, the FSM returns to idle state and an interrupt is generated.

The PDOs 2 and 4 are mapped to read commands, in which the user shall indicate the target AMBA address and, optionally, the number of bytes to read. The controller supports read accesses of 0-8 bytes. If no reply length is indicated, the default access is 4 bytes. Once the internal read access to the AMBA bus is finalized, GRCANFD transmits the reply with the requested data. The FSM returns to idle state when the codec has successfully transmitted the frame, and an interrupt is generated.

For all PDOs, the controller supports the reception of RPDOs and, for read commands, the transmission of the corresponding TPDO with the reply.

The Heartbeat command is used to keep alive the bus. It enabled, GRCANFD features an internal counter that is incremented every CAN bit time (i.e. with the Sample Point of the CAN bus). Every time a Heartbeat command is received, the counter is reset. If the counter reaches the programmed timeout, the controller will generate an interrupt and, optionally, switch to the alternate CAN bus, by inverting the LineSel bit in the Configuration Register. This functionality can be enabled in the CANOpen Control Register with the Redundant Control bit.

The SYNC command is used to synchronize all nodes in the network. When GRCANFD receives a SYNC command, an interrupt is generated. Software is required to handle the node synchronization in this case.

If GRCANFD receives a frame which does not pass the preliminary format check (e.g. the frame has FD format or is a Remote Frame) or it does not match any CANOpen object supported, it generates the RxMiss interrupt and the frame is discarded. Then, the controller starts listening to the CAN bus again waiting for the next frame.

26.9.2 Write commands

Write commands are mapped to the CANOpen PDOs 1 and 3, and are used to trigger an internal AMBA write access. Both PDOs will result in the same functionality. The format of the incoming PDOs shall be as follows:

- Frame format: Classical CAN data frame.
- Function code: RPDO1 (0100b) or RPDO3 (1000b).
- Destination node: GRCANFD node ID (node ID field in the CANOpen Control Register).
- DLC: 5 - 8

The DLC field is used to indicate the size of the AMBA write access. The controller supports write accesses of 1 - 4 bytes per PDO. The AMBA address is mapped to the data bytes 1 - 4 of the frame (the first byte being the MSB and the fourth byte the LSB), whereas the data bytes 5 - 8 represent the data to write (the fifth byte being the MSB and the last byte the LSB).

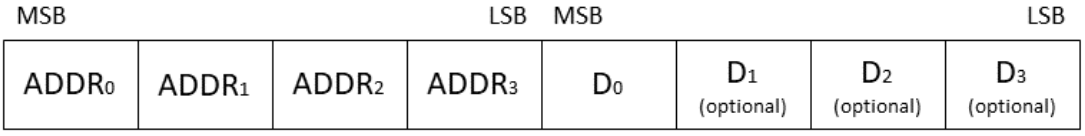


Figure 67. Use of data bytes in write commands (RPDO1 and RPDO3)

Any write command with less than 5 data bytes is simply ignored, as it would not contain enough information to trigger an internal AMBA write access.

26.9.3 Read commands

Read commands are mapped to the CANOpen PDOs 2 and 4, and are used to trigger an internal AMBA read access. Both PDOs will result in the same functionality. The format of the incoming PDOs shall be as follows:

- Frame format: Classical CAN data frame.
- Function code: RPDO2 (0110b) or RPDO4 (1010b).
- Destination node: GRCANFD node ID (node ID field in the CANOpen Control Register).
- DLC: 4 or 5

The size of the AMBA read access is configurable. If the frame contains 4 data bytes (DLC = 4), a standard 32-bit read access is performed. The AMBA address is mapped to these first 4 data bytes of the frame (the first byte being the MSB).

GRCANFD also supports AMBA read accesses of configurable size by means of frames with 5 data bytes (DLC = 5). The four data bytes are still the AMBA address, whereas the fifth byte indicates the size of the read access in number of bytes. Valid sizes range from 0 to 8 bytes, the controller will cap the value to the maximum of 8 bytes.

Figure 68. Use of data bytes in read commands (RPDO2 and RPDO4)

Any read command with DLC other than 4 or 5 is silently discarded by the IP.

Once GRCANFD has completed the internal AMBA read access, it will transmit a TPDO with the data requested as soon as the intermission time (3 bits of nominal CAN bit-rate) is over. The main characteristics of the reply are listed below:

- Frame format: Classical CAN data frame.
- Function code: TPDO2 (0101b) if replying to an RPDO2, or TPDO4 (1001b) for an RPDO4.
- Destination node: GRCANFD node ID (node ID field in the CANOpen Control Register).
- DLC: 0 - 8, according to the size field of the read command (4 by default).

The first data byte of the reply TPDO is the MSB and the last byte, the LSB. Note that it is possible to request the transmission of a TPDO without any data bytes by setting the size of the read access to 0. In this case, no internal read access is performed, and the DLC of the reply is set to 0 accordingly.

26.9.4 SYNC commands

The SYNC commands are employed to synchronize the slave nodes in a CAN bus. Once a frame is interpreted as a SYNC command, GRCANFD triggers an interrupt to be handled by software. The

controller does not maintain any internal synchronization timer. The format of the incoming SYNC command is as follows:

- Frame format: Classical CAN data frame.
- Function code: SYNC (0001b).
- Destination node: broadcast (0000000b).

Note that the DLC and the data bytes of a SYNC command are neither checked nor used by the IP.

26.9.5 Heartbeat commands

The Heartbeat command is used as the NMT Error Control mechanism to keep the bus alive. GRCANFD features an internal counter incremented once per CAN bit time if the NMT Error Control functionality is enabled, which is done by setting the CANOpen Heartbeat Timeout Register to a value different from zero. The timeout is expressed in number of CAN bit times.

Every time a Heartbeat command is received, GRCANFD restarts the counter. If the timeout is reached, an interrupt is triggered and the controller may switch to the alternate bus by inverting the LineSel signal of the Configuration Register. This behavior is controlled via the Redundant Control bit in the CANOpen Control Register.

The format of the Heartbeat command shall be as follows:

- Frame format: Classical CAN data frame.
- Function code: NMT Error Control / Heartbeat (1110b).
- Destination node: GRCANFD node ID (node ID field in the CANOpen Control Register).

Note that the DLC and the data bytes of a Heartbeat command are neither checked nor used by the IP.

26.9.6 AMBA errors

As in the standard mode, the CANFD core will raise an interrupt if any access to the internal AMBA bus ends with errors. The CANOpen mode uses the same Bus Master Error interrupt bits as the standard mode. Therefore, if an AMBA write access triggered by a CANOpen write command ends with errors, the controller will raise the BmWrErr interrupt. On the other hand, if the errors are observed during a read access triggered by a CANOpen read command, the corresponding BmRdErr interrupt is raised.

The CANFD does not use the Abort bit of the Configuration Register, unlike the standard mode. In CANOpen mode, AMBA accesses are never retried (i.e. this corresponds to Abort bit set to 1b) in order to prevent the controller from locking if an external node targeted an invalid AMBA address. Therefore, the internal CANOpen FSM will automatically return to idle after any bus master errors without processing the command. The corresponding interrupt is generated as explained in the paragraph above.

26.9.7 Interrupts

GRCANFD includes four maskable interrupts exclusive to the CANOpen mode:

- CoHbErr: CANOpen Heartbeat timeout error
- CoSync: SYNC command received
- CoWrCmd: CANOpen write command processed
- CoRdCmd: CANOpen read command processed

Note that the CANFD can still produce other general interrupts in CANOpen mode, such as those related to the CAN error counters, loss of arbitration or bus master errors. More information can be found under the Register section, in 26.11.28.

26.10 CANFD reset and enable

When the RESET bit in the Control Register is set to 1b, a reset of the whole controller is performed, including the internal CAN-FD codec. The reset clears all the register fields to their default values. Any ongoing frame transfer request will be aborted, potentially violating the CAN protocol.

When the ENABLE bit in the Control Register is cleared to 0b, the CAN-FD codec is reset, so it is safe to modify the configuration registers. When disabled, the CAN-FD controller will be in sleep mode. It will only send recessive bits thus not affecting the CAN bus. It will neither receive nor acknowledge any frame on the bus.

Once the codec is enabled again, it will enter bus integration state before being able to transmit or receive frames. It requires that 11 consecutive recessive bits are detected in the CAN bus prior to starting the normal operation.

26.11 Registers

The core is programmed through registers mapped into APB address space.

*Table 275.*GRCANFD registers

APB address offset	Register
0x000	Configuration Register
0x004	Status Register
0x008	Control Register

Table 275. GRCANFD registers

APB address offset	Register
0x00C	Capability Register
0x018	SYNC Mask Filter Register
0x01C	SYNC Code Filter Register
0x040	Nominal Bit-Rate Configuration Register
0x044	Data Bit-Rate Configuration Register
0x048	Transmitter Delay Compensation Register
0x080	CANOpen Control Register
0x084	CANOpen Heartbeat Timeout Register
0x088	CANOpen Heartbeat Count Register
0x08C	CANOpen Status Register
0x100	Pending Interrupt Masked Status Register
0x104	Pending Interrupt Masked Register
0x108	Pending Interrupt Status Register
0x10C	Pending Interrupt Register
0x110	Interrupt Mask Register
0x114	Pending Interrupt Clear Register
0x200	Transmit Channel Control Register
0x204	Transmit Channel Address Register
0x208	Transmit Channel Size Register
0x20C	Transmit Channel Write Register
0x210	Transmit Channel Read Register
0x214	Transmit Channel Interrupt Register
0x300	Receive Channel Control Register
0x304	Receive Channel Address Register
0x308	Receive Channel Size Register
0x30C	Receive Channel Write Register
0x310	Receive Channel Read Register
0x314	Receive Channel Interrupt Register
0x318	Receive Channel Acceptance Mask Register
0x31C	Receive Channel Acceptance Code Register

26.11.1 Configuration Register

Table 276. 0x000 - CONF - Configuration Register

31 8	7	6	5	4	3	2	1	0
Reserved	LBS	LB	Res	Silent	Select	Enable1	Enable0	Abort
0x000000	0	0	0	0	0*	0*	0*	0
r	rw	rw	r	rw	rw	rw	rw	rw

7: LBS Loop-back selector. Selects between external (0b) and internal (1b) loop-back. Used only if LB is set.
6: LB Loop-back mode. When transmitting a frame, the core will drive the ACK bit and store its own frames.
4: SILENT Listen only to the CAN bus, send recessive bits.
3: SELECT Line selector for transmitter and receiver:
Select receiver input 0 and transmitter output 0 as active when 0b,
Select receiver input 1 and transmitter output 1 as active when 1b.

2: ENABLE1 Set value of output 1 enable. Its polarity depends on the physical transceiver.
1: ENABLE0 Set value of output 0 enable. Its polarity depends on the physical transceiver.
0: ABORT Abort transfer after AMBA bus errors (standard-mode only).

* The reset value of these fields may be overwritten with the input configuration signals (cfg record) in *canopen* mode.

By setting the ABORT bit to 1b, the TX and RX channels are automatically disabled upon detection of an AMBA bus error. Otherwise, the transfer causing the error is issued again. Note that in CANOpen mode, any AMBA accesses resulting in errors are always aborted, regardless of the value of this bit. This prevents any external CAN node from locking the controller if accessing an invalid or protected AMBA address.

When the loop-back mode is enabled, GRCANFD will drive the ACK bit of its own frames. Additionally, these frames will be stored in the RX circular buffer as received frames. The loop-back selector allows to connect the TX and RX bits inside the IP, thus bypassing the external CAN transceiver (internal loop-back), and to have a standard connection to the CAN bus (external loop-back).

The core will only transmit recessive bits over the CAN bus when either the internal loop-back or the Silent mode is activated.

26.11.2 Status Register

*Table 277.*0x004 - STAT - Status Register

31	24	23	16
Reserved		TxErrCntr	
0x00		0x00	
r		r	

15	8	7	6	5	4	3	2	1	0
RxErrCntr		Reserved		MD	Active	BM Err	OR	Bus Off	Err Pass
0x00		0x0		0	0	0	0	0	0
r		r		r	r	r	r	r	r

23-16: TxErrCntr Transmission error counter, 8-bit
15-8: RxErrCntr Reception error counter, 8-bit
5: MD GRCANFD operational mode: standard (0b) or CANOpen (1b)
4: ACTIVE Transmission ongoing
3: BMErr Errors detected during a previous transfer via the bus master interface
2: OR Overrun during reception
1: Bus Off Bus-off condition
0: Err Pass Error-passive condition

The OR bit is set if a frame with an ID matching the acceptance filter cannot be stored within the local SRAM due to lack of space, i.e. the SRAM already is full of CAN messages yet to be stored into the circular buffer

The OR and BMErr status bits are cleared when the Status Register is read.

Note that TxErrCntr and RxErrCntr are defined and updated according to the CAN protocol.

Additionally, the fields TxErrCntr, RxErrCntr, BMErr, OR, Bus Off and Error Passive are stored in the circular buffer during reception as part of the first descriptor of every frame.

The MD bit indicates the mode in which GRCANFD operates. This bit only changes once all conditions to switch from one mode to the other are satisfied.

26.11.3 Control Register

*Table 278.*0x008 - CTRL - Control Register

31	2	1	0
Reserved		Reset	Enable
0x00000000		0	0*
r		w	rw

1: RESET Reset complete core when 1. Self-clearing.
0: ENABLE Enable CAN-FD codec when 1. Disable CAN-FD codec when 0.

* The reset value of this field may be overwritten with the input configuration signals (cfg record) if *canopen* mode is used.

RESET takes effect on the whole IP, including the internal codec. It is self-clearing, so it is read back as 0b.

The internal codec shall be disabled by setting ENABLE to 0 before modifying any CAN-related settings, such as the configuration of the bit times. This ensures that the integration with the CAN bus is correctly performed.

When ENABLE is cleared to 0b, the CAN interface is in sleep mode, only outputting recessive bits.

Once the CAN-FD codec is enabled again, it needs to detect 11 consecutive recessive bits on the CAN bus prior to starting the normal operation, i.e. transmit or receive any frame.

26.11.4 Capability Register

*Table 279.*0x00C - CAP - Capability Register

31 11	10 8	7	6 4	3 1	0
Res	TxBufSize	Res	RxBufSize	Res	SingIRQ
1100...0	-	0	-	000	-
r	r	r	r	r	r

10-8: TxBufSizeSize of the internal TX buffer, expressed in number of frames - 1.
6-4: RxBufSizeSize of the internal RX buffer, expressed in number of frames - 1.
0: SingIRQThe controller has a single IRQ output if SingIRQ is set to 1b, otherwise there are three different interrupts.

The capability register contains general information about the instantiation of the IP.

If SingIRQ is set to 1b, the controller combines all the interrupts in a single interrupt output. If set to 0b, GRCANFD features up to 3 interrupt outputs: a common line, together with dedicated lines for TxSYNC and RxSYNC interrupts, respectively.

26.11.5 SYNC Mask Filter Register

*Table 280.*0x018 - SYNCMASK - SYNC Mask Filter Register

31 30 29	28 0
Reserved	MASK
0x0	0x1FFFFFFF
r	rw

28-0: MASK Mask for SYNC filter

All bits of the MASK field are set to 1 at reset.

Note that Base ID corresponds to the bits 28 to 18 and Extended ID corresponds to bits 17 to 0.

26.11.6 SYNC Code Filter Register

*Table 281.*0x01C - SYNCCODE - SYNC Code Filter Register

31 30 29	28 0
Reserved	CODE
0x0	0x00000000
r	rw

28-0: CODE Code for SYNC filter

The CAN Base ID corresponds to the bits 28 to 18 and Extended ID corresponds to bits 17 to 0.

The SYNC filter is applied to the transmitted frames as soon as the codec transmits the frame. For the RX channel, the filter is applied as soon as the codec receives the frame, regardless of whether the frame is to be stored into the circular buffer afterwards, so it does not depend on the configuration of the Acceptance filter. Specific interrupts are available for both SYNC filters.

An ID matches the RxSYNC filter when:

((Received-ID) XOR (SYNC CODE)) AND (SYNC MASK) = 0

An ID matches the TxSYNC filter when:

((Transmitted-ID) XOR (SYNC CODE)) AND (SYNC MASK) = 0

26.11.7 Nominal Bit-Rate Configuration Register

Table 282. 0x040 - NOMBR - Nominal Bit-Rate Configuration Register

31 24	23 16
Reserved	SCALER
0x00	0x00*
r	rw

15 10	9 5	4 0
PS1	PS2	SJW
0x00*	0x00*	0x00*
rw	rw	rw

23-16: SCALER Prescaler setting for nominal bit rate, 8-bit: system clock / (SCALER +1)
15-10: PS1 Phase Segment 1 for nominal bit rate, 6-bit
9-4: PS2 Phase Segment 2 for nominal bit rate, 5-bit
3-0: SJW Synchronization Jump Width, 5-bit

* The reset value of these fields may be overwritten with the input configuration signals (cfg record) .

The prescaler sets the number of clock cycles per nominal time quantum (plus an offset of 1). PS1, PS2 and SJW define the number of nominal quantum within the Phase Segment 1, Phase Segment 2 and Synchronization Jump Width, respectively.

Certain constraints apply to the previous parameters. Since GRCANFD is an FD enabled implementation with separate prescalers for the nominal and the data bit rate, the valid ranges are as follows:

- Prescaler: 0 - 255
- PS1: 2 - 63

- PS2: 2 - 16
- SJW: 1 - 16

Additional considerations must be taken when defining the parameters:

- PS2 >= SJW
- SJW <= min (PS1, PS2)

For more information regarding the parameters defining the nominal bit time, please refer to the ISO standard 11898-1:2015 (2nd edition).

Therefore, the Nominal time quantum can be obtained as follows:

(system clock period) * (SCALER+1)

whereas the resulting Nominal bit rate is:

(system clock frequency) / ((SCALER+1) * (1+ PS1 + PS2))

26.11.8 Data Bit-Rate Configuration Register

*Table 283.*0x044 - DATABR - Data Bit-Rate Configuration Register

31	24	23	16
Reserved		SCALER	
0x00		0x00	
r		rw	

15	14	13	10	9	8	5	4	3	0
Reserved		PS1		Res.	PS2		Res.	SJW	
00		0x0		0	0x0		0	0x0	
r		rw		r	rw		r	rw	

23-16: SCALER Prescaler setting for data bit rate, 8-bit: system clock / (SCALER +1)
13-10: PS1 Phase Segment 1 for data bit rate, 4-bit
8-5: PS2 Phase Segment 2 for data bit rate, 4-bit
3-0: SJW Synchronization Jump Width, 4-bit

The prescaler sets the number of clock cycles per data time quantum (plus an offset of 1). PS1, PS2 and SJW define the number of data quantum within the Phase Segment 1, Phase Segment 2 and Synchronization Jump Width, respectively.

Certain constraints apply to the previous parameters. Since GRCANFD is an FD enabled implementation with separate prescalers for the nominal and the data bit rate, the valid ranges are as follows:

- Prescaler: 0 - 255
- PS1: 1 - 15
- PS2: 2 - 8
- SJW: 1 - 8

Additional considerations must be taken when defining the parameters:

- SJW <= min (PS1, PS2)
- Data bit-rate >= Nominal bit-rate

For more information regarding the parameters defining the data bit time, please refer to the ISO standard 11898-1:2015 (2nd edition).

Therefore, the Data time quantum can be obtained as follows:

(system clock period) * (SCALER+1)

whereas the resulting Data bit rate is:

(system clock frequency) / ((SCALER+1) * (1+ PS1 + PS2))

26.11.9 Transmitter Delay Compensation Register

*Table 284.*0x048 - DELCOMP - Transmitter Delay Compensation Register

31 6	5 0
Reserved	TxCompVal
0x0000000	0x00
r	rw

5-0: TxCompVal Number of time quantum for the transmitter delay compensation.

This register configures the delay in terms of number of data quantum to be compensated during the data phase of an FD frame. A maximum of 2 data bit times may be compensated.

If set to 0, the transmitter delay compensation is internally disabled. Otherwise, the register defines the delay between the synchronization segment of a bit and its corresponding secondary sample point, when the signal may be safely read-back.

26.11.10CANOpen Control Register

*Table 285.*0x080 - COCTRL- CANOpen Control Register

31 11	10 4	3	2	1	0
Reserved	ID	Res	RC	SS	EN
0x000000	1111111*	0	0	0	0*
r	rw	r	rw	rw	rw

10-4: ID CANOpen Node ID. Used to determine if the input commands shall be processed by GRCANFD.
2: RC Redundant Control. If set, the controller will switch to the alternate CAN bus after a Heartbeat Timeout error.
1: SS Single-shot mode. If set, the controllerwill not retry to transmit a reply if it fails due to errors or arbitration.
0: EN Enable CANOpen mode when set to 1b.

* The reset value of these fields may be overwritten with the input configuration signals (cfg record) (TBD).

26.11.11CANOpen Heartbeat Timeout Register

*Table 286.*0x084 - COHBTO- CANOpen Heartbeat Timeout Register

31 0
TO
0x00000000
rw

31-0: TO Heartbeat Timeout expressed as number of CAN bit times. Once the timeout is reached without receiving a Heartbeat command, GRCANFD will trigger an interrupt.

26.11.12CANOpen Heartbeat Count Register

*Table 287.*0x088 - COHBCT- CANOpen Heartbeat Count Register

31	0
CT	
0x00000000	
r	

31-0: CT Heartbeat internal counter expressed as number of CAN bit times. Restarted every time a Heartbeat command is received. Read-only.

26.11.13CANOpen Status Register

*Table 288.*0x08C - COSTS- CANOpen Status Register

31	4	3	0
Reserved		STS	
0x0000000		0x0	
r		r	

3-0: STS Status of the CANOpen FSM, encoded as follows:
0x0: IDLE. Listening to the CAN bus while waiting for a new RX frame.
0x1: RX_DATABYTES. A frame passing the preliminary format check is being received.
0x2: DECODE. The reception has ended and the Controller checks if the frame is a CANOpen object.
0x3: INIT_WRITE. Initializing an AMBA write access after receiving a CANOpen write command.
0x4: END_WRITE. Waiting for the end of the write access.
0x5: INIT_READ. Initializing an AMBA read access after receiving a CANOpen read command.
0x6: END_READ. Waiting for the end of the read access.
0x7: INIT_REPLY. Initializing the TPDO as a reply to a CANOpen read command.
0x8: TX_DATABYTES. Transmitting the TPDO.
0x9: REPLY_NOK. The TPDO failed during transmission due to errors or loss of arbitration.

26.11.14 Transmit Channel Control Register

*Table 289.*0x200 - TXCTRL - Transmit Channel Control Register

31	4	3	2	1	0
Reserved		Dis Ack	Sin-gle	Ong oing	Ena ble
0x0000000		0	0	0	0
r		r	rw	r	rw

3: DisAck Disable request acknowledged
2: SINGLE Single shot mode
1: Ongoing Transmission ongoing
0: ENABLE Enable channel

The TX channel is enabled by setting ENABLE to 1b.

The Ongoing bit indicates whether the codec is transmitting a frame. If the TX channel is disabled while Ongoing is set to 1, the TX channel is disabled, but any ongoing transmission will continue until it finishes (either successfully or with errors). In this case, the DisACK bit is set to 1b to indicate the user that the TX channel will be completely disabled as soon as the ongoing transmission ends.

Changing the configuration of the TX channel (pointers, parameters of the buffer) is not safe when ongoing is set to 1b. The user shall wait until the TX channel is effectively disabled by monitoring both DisACK and ENABLE. When both are set to ‘0’, the codec is no longer transmitting a frame, so the TX channel can be safely reconfigured.

If the single shot mode is enabled, the TX channel will be automatically disabled when a transmission does not complete successfully. This may be due to loss of arbitration, transmission errors, ACK not being sent by the receivers, etc. In this case, the content of the SRAM is removed to avoid blocking any future transmission and GRCANFD does not update the TX read pointer.

The TX channel will also be automatically disabled if the ABORT bit is set to 1b in the Configuration Register, and an AMBA error occurs while fetching a descriptor from the circular buffer. If the codec is transmitting a frame in parallel, the transmission is not interrupted, since that frame did not cause the error accessing the bus (a transmission only starts when all the descriptors describing the frame are fetched).

26.11.15 Transmit Channel Address Register

*Table 290.*0x204 - TXADDR - Transmit Channel Address Register

31 10	9 0
ADDR	Reserved
0	0x000
rw	r

31-10: ADDR Base address for TX circular buffer

26.11.16 Transmit Channel Size Register

*Table 291.*0x208 - TXSIZE - Transmit Channel Size Register

31 21	20 6	5 0
Reserved	SIZE	Reserved
0x000	0x0000	0x00
r	rw	r

20-6: SIZE The size of the TX circular buffer is SIZE*4 descriptors

Valid SIZE values are between 0 and 16384.

Each descriptor occupies four 32-bit words. A frame may consist of 1 to 5 descriptors depending on its format and data length.

Note that the resulting behavior of invalid SIZE values is undefined.

Note that only (SIZE*4)-1 descriptors can be stored simultaneously in the buffer. This is to simplify wrap-around condition checking.

26.11.17 Transmit Channel Write Register

*Table 292.*0x20C - TXWR - Transmit Channel Write Register

31 20	19 4	3 0
Reserved	WRITE	Reserved
0x000	0x0000	0x0
r	rw	r

19-4: WRITE Pointer to last written descriptor+1

The WRITE field is written to in order to initiate a transfer, indicating the position +1 of the last descriptor to transmit.

Note that it is not possible to fill the buffer. There is always one descriptor position in buffer unused. Software is responsible for not over-writing the buffer on wrap around (i.e. setting WRITE=READ).

The field is implemented as relative to the buffer base address (scaled with the SIZE field).

GRCANFD reads this register to know if there are more descriptors to fetch. While operating, it will never modify the write pointer, unless the core is being reset.

26.11.18 Transmit Channel Read Register

*Table 293.*0x210 - TXRD - Transmit Channel Read Register

31 20	19 4	3 0
Reserved	READ	Reserved
0x000	0x0000	0x0
r	rw	r

19-4: READ Pointer to last read descriptor+1

The READ field is written to automatically by the core when a transfer has been completed successfully, indicating the position +1 of the last descriptor transmitted. If a frame consists of more than 1 descriptor, GRCANFD will not increment the pointer one by one, but update it with the final position directly.

Note that the READ field can be use to read out the progress of a transfer of a set of frames.

Note that the READ field can be written to in order to set up the starting point of a transfer. This should only be done while the transmit channel is not enabled and the codec is not transmitting any frame.

Note that the READ field may be incremented even if the transmit channel has been disabled, in case the codec was already transmitting a frame when the ENABLE was set to 0b. As explained previously, this is indicated by both the Ongoing and the DisACK bits of the TX control register.

When the Transmit Channel Read Pointer catches up with the Transmit Channel Write Register, an interrupt is generated (TxEmpty). Note that this implies that all descriptors stored in the buffer have been transmitted.

The field is implemented as relative to the buffer base address (scaled with the SIZE field).

26.11.19 Transmit Channel Interrupt Register

*Table 294.*0x214 - TXIRQ - Transmit Channel Interrupt Register

31 20	19 4	3 0
Reserved	IRQ	Reserved
0x000	0x0000	0x0
r	rw	r

19-4: IRQ Interrupt is generated when the value in the Tx Read register becomes equal to IRQ, as a consequence of the transmission of a frame

This register configures the interrupt TxIRQ, which indicates that a programmed number of descriptors have been transmitted.

The field is implemented as relative to the buffer base address (scaled with the SIZE field).

Since CAN-FD frames may consist of up to 5 descriptors, the TX read pointer may be incremented internally several times per frame. The content of this register is therefore compared with the TX read pointer after each increment. This means that the interrupt will be asserted if the position of any of the

descriptors conforming the frame matches the programmed position. Note that the interrupt is only generated once the transmission of the frame is successfully completed.

26.11.20 Receive Channel Control Register

*Table 295.*0x300 - RXCTRL - Receive Channel Control Register

31 4	3	2	1	0
Reserved	OF	Dis Ack	Ong oing	Ena ble
0x0000000	0	0	0	0
r	rw	r	r	rw

3: OF Overload Frame enable
2: DisAck Disable request acknowledged
1: ONGOING Reception ongoing (read-only)
0: ENABLE Enable channel

The RX channel is enabled by setting ENABLE to 1b.

The Ongoing bit indicates that a reception is taking place. There are two scenarios for this: the codec may be receiving the frame, or the frame has already been received, matches the acceptance filter and it is yet to be written to the circular buffer.

If the user disables the RX channel by setting ENABLE to 0b, any ongoing write access over the AMBA bus will finish. If it was the last descriptor of a frame, the reception is complete and the pointers, status registers and interrupts are updated accordingly. The DisACK bit is then set to 1b to acknowledge the disable request, and is cleared as soon as the RX channel is effectively disabled.

The RX channel may not be configured while Ongoing is 1b, as this may disrupt the status of the CAN bus. The user shall monitor both Ongoing and DisACK bits: when both are 0b and the channel has been disabled, it is safe to modify the configuration of the RX channel.

The RX channel is automatically disabled if the ABORT bit is set to 1b in the Configuration Register and an AMBA error is detected when accessing the bus.

The content of the RX SRAM is emptied whenever the RX channel is effectively disabled.

26.11.21 Receive Channel Address Register

*Table 296.*0x304 - RXADDR - Receive Channel Address Register

31 10	9 0
ADDR	Reserved
0x000000	0x000
rw	r

31-10: ADDR Base address for RX circular buffer

26.11.22 Receive Channel Size Register

*Table 297.*0x308 - RXSIZE - Receive Channel Size Register

31 21	20 6	5 0
Reserved	SIZE	Reserved
0x000	0x0000	0x00
r	rw	r

20-6: SIZE The size of the RX circular buffer is SIZE*4 descriptors

Valid SIZE values are between 0 and 16384.

Note that each descriptor occupies four 32-bit words. A frame may consist of 1 to 5 descriptors depending on its format and data length.

Note that the resulting behavior of invalid SIZE values is undefined.

Note that only (SIZE*4)-1 descriptors can be stored simultaneously in the buffer. This is to simplify wrap-around condition checking.

26.11.23 Receive Channel Write Register

*Table 298.*0x30C - RXWR - Receive Channel Write Register

31 20	19 4	3 0
Reserved	WRITE	Reserved
0x000	0x0000	0x0
r	rw	r

19-4: WRITE Pointer to last written descriptor +1

The field is implemented as relative to the buffer base address (scaled with the SIZE field).

The WRITE field is written to automatically when a transfer has been completed successfully, indicating the position +1 of the last descriptor received. For frames consisting of more than 1 descriptor, GRCANFD will not increment the Write pointer one by one, but update it with the final position directly.

The Write pointer may be updated even after disabling the RX channel. This may be due to the fact that there was a frame being received or already in the local SRAM, and the storage takes place after the disable request.

Note that the WRITE field can be use to read out the progress of a transfer of a set of frames.

Note that the WRITE field can be written to in order to set up the starting point of a transfer. This should only be done while the receive channel is not enabled.

If the RX write pointer catches up with the read pointer, there is no space in the buffer for any additional descriptor. An interrupt is asserted informing that the buffer is full. GRCANFD will detect this and transmit up to 2 consecutive Overload Frames to delay the reception of the next frame. If a new frame arrives under this circumstance, the controller misses it and the Overrun bit in the Status Register is set, together with the Overrun interrupt.

26.11.24 Receive Channel Read Register

*Table 299.*0x310 - RXRD - Receive Channel Read Register

31 20	19 4	3 0
Reserved	READ	Reserved
0x000	0x0000	0x0
r	rw	r

19-4: READ Pointer to last read descriptor +1

The field is implemented as relative to the buffer base address (scaled with the SIZE field).

The READ field is written to in order to release the receive buffer, indicating the position +1 of the last descriptor that has been read out.

Note that it is not possible to fill the buffer. There is always one descriptor position in buffer unused. Software is responsible for not over-reading the buffer on wrap around (i.e. setting WRITE=READ).

GRCANFD will never modify the RX read pointer, unless the controller is being reset.

26.11.25 Receive Channel Interrupt Register

*Table 300.*0x314 - RXIRQ - Receive Channel Interrupt Register

31 20	19 4	3 0
Reserved	IRQ	Reserved
0x000	0x0000	0x0
r	rw	r

19-4: IRQ Interrupt is generated when the value in the Rx Write register becomes equal to IRQ, as a consequence of reception of a message

This register configures the interrupt which indicates that a programmed number of descriptors have been received.

The field is implemented as relative to the buffer base address (scaled with the SIZE field).

Since CAN-FD frames may consist of up to 5 descriptors, the RX write pointer may be incremented internally several times per frame. The content of this register is therefore compared with the RX write pointer after each increment. This means that the interrupt will be asserted if the position of any of the descriptors conforming the frame matches the programmed position. Note that the interrupt is only generated once the storage of the frame is successfully completed.

26.11.26 Receive Channel Acceptance Mask Register

*Table 301.*0x318 - RXMASK - Receive Channel Acceptance Mask Register

31 30 29	28 0
Reserved	AM
000	0x1FFFFFFF
r	rw

28-0: AM Acceptance Mask: bits set to 1b are taken into account in the comparison between the received frame ID and the CanRxCODE.AC field

All bits are set to 1 at reset.

Note that Base ID corresponds to the bits 28 to 18 and Extended ID corresponds to the bits 17 to 0.

26.11.27 Receive Channel Acceptance Code Register

*Table 302.*0x31C - RXCODE - Receive Channel Acceptance Code Register

31 30 29	28 0
Reserved	AC
000	0x00000000
r	rw

28-0: AC Acceptance Code, used in comparison with the received frame ID

Note that Base ID corresponds to the bits 28 to 18 and Extended ID corresponds to the bits 17 to 0.

When a frame is received by the internal CAN-FD codec, GRCANFD applies the Acceptance filter to decide whether it should be stored into the circular buffer. A frame matches the filter if the following condition is verified:

((Received-ID) XOR (ACCPT CODE)) AND (ACCPT MASK) = 0

26.11.28Interrupt registers

The interrupt registers give complete freedom to the software, by providing means to mask interrupts, clear interrupts, force interrupts and read interrupt status.

When an interrupt occurs the corresponding bit in the Pending Interrupt Register is set. The normal sequence to initialize and handle a module interrupt is:

- Set up the software interrupt-handler to accept an interrupt from the module.
- Read the Pending Interrupt Register to clear any spurious interrupts.
- Initialize the Interrupt Mask Register, unmasking each bit that should generate the module interrupt.
- When an interrupt occurs, read the Pending Interrupt Status Register in the software interrupt-handler to determine the causes of the interrupt.
- Handle the interrupt, taking into account all causes of the interrupt.
- Clear the handled interrupt using Pending Interrupt Clear Register.

Masking interrupts: After reset, all interrupt bits are masked, since the Interrupt Mask Register is zero. To enable generation of a module interrupt for an interrupt bit, set the corresponding bit in the Interrupt Mask Register.

Clearing interrupts: All bits of the Pending Interrupt Register are cleared when it is read or when the Pending Interrupt Masked Register is read. Reading the Pending Interrupt Masked Register yields the contents of the Pending Interrupt Register masked with the contents of the Interrupt Mask Register. Selected bits can be cleared by writing ones to the bits that shall be cleared to the Pending Interrupt Clear Register.

Forcing interrupts: When the Pending Interrupt Register is written, the resulting value is the original contents of the register logically OR-ed with the write data. This means that writing the register can force (set) an interrupt bit, but never clear it.

Reading interrupt status: Reading the Pending Interrupt Status Register yields the same data as a read of the Pending Interrupt Register, but without clearing the contents.

Reading interrupt status of unmasked bits: Reading the Pending Interrupt Masked Status Register yields the contents of the Pending Interrupt Register masked with the contents of the Interrupt Mask Register, but without clearing the contents.

The interrupt registers comprise the following:

- Pending Interrupt Masked Status Register[CanPIMSR]R

- Pending Interrupt Masked Register[CanPIMR]R
- Pending Interrupt Status Register[CanPISR]R
- Pending Interrupt Register[CanPIR]R/W
- Interrupt Mask Register[CanIMR]R/W
- Pending Interrupt Clear Register[CanPICR]W

*Table 303.*Interrupt registers

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
											CO Hb Err	CO Sync	CO Wr cmd	CO Rd cmd	Tx Loss
											0	0	0	0	0
											*	*	*	*	*

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Rx Miss	Tx Err Cntr	Rx Err Cntr	Tx Sync	Rx Sync	Tx	Rx	Tx Empty	Rx Full	Tx IRQ	Rx IRQ	BM Rd Err	BM Wr Err	OR	Off	Pass
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

20: COHbErr CANOpen Heartbeat Timeout Error
19: COSync CANOpen SYNC command received
18: COWrCmd CANOpen write command processed
17: CORdCmd CANOpen read command processed
16: TxLoss Unsuccessful transmission (due to loss of arbitration or errors in the frame)
15: RxMiss Message filtered away during reception
14: TxErrCntr Transmission error counter incremented
13: RxErrCntr Reception error counter incremented
12: TxSync Synchronization message transmitted by the codec
11: RxSync Synchronization message received by the codec
10: Tx Successful transmission of message
9: Rx Successful reception of message
8: TxEmpty Successful transmission of all frames in TX circular buffer
7: RxFull Successful reception of all frames possible to store in RX circular buffer
6: TxIRQ The Tx Read Pointer is equal to the value stored in Tx IRQ register
5: RxIRQ The Rx Write Pointer is equal to the value stored in Rx IRQ register
4: BmRdErr Error during AMBA read access
3: BmWrErr Error during AMBA write access
2: OR Over-run during reception
1: OFF Bus-off condition
0: PASS Error-passive condition

All bits in all interrupt registers are reset to 0b after reset.

Note that the BmRdErr interrupt is generated in such way that the corresponding read and write pointers are valid for failure analysis in standard mode. The interrupt generation is independent of the CanCONF.ABORT field setting.

Note that the BmWrErr interrupt is generated in such way that the corresponding read and write pointers are valid for failure analysis in standard mode. The interrupt generation is independent of the CanCONF.ABORT field setting.

GRCANFD is designed following the same approach as GRCAN, but adapting the register interface to the new frame formats and data lengths. This section summarizes the main differences introduced by GRCANFD from the user's point of view:

- While frames in GRCAN were always represented by a single descriptor, FD frames in GRCANFD may require from 1 to 5 descriptors in the circular buffers due to the larger data payload. Whereas the first descriptor of a frame is still compatible, the descriptors 2 to 5 (if applicable) only contain data bytes and do not replicate the bits describing the format of the frame.
- The first descriptor of a frame includes the FDF and BRS bits. These are both set to 0b for classical CAN frames, so it is fully compatible with the memory representation of CAN frames in GRCAN, as these bits were unused.
- The Configuration Register (APB offset 0x000) does not include the CAN timing parameters anymore. Two new registers are created for this purpose: the Nominal Bit-Rate Configuration Register (0x040) and Data Bit-Rate Configuration Register (0x044).
- BPR (Baud Rate) and SAM (Triple Sampling) not supported by GRCANFD. In GRCAN these features may be enabled in the Configuration Register (0x000).
- New control bits to enable the generation of Overload Frames (Receive Channel Control Register, 0x300) and the internal/external loop-back modes (Configuration Register 0x000).
- New Capability Register (0x00C) providing information about the configuration.
- New register for configuring the Transmitter Delay Compensation (0x048). This only applies to the data phase of FD frames (i.e. when switching the bit-rate).
- Since multiple descriptors may be necessary for describing a frame, the interrupts TxIRQ and RxIRQ are asserted when **any** of the descriptors of a transmitted/received frame matches the position programmed via the registers.
- Different device identifier: 0x03D for GRCAN; 0x0B5 for GRCANFD.
- New CANOpen mode. This mode uses a different set of registers. By default the controller operates in standard mode, which is the same general behavior as in GRCAN.

27 Clock gating unit (primary and secondary)

The GR716B microcontroller has 2 separate clock gating units. Each clock gating unit controls its own clock domains and has a unique AMBA address described in chapter 2.10.

27.1 Overview

The clock gating units provide a means to save power by disabling the clock to unused functional blocks. The primary clock gating unit provides a mechanism to automatically disabling the clock to the LEON processor when it is in power-down mode, and also to disable the clock for the shared floating-point unit. The primary clock gating unit also provides a mechanism to reset, enable clock and disable clock for following cores:

- FTMCTRL
- SPI4S
- GRSPWTDP
- GRMEMPROT
- L3STAT
- UART
- AHBUART
- GRPWM
- I2CMST
- I2CSLV
- SPICTRL
- SPIMCTRL
- SPI2AHB
- GRPWRX
- GRPWTX
- SPI2AHB
- I2C2AHB
- NVRAM
- APWMCTRL
- ACOMP
- GPIO

The secondary clock gating unit provides a mechanism to reset, enable clock and disable clock for the following cores:

- GRDMAC2
- GR1553B
- GRCANFD
- GRSPWROUTER
- ADC
- DAC
- RTA
- APWM

- FIR
- APWMDAC
- GPTIMER
- GPIO
- GRSCRUB

Both units provide a register interface via their APB slave bus interface.

27.2 Operation

The operation of the clock gating unit is controlled through four registers: the unlock, clock enable, core reset and CPU/FPU override registers. The clock enable register defines if a clock is enabled or disabled. A '1' in a bit location will enable the corresponding clock, while a '0' will disable the clock. The core reset register allows to generate a reset signal for each generated clock. A reset will be generated as long as the corresponding bit is set to '1'. The bits in clock enable and core reset registers can only be written when the corresponding bit in the unlock register is 1. If a bit in the unlock register is 0, the corresponding bits in the clock enable and core reset registers cannot be written.

To gate the clock for a core, the following procedure should be applied:

1. Disable the core through software to make sure it does not initialize any AHB accesses
2. Write a 1 to the corresponding bit in the unlock register
3. Write a 0 to the corresponding bit in the clock enable register
4. Write a 0 to the corresponding bit in the unlock register

To enable the clock for a core, the following procedure should be applied

1. Write a 1 to the corresponding bit in the unlock register
2. Write a 1 to the corresponding bit in the core reset register
3. Write a 1 to the corresponding bit in the clock enable register
4. Write a 0 to the corresponding bit in the clock enable register
5. Write a 0 to the corresponding bit in the core reset register
6. Write a 1 to the corresponding bit in the clock enable register
7. Write a 0 to the corresponding bit in the unlock register

The clock gating unit also provides gating for the processor core and floating-point unit. The processor core will be automatically gated off when it enters power-down mode.

The FPU will be gated off when the LEON3FT processor core connected to the FPU have floating-point disabled or when the LEON3FT processor core is in power-down mode.

Processor/FPU clock gating can be disabled by writing '1' to bit 0 of the CPU/FPU override register.

27.3 Registers

The core's registers are mapped into APB address space.

Table 304. Clock gate unit registers

AMBA address	Register	Acronym
0x80006000	Unlock register 0 (primary clock gating unit)	UNLOCK0
0x80006004	Clock enable register 0 (primary clock gating unit)	CLKEN0
0x80006008	Core reset register 0 (primary clock gating unit)	RESET0
0x8000600C	CPU/FPU override register 0 (primary clock gating unit)	OVERRIDE0
0x80006010 - 0x800060FF	Reserved	N/A
0x80007000	Unlock register 1 (secondary clock gating unit)	UNLOCK1
0x80007004	Clock enable register 1 (secondary clock gating unit)	CLKEN1
0x80007008	Core reset register 1 (secondary clock gating unit)	RESET1
0x8000700C - 0x800070FF	Reserved	N/A

27.3.1 Unlock register 0

*Table 305.*0x80006000 - UNLOCK0 - Unlock register 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
G0	NV	SP	TD	AS	MP	AU	L3	R	R	H5	U4	U3	U2	U1	U0	P1	P0	DA	IS1	IS0	IM1	IM0	S1	S0	M1	M0	MC	PX	PR	IA	SA
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
rw	rw	rw	rw	rw	rw	rw	rw	rw	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

31: 0 Unlock clock enable and reset registers (UNLOCK) - The bits in clock enable and core reset registers can only be written when the corresponding bit in this field is 1. See Table 306 for bit field description

27.3.2 Clock enable register 0

*Table 306.*0x80006004 - CLKEN0 - Clock enable register 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
G0	NV	SP	TD	R	MP	AU	L3	AP	R	U5	U4	U3	U2	U1	U0	R	CD	AC	IS1	IS0	IM1	IM0	S1	S0	M1	M0	MC	PX	PR	IA	SA
0	*	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	*	*	0	0	*	*	*	0	0	*	*
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

* Reset value depends on bootstrap signals.

** Bit field may be set to 1 by boot SW during startup of the device.

31: 0 Clock enable - A ‘1’ in a bit location will enable the corresponding clock, while a ‘0’ will disable the clock.

31 Clock enable GPIO0 (G0)
30 Clock enable NVRAM (NV)
29 Clock enable SPI4S (SP)
28 Clock enable GRSPWTDP (TD)
27 RESERVED
26 Clock enable MEMPROT (MP)
25 Clock enable AHBUART (AU)
24 Clock enable L3STAT (L3)
23 Clock enable for AMBA CTRL of APWM, APWMDAC, CLKDET, FIR, ACOMP and PROT (AP)
22 RESERVED
21 Clock enable APBUART5 (U5)
20 Clock enable APBUART4 (U4)
19 Clock enable APBUART3 (U3)
18 Clock enable APBUART2 (U2)
17 Clock enable APBUART1 (U1)
16 Clock enable APBUART0 (U0)
15 RESERVED
14 Clock enable CLKDET (CD)
13 Clock enable ACOMP (AC)
12 Clock enable I2CLSV1 (IS1)
11 Clock enable I2CLSV0 (IS0)
10 Clock enable I2CMST1 (IM1)
9 Clock enable I2CMST0 (IM0)
8 Clock enable SPICTRL1 (S1)
7 Clock enable SPICTRL0 (S0)
6 Clock enable SPIMCTRL1 (M1)
5 Clock enable SPIMCTRL0 (M0)
4 Clock enable FTMCTRL (MC)
3 Clock enable GRPWTX (PX)
2 Clock enable GRPWRX (PR)
1 Clock enable I2C2AHB (IA)
0 Clock enable SPI2AHB (SA)

27.3.3 Core reset register 0

Table 307. 0x80006008 - RESET0 - Reset register 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
G0	NV	SP	TD	AS	MP	AU	L3	ID	R	U5	U4	U3	U2	U1	U0	P1	P0	DA	IS1	IS0	IM1	IM0	S1	S0	M1	M0	MC	PX	PR	IA	SA
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
rw	rw	rw	rw	rw	rw	rw	rw	rw	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

31: 0 Reset (RESET) - A reset will be generated as long as the corresponding bit is set to ‘1’. See Table 306 for bit field description. The reset value for each bit is the inverse of the reset value of the corresponding bit in the CLKEN0 register.

27.3.4 CPU/FPU override register 0

Table 308. 0x8000600C - OVERRIDE0 - CPU/FPU override register 0

31 17	16	15 1	0
RESERVED	FOVERRIDE	RESERVED	OVERRIDE
0	0	0	0
r	rw	r	rw

31: 17 RESERVED

16 Override FPU clock gating (FOVERRIDE) - If bit n of this field is set to ’1’ then the clock for FPU n will be active regardless of the value of %PSR.EF.

15: 1 RESERVED

0 Override CPU clock gating (OVERRIDE) - If bit n of this field is set to ’1’ then the clock for the processor and FPU will always be active.

27.3.5 Unlock register 1

*Table 309.*0x80007000 - UNLOCK1 - Unlock register 1

31	30	29	28 26	25	24	23	22	21 20 19 18 17 16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
G1	T1	FR	RESERVED	D4	R1	R0	SC	RESERVED	A3	A2	A1	A0	D3	D2	D1	D0	SP	LV	C0	ML	GR	AP	R	M0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

31: 0 Unlock clock enable and reset registers (UNLOCK) - The bits in clock enable and core reset registers can only be written when the corresponding bit in this field is 1. See Table 310 for bit field description

27.3.6 Clock enable register 1

*Table 310.*0x80007004 - CLKEN1 - Clock enable register 1

31	30	29	28 26	25	24	23	22	21 20 19 18 17 16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
G1	T1	FR	RESERVED	D4	R1	R0	SC	RESERVED	A3	A2	A1	A0	D3	D2	D1	D0	SP	LV	C0	ML	GR	AP	R	M0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	*	0	*	0	0	0	0	0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

* The reset value of this bit depends on bootstrap signals.

31: 0 Clock enable - A '1' in a bit location will enable the corresponding clock, while a '0' will disable the clock.

31 Clock enable GPIO1 (G1)
30 Clock enable GPTIMER1 (T1)
29 Clock enable FIR (FR)
28: 26 RESERVED
25 Clock enable APWMDAC0, APWMDAC1, APWMDAC2 and APWMDAC3 (D4)
24 Clock enable RTA1 (R1)
23 Clock enable RTA0 (R0)
22 Clock enable GRSCRUB (SC)
21: 16 RESERVED
15 Clock enable ADC3 (A3)
14 Clock enable ADC2 (A2)
13 Clock enable ADC1 (A1)
12 Clock enable ADC0 (A0)
11 Clock enable DAC3 (D3)
10 Clock enable DAC2 (D2)
9 Clock enable DAC1 (D1)
8 Clock enable DAC0 (D0)
7 Clock enable GRSPWROUTER (SP)
6 Clock enable LVDSIO Control (LV)
5 Clock enable GRCANFD (C0)
4 Clock enable GR1553B (ML)
3 Clock enable GRETH (GR)
2 Clock enable APWM (AP)
1 RESERVED
0 Clock enable GRDMAC2 (M0)

27.3.7 Core reset register 1

Table 311. 0x80007008 - RESET1 - Reset register 1

31	30	29	28 26	25	24	23	22	21 20 19 18 17 16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
G1	T1	FR	RESERVED	D4	R1	R0	SC	RESERVED	A3	A2	A1	A0	D3	D2	D1	D0	SP	LV	C0	ML	GR	AP	R	M0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

31: 0 Reset (RESET) - A reset will be generated as long as the corresponding bit is set to '1'. See Table 310 for bit field description.

28 Digital Finite Impulse Response filter (FIR)

The GR716B microcontroller contains 8 digital Finite Impulse Response filters (FIR). The FIR filter inputs can be configured to be connected to ACOMP output (section 16), GPIO inputs, and APWM G*/G outputs (section 53). Each filter has 27 binary taps, and the input signal is binary with binary tap configuration.

The general 27 tap FIR filter with 1 bits per tap can implement a window function and some application areas are:

- Flexible programmable latch-up detectors
- Simple FIR filters
- Simple patterns recognition in communications, sensors, etc
- Continuous background-monitoring with interrupt trigger on wave patterns, sensors, etc.

Configuration and readout of the FIR filters is done through register interfaces on APB buses. The registers are documented in section 28.2. Figure 69 contains a bus diagram showing the FIR filters, their clock generator and supporting peripherals.

The clock gating unit **GRCLKGATE** described in section 27 is used to enable/disable FIR block. The unit **GRCLKGATE** can also be used to perform reset of the FIR block. Software must enable clock and release reset described in section 27 before FIR configuration and operation can start. Note that there is a separate clock for the register interface that must also be enabled in the **GRCLKGATE** before any FIR registers can be read or written.

The system can be configured to protect and restrict access to FIR registers in the **MEMPROT** and **APBPROT** units. See section 47 for more information.

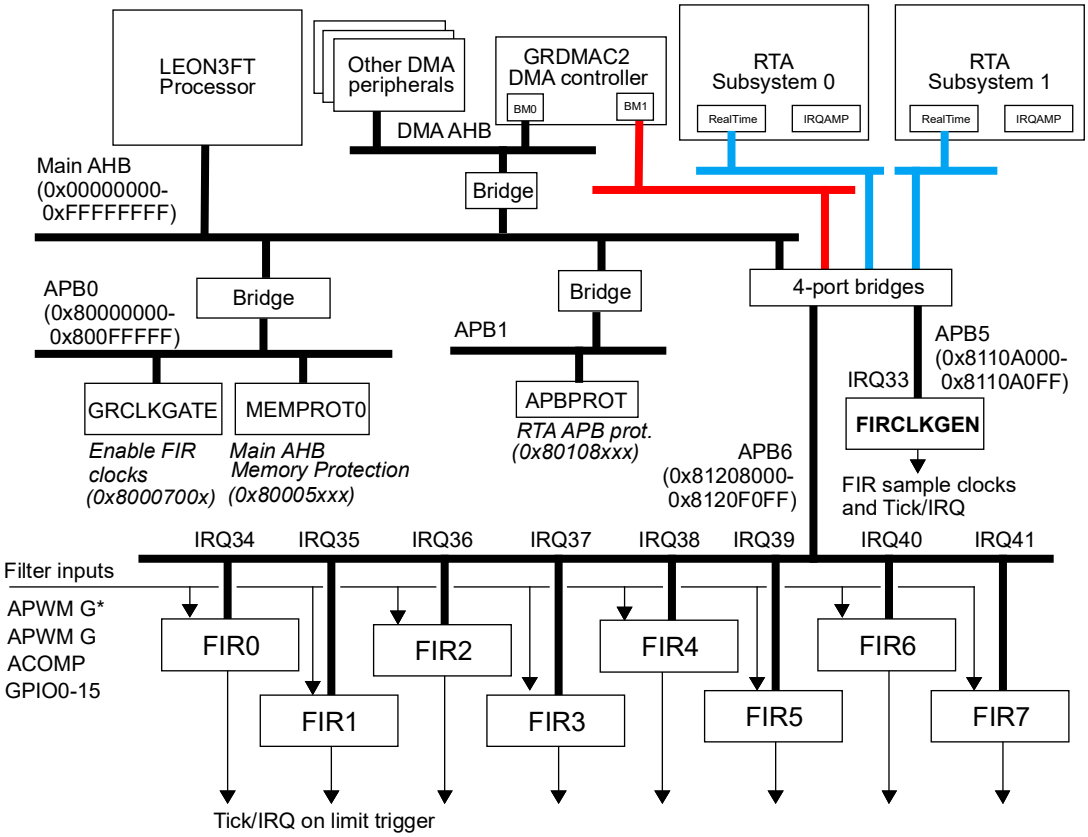


Figure 69. Partial bus diagram showing how the FIR and related peripherals are connected to the system bus.

28.1 FIR detailed description

This section makes frequent reference to FIR register acronyms (e.g. FIR.DATA). See section 28.2 for a detailed description of each register and Table 313 for a list of all registers. The GR716B has 8 FIR filters, each with their own identical set of registers.

The FIR filters have 27 binary taps with 1 bit per tap and their structure is shown in Figure 70. The delay shift register (FIR.DATA) is clocked by a local sample clock generated from the system clock by dividing by an integer. Both the sampled data (FIR.DATA) and the FIR filter output (FIR.SUM) are available for readout by RTA or main processor via the register interface. The filter output is

SUM[4:0] = DATA[0]*COEF[0] + DATA[1]*COEF[1] + ... + DATA[26]*COEF[26]

where DATA[0] is the most recently sampled input bit, and COEF[26:0] are the filter tap coefficients configured by the FIR.FCFG register. The concatenation of the FIR.SUM and FIR.DATA register contents can also be read atomically via the FIRG.STS registers.

In total there are eight FIR filters clocked by two sample clock generators as shown in Figure 71, which also shows some of the internal structure of the FIRCLKGEN block. Clock generator 0 clocks FIR 0-3 while clock generator 1 clocks FIR 4-7. The sample clock is generated by dividing the system clock by an integer, the resulting clock frequency is (system frequency)/(PERIOD+1), where PERIOD is the configuration value in the FIRG.PERIOD0 or FIRG.PERIOD1 register. Optionally, the sample clock can be synchronized to system timer ticks (TIMER32 A/B, see section 53.9). If synchronization is enabled (FIRG.SYNCx.EN=1), then the clock divider will restart any time a tick occurs on the timer selected by FIRG.SYNCx.SEL. Rising edges of the sample clock can optionally be configured to generate ticks (TickLines0(13) for clock generator 0, TickLines0(14) for clock generator 1) or assert interrupt line 33. See section 28.2.1.

A FIR filter can be configured to generate a limit triggered tick and interrupt when the filter output (FIR.SUM) exceeds the FIR.FCFG.COMP configuration value. The current status of the limit trigger can be read from FIR.STAT. A tick/interrupt will be generated only when the FIR.IRQ interrupt flag changes from 0 to 1. This requires first unmasking the interrupt in FIR.IMASK. A tick will be generated during one system clock cycle when the FIR.IRQ.COMP bit changes from 0 to 1. The FIR *n* tick output is connected to TickLines0(4+*n*), see also section 52.5. When any bit in FIR.IRQ changes from 0 to 1, the FIR will also emit a system interrupt on interrupt line 34+*n*, see section 2.12.

A tick will be generated during one system clock cycle when the FIR.IRQ.COMP bit changes from 0 to 1. This tick can be used in the RTA task manager (section 52.5) and as ADC and DAC trigger sources (sections 12 and 15).

Each FIR filter has 32 selectable input sources from APWM G*/G state outputs (section 53.5), ACOMP comparator outputs (section 16) and GPIO inputs. The set of inputs varies by FIR filter, see Table 312 for a list of inputs available in each individual FIR filter. The StateG*/G signals are available for all FIR filters. The ACOMP_SYNCH outputs are split into two groups, with one set connected to FIR0-3 and the other connected to FIR4-7. The GPIO inputs are split into four groups connected to FIR0-1, FIR2-3, FIR4-5, and FIR6-7 respectively.

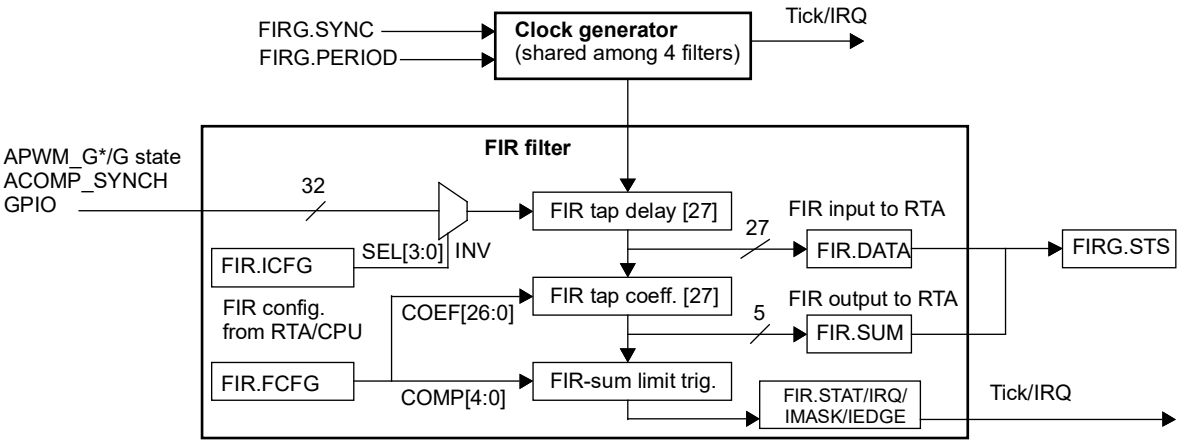


Figure 70. Filter structure of one FIR unit including register names.

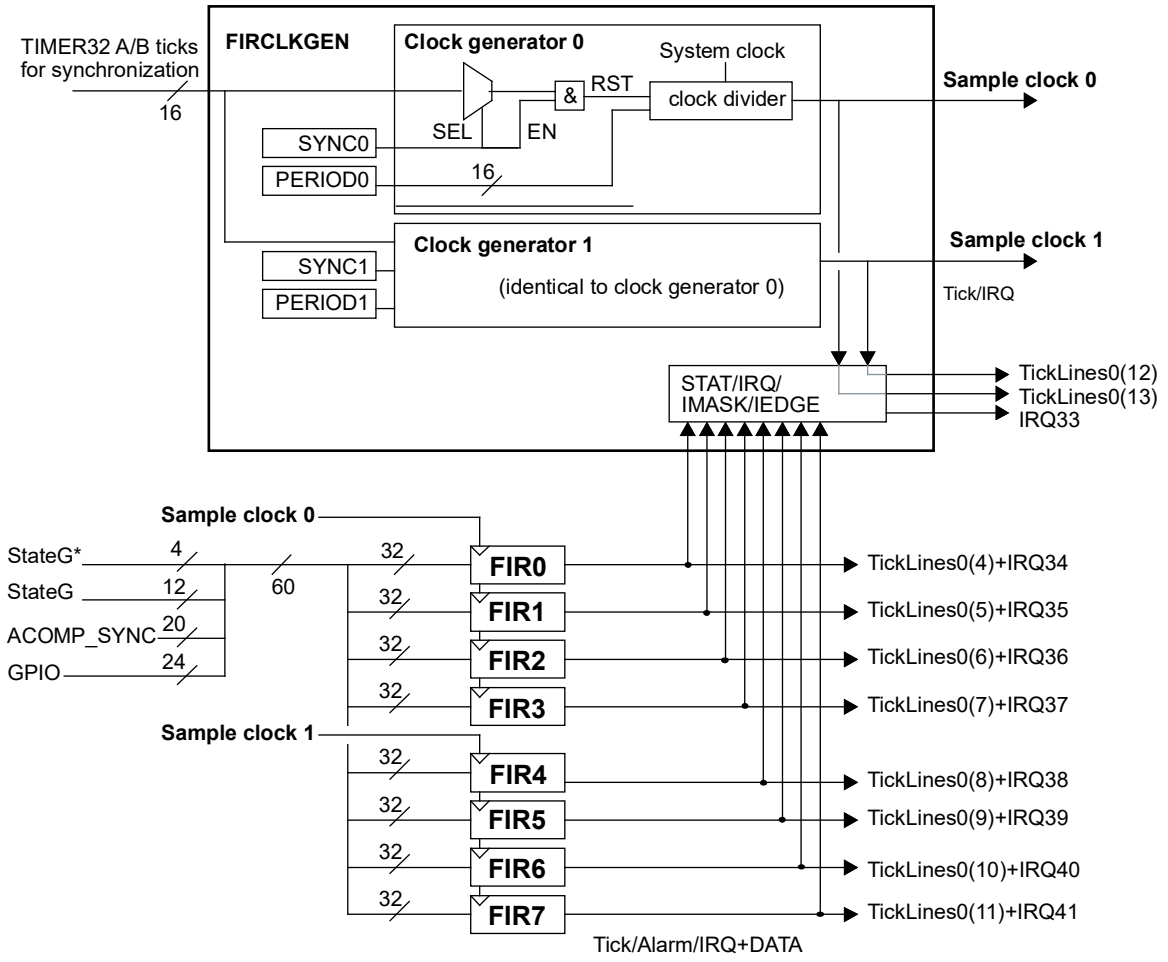


Figure 71. Block diagram of the FIR block with detail of clock generator.

Table 312. Selectable input sources for the eight FIR filters. The column "Input#" contains the value that the FIR*n*.ICFG.SEL field should have to select a particular input source. A filter input may also be inverted by setting FIR*n*.ICFG.INV=1.

Input#	FIR0	FIR1	FIR2	FIR3	FIR4	FIR5	FIR6	FIR7
0-3	[1] StateG 0-3							
4-15	[1] StateG* 0-11							
16-19	[2] ACOMP_SYNCH 0-3				[2] ACOMP_SYNCH 4-7			
20-21	[2] ACOMP_SYNCH 8-9				[2] ACOMP_SYNCH 10-11			
22-25	[2] ACOMP_SYNCH 12-15				[2] ACOMP_SYNCH 16-19			
26-31	[3] GPIO 0-5		[3] GPIO 6-11		[3] GPIO 12-17		[3] GPIO 18-23	

Note 1: StateG* and StateG are outputs of the APWM G units. See section 53.5.

Note 2: ACOMP_SYNCH is an output from the analog comparators. See section 16 and Figure 31.

Note 3: The FIR GPIO inputs are always taken from the Schmitt trigger receivers, regardless of the configuration of the I/O switch matrix or selection in the SYS.CFG.SCHMITT0-1 registers. Since the Schmitt receivers are always active, it is for example possible to use the FIR to monitor outputs of the GR716B itself.

28.2 Registers

The FIRCLKGEN and FIR cores are configured through memory registers connected to an APB bus.

*Table 313.*FIR clock generator (FIRCLKGEN/FIRG) and FIR filter (FIR0-7) registers

AMBA address	Register	Acronym
0x8110A000	FIR Clock Generator Status register	FIRG.STAT
0x8110A004	FIR Clock Generator Interrupt register	FIRG.IRQ
0x8110A008	FIR Clock Generator Mask register	FIRG.IMASK
0x8110A00C	FIR Clock Generator Interrupt Level trigger register	FIRG.IEDGE
0x8110A010	FIR0-3 clock generator clock period register	FIRG.PERIOD0
0x8110A014	FIR4-7 clock generator clock period register	FIRG.PERIOD1
0x8110A018	FIR0-3 clock generator synchronization configuration	FIRG.SYNC0
0x8110A01C	FIR4-7 clock generator synchronization configuration	FIRG.SYNC1
0x8110A020 - 0x8110A07C	RESERVED	N/A
0x8110A080	Combined FIR 0 status register	FIRG.STS0
0x8110A084	Combined FIR 1 status register	FIRG.STS1
0x8110A088	Combined FIR 2 status register	FIRG.STS2
0x8110A08C	Combined FIR 3 status register	FIRG.STS3
0x8110A090	Combined FIR 4 status register	FIRG.STS4
0x8110A094	Combined FIR 5 status register	FIRG.STS5
0x8110A098	Combined FIR 6 status register	FIRG.STS6
0x8110A09C	Combined FIR 7 status register	FIRG.STS7
0x8110A0A0 - 0x8110A0FF	RESERVED	N/A
0x81208000	FIR 0 interrupt status register	FIR0.STAT
0x81208004	FIR 0 interrupt flag register	FIR0.IRQ
0x81208008	FIR 0 interrupt mask register	FIR0.IMASK
0x8120800C	FIR 0 interrupt edge register	FIR0.IEDGE
0x81208010	FIR 0 input configuration register	FIR0.ICFG
0x81208014	FIR 0 filter configuration register	FIR0.FCFG
0x81208018 - 0x8120807C	RESERVED	N/A
0x81208080	FIR 0 filter input data register	FIR0.DATA
0x81208084	FIR 0 filter output sum register	FIR0.SUM
0x81208088 - 0x812080FF	RESERVED	N/A
0x81209000-0x81209FFF	FIR 1 Register area[1)]	FIR1.*
0x8120A000-0x8120AFFF	FIR 2 Register area[1)]	FIR2.*
0x8120B000-0x8120BFFF	FIR 3 Register area[1)]	FIR3.*
0x8120C000-0x8120AFFF	FIR 4 Register area[1)]	FIR4.*
0x8120D000-0x8120DFFF	FIR 5 Register area[1)]	FIR5.*
0x8120E000-0x8120EFFF	FIR 6 Register area[1)]	FIR6.*
0x8120F000-0x8120FFFF	FIR 7 Register area[1)]	FIR7.*

Note 1: FIR 1-7 have identical status and configuration register as “FIR 0”. When there is a need to distinguish registers within different FIR filters, they are referred to as FIR*n*.* where *n* ranges from 0 to 7. But the number is omitted when the identity of the FIR filter is not important.

28.2.1 FIRG Interrupt and tick registers

*Table 314.*0x8110A000 - FIRG.STAT - FIRG interrupt and tick status register

31	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED		P	F7	F6	F5	F4	F3	F2	F1	F0	T1	T0
0		0	0	0	0	0	0	0	0	0	0	0
r		rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*

31: 11 Reserved

10 FIRG configuration register parity error (P)

9 FIR filter 7 Tick (F7) - This bit is 1 for 1 system clock cycle any time a tick is emitted by FIR 7.

8 FIR filter 6 Tick (F6) - This bit is 1 for 1 system clock cycle any time a tick is emitted by FIR 6.

7 FIR filter 5 Tick (F5) - This bit is 1 for 1 system clock cycle any time a tick is emitted by FIR 5.

6 FIR filter 4 Tick (F4) - This bit is 1 for 1 system clock cycle any time a tick is emitted by FIR 4.

5 FIR filter 3 Tick (F3) - This bit is 1 for 1 system clock cycle any time a tick is emitted by FIR 3.

4 FIR filter 2 Tick (F2) - This bit is 1 for 1 system clock cycle any time a tick is emitted by FIR 2.

3 FIR filter 1 Tick (F1) - This bit is 1 for 1 system clock cycle any time a tick is emitted by FIR 1.

2 FIR filter 0 Tick (F0) - This bit is 1 for 1 system clock cycle any time a tick is emitted by FIR 0.

1 FIR Sample clock Tick 1 (T1) - This bit is 1 for 2 system clock cycles on any rising edge of the sampling clock generated by clock generator 1.

0 FIR Sample clock Tick 0 (T0) - This bit is 1 for 2 system clock cycles on any rising edge of the sampling clock generated by clock generator 0.

* For testing purposes (software injection of tick/interrupt), bits 10:0 in this register are writable. If this register is written they will be replaced by the written value for one system clock cycle after which they will return to the normal value.

*Table 315.*0x8110A004 - FIRG.IRQ - FIRG interrupt flag register

31	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED		P	F7	F6	F5	F4	F3	F2	F1	F0	T1	T0
0		0	0	0	0	0	0	0	0	0	0	0
r		wc	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc

Each bit in this register is the interrupt flag for the corresponding bit in FIRG.STAT (see Table 314). An interrupt flag will be set to 1 if the interrupt is unmasked in FIRG.IMASK and an edge occurs in FIRG.STAT. The type of edge (rising/falling) is configured in FIRG.IEDGE. All configuration is per bit. An interrupt flag can be cleared by writing 1.

*Table 316.*0x8110A008 - FIRG.IMASK - FIRG interrupt mask register

31	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED		P	F7	F6	F5	F4	F3	F2	F1	F0	T1	T0
0		0	0	0	0	0	0	0	0	0	0	0
r		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Each bit in this register is the interrupt mask for the corresponding bit in FIRG.STAT (see Table 314). Set to 1 to enable the interrupt, 0 to disable the interrupt.

*Table 317.*0x8110A00C - FIRGC.IEDGE - FIRG interrupt edge register

31	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED		P	F7	F6	F5	F4	F3	F2	F1	F0	T1	T0
0		1	1	1	1	1	1	1	1	1	1	1
r		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Each bit in this register is the interrupt edge selection for the corresponding bit in FIRG.STAT (see Table 314). Set to 1 to trigger on rising edges, 0 for falling edges.

28.2.2 FIRG Clock divider registers

Table 318. 0x8110A010 - FIRG.PERIOD0 - Clock period register for the FIR0-3 clock generator

31	0
PERIOD0	
0x00000000	
rw	

31: 0 FIR 0-3 Filter Period (PERIOD) - One FIR clock period will be PERIOD0+1 system clock cycles long. Applies to FIR 0-3.

Table 319. 0x8110A014 - FIRG.PERIOD1 - Clock period register for the FIR4-7 clock generator

31	0
PERIOD1	
0x00000000	
rw	

31: 0 FIR 4-7 Filter Period (PERIOD) - One FIR clock period will be PERIOD1+1 system clock cycles long. Applies to FIR 4-7.

28.2.3 FIRG synchronization configuration registers

Table 320. 0x8110A018 - FIRG.SYNC0 - Synchronization configuration register for the FIR0-3 clock generator

31 5	4	3 0
RESERVED	EN	SEL
0	0	0x0
r	rw	rw

31: 5 Reserved

4 Enable synchronization from tick input (EN) - If EN=1, then the clock for FIR0-3 will be restarted whenever the TIMER32 tick selected by SEL occurs.

3: 0 Synchronization input tick selection (SEL) - SEL=0-7 selects TIMER32 A 0-7 while SEL=8-15 selects TIMER32 B 0-7.

Table 321. 0x8110A01C - FIRG.SYNC1 - Synchronization configuration register for the FIR4-7 clock generator

31 5	4	3 0
RESERVED	EN	SEL
0	0	0x0
r	rw	rw

31: 5 Reserved

4 Enable synchronization from tick input (EN) - If EN=1, then the clock for FIR4-7 will be restarted whenever the TIMER32 tick selected by SEL occurs.

3: 0 Synchronization input tick selection (SEL) - SEL=0-7 selects TIMER32 A 0-7 while SEL=8-15 selects TIMER32 B 0-7.

28.2.4 FIR 0-7 interrupt registers

Table 322. 0x00 - FIR*n*.STAT - FIR *n* interrupt status register

31	2	1	0
RESERVED		PAR	COMP
0		0	0
r		rw*	rw*

31: 2 Reserved

1 FIR *n* configuration parity error (PAR) - This bit will be 1 if a configuration parity error is detected.

0 FIR *n* comparator output (COMP) - This bit is 1 if FIR*n*.SUM>FIR*n*.FCFG.COMP and 0 otherwise. This does not by itself cause a tick or interrupt, see FIR*n*.IRQ for details.

* For testing purposes (software injection of tick/interrupt), bits 1:0 in this register are writable. If this register is written they will be replaced by the written value for one system clock cycle after which they will return to the normal value.

Table 323. 0x04 - FIR*n*.IRQ - FIR *n* interrupt flag register

31	2	1	0
RESERVED		PAR	COMP
0		0	0
r		wc	wc

31: 2 Reserved

1 FIR *n* configuration parity error interrupt flag (PAR) - See below for general information.

0 FIR *n* comparator output interrupt flag (COMP) - See below for general information. When this bit changes from 0 to 1, a 1-cycle tick and system interrupt will be emitted. The bit must be cleared by writing 1 before a new tick or interrupt will be generated.

Each bit in this register is the interrupt flag for the corresponding bit in FIR*n*.STAT (see Table 322). An interrupt flag will be set to 1 if the interrupt is unmasked in FIR*n*.IMASK and an edge occurs in FIR*n*.STAT. The type of edge (rising/falling) is configured in FIR*n*.IEDGE. All configuration is per bit. An interrupt flag can be cleared by writing 1.

Table 324. 0x08 - FIR*n*.IMASK - FIR *n* interrupt mask register

31	2	1	0
RESERVED		PAR	COMP
0		0	0
r		rw	rw

Each bit in this register is the interrupt mask for the corresponding bit in FIR*n*.STAT (see Table 322). Set mask to 1 to enable the interrupt, 0 to disable.

Table 325. 0x0C - FIR*n*.IEDGE - FIR *n* interrupt edge register

31	2	1	0
RESERVED		PAR	COMP
0		1	1
r		rw	rw

Each bit in this register is the interrupt edge configuration for the corresponding bit in FIR*n*.STAT (see Table 322). Set to 1 for rising edge and 0 for falling edge.

28.2.5 FIR 0-7 Configuration Registers

*Table 326.*0x10 - FIR*n*.ICFG - FIR *n* input configuration registers

31 6	5	4 0
RESERVED	INV	SEL
0	0	0x00
r	rw	rw

31: 6 Reserved

5 FIR *n* invert input (INV) - 1 to invert the input to FIR *n*, 0 to use non-inverted input.

4: 0 FIR *n* input source selection (SEL) - A number from 0-31 selecting an input source. See Table 312 for available sources and mapping for each FIR instance.

*Table 327.*0x14 - FIR*n*.FCFG - FIR *n* filter configuration register

31 27	26 0
COMP	COEF[26:0]
0x00	0x0000000
rw	rw

31: 27 FIR *n* threshold value (COMP) - Whenever FIR*n*.SUM>COMP, then a FIR limit trigger occurs. See register FIR*n*.STAT in section 28.2.4.

26: 0 FIR *n* filter coefficients (COEF[26:0]) - COEF[0] is the coefficient for the newest input sample, COEF[26] is the coefficient for the oldest sample.

28.2.6 FIR 0-7 Filter output registers

*Table 328.*0x80 - FIR*n*.DATA - FIR *n* filter input data register

31 27	26 0
RESERVED	DATA[26:0]
0	0x0000000
r	rw

31: 27 Reserved

26: 0 FIR *n* filter input data register (DATA) - DATA[0] is the current value of the input signal of FIR *n*. DATA[1] is the input value as it was sampled on the previous rising edge of the sampling clock. DATA[*m*] is the input as it was sampled *m* sampling clock cycles ago. On every rising edge of the sampling clock the previous DATA[*m*] becomes the next DATA[*m*+1].

*Table 329.*0x84 - FIR*n*.SUM - FIR *n* filter output sum register

31 5	4 0
RESERVED	SUM
0	0x00
r	rw

31: 5 Reserved

4: 0 FIR *n* filter output sum (SUM) - The output of the FIR filter:

SUM = COEF[0]*DATA[0]+COEF[1]*DATA[1]+...+COEF[26]*DATA[26]

Here COEF and DATA are short for FIR*n*.FCFG.COEF and FIR*n*.DATA respectively.

*Table 330.*0x8110A080+4*n* - FIRG.OUT*n* - Alias for FIR*n* output registers

31 27	26 0
SUM	DATA
0x0	0x0000000
r	r

31: 27 FIR *n* filter output (SUM) - The same value as FIR*n*.SUM, see tableTable 329

26: 0 FIR n sampled data (DATA) - The same value as FIR*n*.DATA, see Table 328.

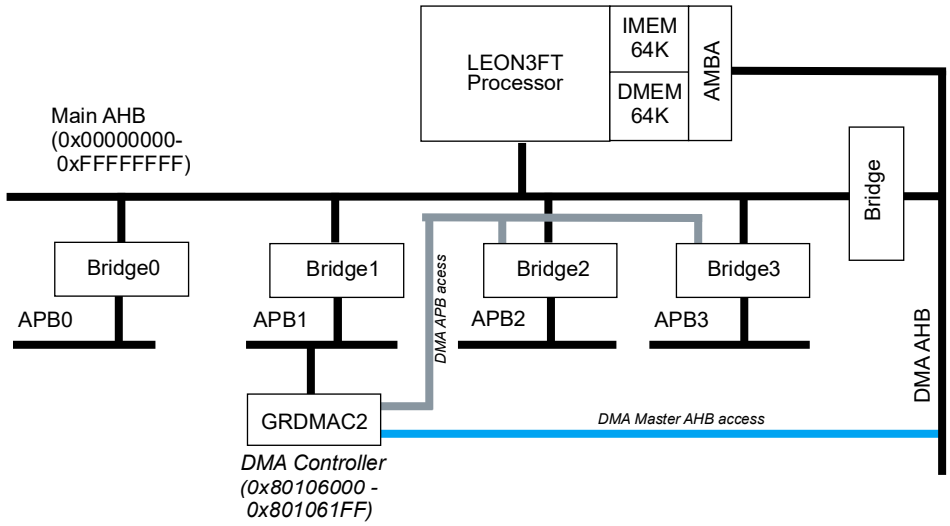
Note: This register combines the information in FIR*n*.SUM and FIR*n*.DATA into a single register. It is located in the FIRCLKGEN register area, not in the FIR.

29 Direct Memory Access Controller

The GR716B microcontroller has a DMA controller with internal AHB/APB bridge units (GRDMAC2). The GRDMAC2 units described in this section provides a flexible direct memory access controller. The core can perform burst transfers of data between AHB and APB peripherals at aligned or unaligned memory addresses.

The control and status register for the DMA controller units are located on APB bus in the address range from 0x80106000 to 0x801061FF. See DMA controller units connections in the next drawing. The drawing picture memory locations and bus connections used for DMA controller units.

Figure 72. GR716B GRDMAC bus connection



The primary clock gating unit **GRCLKGATE** described in section 27 is used to enable/disable the individual DMA controller units. The unit **GRCLKGATE** can also be used to perform reset of individual DMA controller units. Software must enable clock and release reset described in section 27 before configuration.

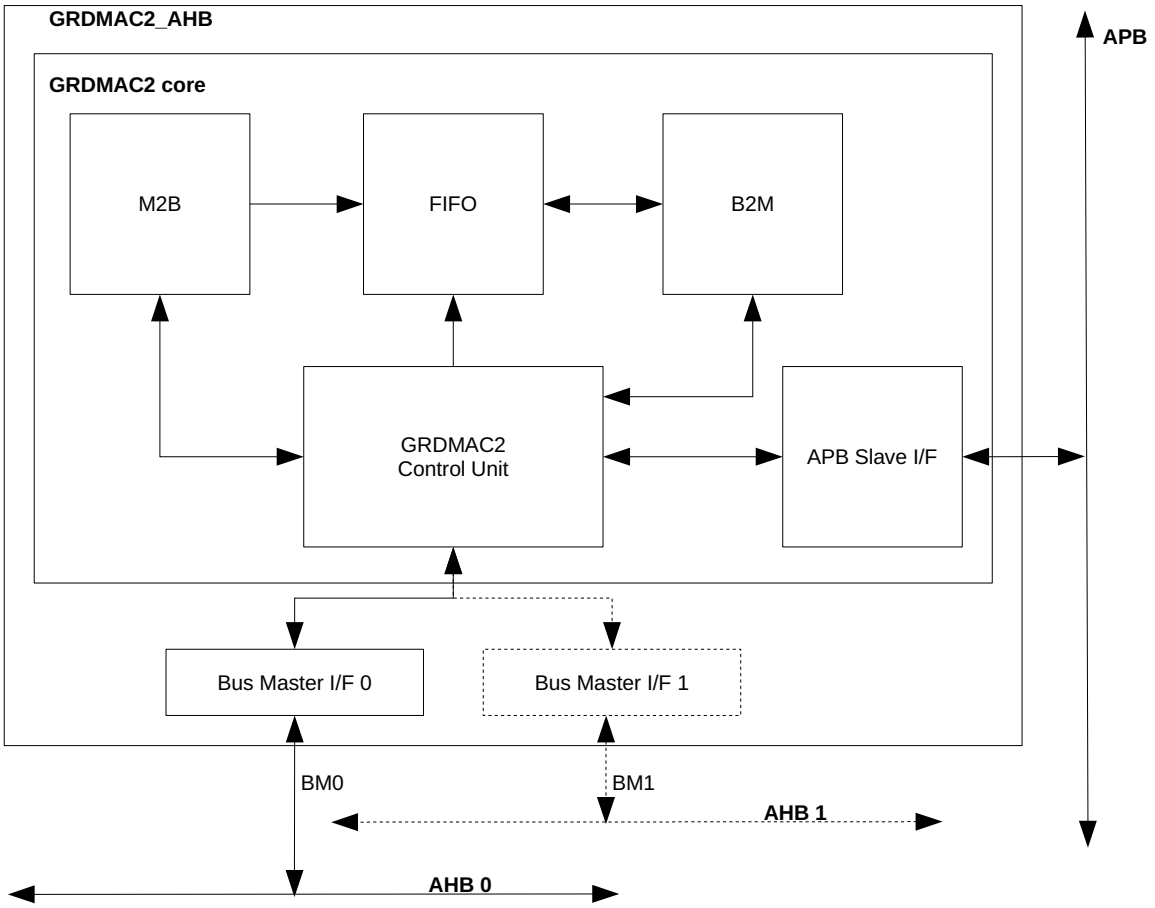
The system can be configured to protect and restrict access to DMA controller units.

29.1 Overview

GRDMAC2 is a flexible direct memory access controller with Bus Master Interface. The basic building blocks of GRDMAC2 are a control module, internal buffer, memory to buffer operation module(M2B), buffer to memory operation module(B2M) and a wrapper layer which provides Bus Master Interface integration and APB register configuration interface. GRDMAC2 can perform burst transfers of data between AHB and APB peripherals at aligned or unaligned memory addresses. Each GRDMAC2 have two Bus Master Interfaces (BM0 and BM1) to perform transfers among different AHB buses, therefore GRDMAC2 has a bridging capability. The APBCTRL2 and APBCTRL3 cores are connected using BM1, the core has direct access to these APB slaves. On-chip RAM, AHBSTAT, external memory and to other places using the BM0 bus master via bridge.

GRDMAC2 works with self contained descriptors, with all information needed for the DMA transaction. Descriptor configuration allows specification of source and destination addresses which allows a scatter/gather behavior. GRDMAC2 allows multiple types of descriptors which can be broadly classified as data descriptor and conditional descriptor. This enables GRDMAC2 to perform regular data transfer as well as perform an action on occurrence of a specific condition. Conditional descriptors allows an if-else mode of execution based on the condition check outcome. Figure below depicts the

basic block diagram of GRMAC2. The diagram assumes that the AHB wrapper is used, but the functionality is equivalent if the bus master or the AXI wrapper are chosen instead.



With the flexible descriptor queue with conditional descriptors and data descriptors, GRDMAC2 can be used with peripherals like APBUART and other controllers like SPI controller. A typical use case example of GRDMAC2 is described in section 29.6.

GRDMAC2 is equipped with multiple error flags for different kinds of error events and eight individual APB registers solely for debug capability of the current descriptor that is being executed. GRDMAC2 always displays the current execution state in status register and freezes this display in case of an error to make debugging easy.

29.2 Operation

29.2.1 Operation overview

When GRDMAC2 execution is enabled by the user by setting *'EN'* bit in the APB control register, descriptor execution starts from the first descriptor which is pointed by Descriptor Pointer APB Register. GRDMAC2 reads descriptor configurations from any AHB mapped address (typically main memory) through its main Bus Master Interface (BM0 if instantiated with support for two Bus Master Interfaces).
GRDMAC2 decodes the descriptor configuration, identifies the type of descriptor and continues with execution procedure specific to the current descriptor type. On completion of each of the descriptor, GRDMAC2 writes back the status to the descriptor status word, if '*wb*' bit is enabled in the descriptor. Execution continues with the next descriptor pointer by the current descriptor's next descriptor pointer field. In the case of a failed conditional descriptor, DMA controller selects address,

pointed by '*fnext.addr'* field, to fetch next descriptor. Fetch-Decode-Execute procedure is performed on each enabled descriptor in the descriptor queue. Any disabled descriptor in the queue is simply skipped and the DMA controller continues execution with the next enabled descriptor in the queue. Descriptor queue should be set up such that the last descriptor in the queue should have '*next.last'* bit value as 1. GRDMAC2 continues execution until it encounters a last descriptor. After completing the last descriptor execution successfully, GRDMAC2 marks the execution status as 'Completed' in the status register and stays idle.
Bus master interface index to be selected for conditional descriptor execution is determined by '*bm*' field in descriptor control word. GRDMAC2 allows the selection of individual Bus Master Interface index for source and destination in data descriptors. Descriptor fetches from the memory and write back to the descriptor status word in the memory is performed through default Bus Master Interface, irrespective of the type of descriptor.
The functionalities supported by GRDMAC2 for DMA operation is described in the following sections.

29.2.1.1 Pause and resume

When DMA controller has started processing the descriptors in the current descriptor queue, it is possible to pause the execution in between. Clearing the '*EN*' bit in GRDMAC2 control register(CTRL) will pause the descriptor queue processing. However, execution will pause only after successfully completing the current descriptor execution. GRDMAC2 sets the '*PAU*' bit as 1, and clears the '*ONG*' bit in the status register. Once paused, GRDMAC2 stays idle until '*EN*' and '*KICK*' bit is set in the CTRL register. Once the paused GRDMAC2 is enabled again and kicked by the user, execution resumes from the next descriptor in the queue, if there are enabled descriptors in the remaining part of queue.

29.2.1.2 Restart

GRDMAC2 restarts current descriptor queue execution when '*RT*' bit is set in GRDMAC2 control register (CTRL). Restarting the descriptor queue execution will be performed only after completing the current descriptor execution. DMA controller restarts execution from the first descriptor pointed by Descriptor Pointer APB Register. When '*RT*' bit is set by the user when a descriptor processing is in progress, the request is acknowledged and '*RSP*' bit is set in GRDMAC2 status register. This self clearing bit is cleared by GRDMAC2 when the descriptor queue execution is restarted after the current descriptor is executed completely.

29.2.1.3 Descriptor queue management

GRDMAC2 allows the modification of descriptor queue even after the DMA has started processing the current descriptor queue. Only appending new descriptors are allowed. In order to append new descriptor at the end of descriptor queue, user need to clear '*next.last*' bit of the last descriptor in the current queue and point to the new descriptor as the next descriptor. Mark the new descriptor as the last one. Adding multiple new descriptors also have the same procedure apart from the fact that the last descriptor will be at the end of the newly added part of the queue. Once the descriptor queue is modified, it is mandatory to kick GRDMAC2 to indicate the presence of modified descriptor queue. DMA controller acknowledges kick by setting '*KP*' bit in GRDMAC2 status register. '*KP*' bit is cleared by GRDMAC2 when a new descriptor is taken up for execution. Any type of descriptor queue modification other than appending at the end of the current descriptor queue, is illegal.

29.2.1.4 Reset

Setting '*RST*' bit in GRDMAC2 control register(CTRL) resets the DMA controller, canceling any ongoing descriptor execution. The reset clears all the register fields to their default values.

29.2.2 Data descriptor execution

During a standard data descriptor execution, GRDMAC2 will perform two types of DMA transfers through one of the Bus Master Interfaces: from memory to the internal buffer (M2B) and from the internal buffer to memory (B2M). GRDMAC2 splits the total size of data to be transferred for a data descriptor, in to data chunks of size equal to minimum of the following factors. Size of the internal buffer, maximum burst length or a maximum of 1024 bytes.
After fixing the size of each burst, the DMA controller transfers data chunks from source to destination through M2B and B2M operations. A chunk of data is transferred from memory to the internal buffer as part of the M2B operation and M2B operation is paused. Execution control is switched to B2M operation, where the DMA controller transfers the data chunk from the internal buffer to the destination memory and pauses B2M operation. GRDMAC2 then switches back to the M2B operation which was paused earlier and repeats the same process for the next chunk of data. M2B-B2M cycles are repeated until the total size of data transfer is completed and descriptor is marked as completed. On descriptor completion, based on whether '*wb'* bit is enabled in the descriptor control word, status of the descriptor execution is written back to the descriptor status word.
Data is transferred between source address and destination address which are configured in the descriptor. The offset of source address from which data is fetched is determined by the bit field '*src_fixed_addr*' in the descriptor control word. Similarly the offset of destination address to which data is written is determined by the bit field 'dest_fixed_addr' in the descriptor control word. If src_fixed_addr field is set to 1, GRDMAC2 always fetches data from the same source address. Similarly if '*dest_fixed_addr'* field is set to 1, GRDMAC2 always writes to the same destination address. If any of these fields are not set to 1, GRDMAC2 automatically considers incrementing offsets for source/destination addresses for data read/write. In short, GRDMAC2 can perform burst transactions in the following ways based on the different configurations of '*src_fixed_addr'* and '*dest_fixed_addr'* fields:

1. Data fetched from incrementing source address offsets, written to incrementing destination address offsets.
2. Data fetched from incrementing source address offsets, written to fixed destination address.
3. Data fetched from fixed source address, written to incrementing destination address offsets.
4. Data fetched from fixed source address, written to fixed destination address.

M2B and B2M operation can be performed through the same or different Bus Master Interface index. This selection is performed by configuring *'srcbm'* and *'dstbm'* in descriptor control word provided that second Bus Master Interface.

- **Data descriptor execution example**

Steps of a data descriptor execution example is given below.

a) GRDMAC2 starts M2B operation.
b) GRDMAC2 checks size of data to be transferred between source and destination for descriptor completion. GRDMAC2 compares the size of internal buffer, maximum burst length and maximum boundary of 1024 bytes and finds out the current burst size.
c) A read burst initiated on the selected Bus Master Interface transfers data from source memory address (incrementing offsets if *'src_fid_addr'* field is not set.) to GRDMAC2 internal buffer. If *'src_fid_addr'* field is set in the descriptor, data is always read from the same source address.
d) GRDMAC2 checks if there is data yet to be fetched from source to complete the descriptor execution. If yes goes to step **e**. Otherwise goes to step **f**.
e) GRDMAC2 pauses M2B operation and switches to B2M operation. Goes to step **g**.
f) GRDMAC2 completes M2B operation and Switches to B2M operation. Goes to step **g**.
g) GRDMAC2 checks size of data to be transferred between source and destination for descriptor completion. GRDMAC2 compares the size of internal buffer, maximum burst length and maximum

boundary of 1024 bytes and finds out the current burst size.
h) A write burst is initiated on the selected Bus Master Interface transfers data from GRDMAC2 internal buffer to destination memory address(incrementing offsets if *'dest_fix_addr'* field is not set). If *'dest_fix_addr'* field is set in the descriptor, data is always written to the same destination address.
i) GRDMAC2 checks if there is data yet to be written to destination in order to complete the descriptor execution. If yes goes to step **j**. Otherwise goes to step **k**.
j) GRDMAC2 pauses B2M operation and switches to M2B operation. Goes to step **b**.
k) GRDMAC2 completes B2M operation. Mark descriptor as completed.
l) Writes back the execution status to descriptor status word, if *'wb'* bit is enabled in the descriptor control word.
m) Fetches the next descriptor (pointed by '*next.addr'*) in the queue if there are any. On the other hand, if the current descriptor is the last, marks the execution completion in GRDMAC2 status register.

29.2.3 Conditional descriptor execution

A conditional descriptor is a special kind of descriptor which can be executed to perform a task with conditional behavior. A conditional descriptor can be used to create a DMA transfer that retrieves data from IO cores, therefore off loading the CPU from the task. Usually IO cores provide a status register or an interrupt line to notify the CPU of the availability of new data. A conditional descriptor can be set up to poll this status register or to be triggered by an interrupt, signaling for instance, the availability of new data. This is a typical use case of conditional descriptor execution.

GRDMAC2 identifies a conditional descriptor from the *'type'* field in the descriptor control word. Current version of GRDMAC2 supports 3 types of conditional descriptors. Polling type('*type'* = 1), Triggering type('*type'* = 2) and Poll-on-trigger type('*type*' = 3). The field '*bm*' in descriptor control word, decides the Bus Master Interface index which performs memory accesses during conditional descriptor execution.

Conditional descriptor *'next.addr'* field points to the next descriptor's control word address. Setting *'next.last'* bit marks the current descriptor as the last descriptor in the descriptor queue. Conditional descriptors allows an 'if-else' method of condition check by *'next'* and *'fnext'* fields. On successful execution of a conditional descriptor, GRDMAC2 execution control continues to '*next.addr'* as long as the *'next.last'* bit is not set. On the other hand, if the conditional descriptor execution was failed, execution continues with the next descriptor pointed by *'fnext'*. It is mandatory that *'fnext.last'* bit is never set. Setting this bit will result in a decode descriptor error and *'DE'* will be set in GRDMAC2 status register. This applies to all types of conditional descriptors.
Conditional descriptor fields like *'poll.addr'*, *'expd.data'* and *'cond.mask'* are relevant only for polling or Poll-on-trigger type of descriptors.

29.2.3.1Polling descriptor execution

A polling type descriptor should have *'ctrl.type'* field value 1 in descriptor control word. During a polling descriptor execution, GRDMAC2 polls the address configured in *'poll.addr'* field. Polling is performed in specific number of clock cycle intervals. Interval between polling accesses is decided by *'intrv'* field in descriptor control word. Setting interval between each poll access avoids holding the bus and makes the bus available for other cores in the system.
Polling result data is compared with expected data value configured in *'expd.data'* descriptor field and conditional mask for comparison configured in *'cond.mask'* descriptor field to evaluate the success of the conditional descriptor execution. Condition check is performed as follows.

Table 331. Condition for successful polling descriptor execution

(Content of 'poll.addr') AND 'cond.mask' = 'expd.data' AND 'cond.mask'

In other words, the polling result and expected data is compared only on the bit positions that are set in the conditional mask field. After each polling, condition is checked and this polling-condition check process id repeated for specific number of times based on the value configured in '*count*' field in descriptor control word. If the condition check is not successful even after repeating the polling-condition check process *'count'* times, polling descriptor execution will have a failed status. Otherwise if the condition check was successful before *'count'* times, the polling descriptor execution will be completed successfully.
Apart from the iteration limit on Polling and condition check process, polling descriptor execution is bound by a timeout mechanism if timeout check is enabled in GRDMAC2 control register(CTRL) and during controller instantiation. GRDMAC2 waits for number of clock cycles specified in APB register '*TRST*'(Timer reset value register) before generating a timeout status. Timeout is interpreted either as a failed execution or as an error, based on the field '*errto*' in descriptor control word. Setting '*ctrl.errto*', treats timeout as an error and a failed condition otherwise. On successful condition check, GRDMAC2 marks the descriptor as completed and moves on with the next descriptor execution if there are any present. If the polling descriptor execution is failed, GRDMAC2 fetches next descriptor from address pointed by *'fnext.addr'* descriptor field and continues execution.

- **Polling descriptor execution example**

Steps of a polling descriptor example is given below. A counter is decremented from reset value as soon as the polling descriptor execution has started. Reset value of this counter can be configured through GRDMAC2 '*TRST'* register.

a) GRDMAC2 checks if timeout counter value is zero. If yes, Goes to step **i**. Goes to step **c** otherwise

b) GRDMAC2 polls memory address configured in *'poll.addr'*. Starts incrementing counter(C0) which increments every clock cycle. Goes to step **d**.

c) Clears the counter C0. Goes to step **b**.

d) Increment a polling iteration count by 1. Goes to step **e**.

e) GRDMAC2 compares the read data with *'expd.data'* and *'cond.mask'* based on condition in table 331. If condition check is successful, Goes to step **h**. If the condition check is unsuccessful goes to step **f**.
f) GRDMAC2 checks if polling iteration count is equal to *'count*' configured in descriptor control word. If yes goes to step **k**, else goes to step **g**.
g) Checks if the counter (C0) value is equal to *'intrv'* configured in descriptor control word. If yes goes to step **a**, else goes to step **g**.
h) Marks descriptor as completed. Writes back the execution status to descriptor status word, if '*wb'* bit is enabled in the descriptor control word. Goes to step **j**.
i) Marks timeout. If *'errto'* field is set in the descriptor control word, marks error bit *'ERR'* and '*PE'* bit in GRDMAC2 status register and stop ongoing execution. If *'errto'* field is not set in the descriptor control word, goes to step **k**.
j) Fetches the next descriptor(pointed by *'next.addr'*) in the queue if there are any. On the other hand, if the current descriptor is the last, marks the execution completion in GRDMAC2 status register.
k) Writes back descriptor status if *'wb'* bit is enabled in the descriptor control word. Fetches the next descriptor pointed by *'fnext.addr'* and continue execution.

29.2.3.2 Triggering descriptor execution

A triggering type descriptor must have *'type'* field value 2 in descriptor control word. Triggering descriptor monitors the input trigger on the interrupt line specified in the field *'irqn'* in the descriptor control word. If the timeout mechanism is enabled, the descriptor execution continues the process of monitoring for input trigger, either till the expected trigger is received or till timeout. Based on the *'errto'* bit field in the control word of a triggering descriptor, timeout is treated either as an error or as a failed descriptor execution status.

On successful execution of triggering descriptor, GRDMAC2 marks the descriptor as completed and moves
on with the next descriptor execution if there are any present. If triggering descriptor execution is failed, GRDMAC2 fetches next descriptor from address pointed by *'fnext.addr'* descriptor field and continues execution.
The expected trigger event type is configured by *'ctrl.trtp'* field and *'poll.trig_val'* field in the descriptor. Setting *'ctrl.trtp'* field to 1 in the descriptor will configure the expected event as level and clearing *'ctrl.trtp'* field will configure the expected event as edge. Setting *'poll.trig_val'* field to 1 in the descriptor will configure the expected event as a negative edge/
low level(based on the *'ctrl.trtp'* value). Clearing *'poll.trig_val'* field in the descriptor will configure the expected event as a positive edge/high level(based on the *'ctrl.trtp'* value).

- **Triggering descriptor execution example**

Steps of a triggering descriptor example is given below. A counter is decremented from reset value as soon as the triggering descriptor execution has started. Reset value of this counter can be configured through GRDMAC2 '*TRST'* register.

a) GRDMAC2 checks if timeout counter value is zero. If yes, Goes to step **e**. Goes to step **b** otherwise.
b) GRDMAC2 monitors the interrupt line number *'irqn'* configured in descriptor control word for expected event.
c) If expected event occurs, goes to step **d**, else goes to step **a**.
d) Marks descriptor as completed. Writes back the execution status to descriptor status word, if *'wb'* bit is enabled in the descriptor control word. Goes to step **f**.
e) Marks timeout. If *'errto'* field is set in the descriptor control word, Marks error bit *'ERR'* and *'PE'* bit in GRDMAC2 status register and stop ongoing execution. If *'errto'* field is not set in the descriptor control word, goes to step **k**.
f) Fetches the next descriptor (pointed by *next.addr*) in the queue if there are any. On the other hand, if the current descriptor is the last, marks the execution completion in GRDMAC2 status register.
g) Writes back descriptor status if '*wb*' bit is enabled in the descriptor control word. Fetches the next descriptor pointed by *fnext.addr* and continue execution.

29.2.3.3 Poll-on-trigger descriptor execution

A poll-on-trigger type descriptor must have '*ctrl.type*' field value 3 in descriptor control word. Poll-on-trigger type descriptor execution combines the functionality of a triggering descriptor and a polling descriptor. Poll-on-trigger type of descriptor execution monitors the input trigger on the interrupt line specified in the field '*irqn*' in the descriptor control word. On reception of the expected trigger on IRQ line of interest, GRDMAC2 polls the address configured in '*poll.addr*' field. Each polling access is separated by '*intrv*' number of clock cycle as configured in descriptor control word. After each polling the result(content of '*poll.addr'* address) is compared with expected data('*expd.data'*) and mask(*'cond.mask'*) using the condition described in table 331. Polling and condition check process is repeated for *'count'* times as configured in the descriptor control word. Both trigger monitoring and polling are individually bound by timeout if timeout mechanism is enabled.

On successful execution of poll-on-trigger descriptor, GRDMAC2 marks the descriptor as completed and moves on with the next descriptor execution if there are any present. If poll-on-trigger descriptor execution is failed, GRDMAC2 fetches next descriptor from address pointed by '*fnext.addr'* descriptor field and continues execution.

29.3 Configuration

The DMA controller and descriptors should be configured properly for successful operation of the controller. The following sections describes the DMA controller and descriptor configuration in detail.

29.3.1 GRDMAC2 configuration

GRDMAC2 can be configured through APB register interface. The following APB registers should be used to configure DMA controller.

- **Control register (CTRL)**

Control register allows the configuration of interrupt generation on error and descriptor completion. Control register also allows the user to enable timeout mechanism for conditional descriptors. Timeout mechanism for conditional descriptors can be enabled by setting *'TE'* bit in control register. Conditional descriptor execution is bound by timeout if timeout mechanism is enabled. Conditional descriptor execution will continue for ever or until the successful condition check result if timeout is disabled.

There are designated bits which allows different functionalities like kicking, enabling, resetting, pausing and restarting GRDMAC2. These functionalities are explained in section 29.2.1, in detail.

- **Timer reset value register (TRST)**

Configuring TRST register has an impact only on the conditional descriptor execution, provided that timeout mechanism is enabled. During a conditional descriptor execution, GRDMAC2 will wait for, number of clock cycles configured in Timer reset value register, before generating a timeout status.

- **First descriptor pointer (FPTR)**

First descriptor pointer register should be configured to point to the first descriptor in a descriptor queue.

29.3.2 Descriptor configuration

Descriptor types supported by GRDMAC2 can be broadly classified as data and conditional descriptors. Conditional descriptors are further classified based on the type of condition check they are trying to satisfy. GRDMAC2 demands pre-defined descriptor configuration for each of these types, as described in sections 29.3.2.1 and 29.3.2.2. It is mandatory that descriptors are written to two byte aligned addresses in the memory. By default, Bus Master Interface index 0 is selected for all descriptor reading and execution status write back.

29.3.2.1Data descriptor format

Data descriptor format and description of individual fields are displayed below.

Table 332. Data descriptor format

Address offset	Field
0x00	Control word
0x04	Next descriptor pointer
0x08	Destination base address
0x0C	Source base address
0x10	Status word

Table 333. 0x00 - ctrl - Data descriptor control word

31 11	10	9	8	7	6	5	4 1	0
size	destfix	srcfix	irqe	dstbm	srcbm	wb	type	en

31 : 11	size	Total size of data to be transferred from source to destination.
10	destfix	All data is to be written to the same(fixed) destination address 0- Non fixed destination address(incrementing address offsets) 1- Fixed destination address
9	srcfix	All data is to be read from the same(fixed) source address 0- Non fixed source address(incrementing address offsets) 1- Fixed source address
8	irqe	Enable interrupt on descriptor completion 0- Disable interrupt 1 – Enable interrupt
7	dstbm	Bus master interface index through which data is to be written to the destination. 0 - BM0 1 - BM1
6	srcbm	Bus master interface index through which data is to be read from the source. 0 - BM0 1 - BM1
5	wb	Write back status to current descriptor status word, after execution. 0- disable write back 1- enable write back
4 : 1	type	Descriptor type. 0- data descriptor
0	en	Enabled data descriptor. 0- Disabled 1- Enabled

Table 334. 0x04 - next - Next descriptor pointer

31 1	0
addr	last

31 : 1	addr	MSB of 2 byte aligned next descriptor start address.
0	last	Last descriptor in the descriptor queue. 0 - Not last descriptor 1 - Last descriptor

Table 335. 0x08 - dest - Destination base address

31	0
addr	

31 : 0	addr	Destination base address to which data is to be written.

Table 336. 0x0C - src - Source base address

31			0
addr			
31 : 0	addr	Source base address to which data is to be written.	

Table 337. 0x10 - sts - Descriptor status word

31			2	1	0
Reserved				err	done
31 : 2		Reserved			
1	err	Descriptor execution error status. 0 - No error. 1 - Error during execution			
0	done	Descriptor completion without error. 0 - Not completed 1 - Completed			

29.3.2.2 Conditional descriptor format

Conditional descriptor format and description of individual fields are displayed below.

Table 338. Conditional descriptor format

Address offset	Field
0x00	Control word
0x04	Next descriptor pointer
0x08	Next descriptor pointer on failure
0x0C	Polling address
0x10	Status word
0x14	Expected data
0x18	Conditional mask

Table 339. 0x00 - ctrl - Conditional descriptor control word

31 24	23 16	15	14	13	12 7	6	5	4 1	0
count	intrv	trtp	irqe	errto	irqn	bm	wb	type	en

31 : 24	count	Number of time the polling and condition check to be repeated before assuming a failed execution status
23 : 16	intrv	Number of clock cycle interval between consecutive polling accesses.
15	trtp	Type of input trigger event expected. 0- Edge 1- Level
14	irqe	Enable interrupt on descriptor completion 0- Disable interrupt 1 – Enable interrupt
13	errto	Timeout is to be treated as an error or a failed conditional descriptor execution. 0 - Timeout is a failed conditional descriptor execution, 1 - Timeout is an error.
12 : 7	irqn	Interrupt line number to be monitored for input trigger, if the current descriptor is a triggering/poll-on-trigger descriptor.
6	bm	Bus master interface index through which polling accesses are to be performed, if the current descriptor is a polling descriptor. 0 – BM0 1 - BM1
5	wb	Write back status to current descriptor status word, after execution. 0- disable write back 1- enable write back
4 : 1	type	Descriptor type. 1- Polling 2- Triggering 3- Poll-on-trigger
0	en	Enabled data descriptor. 0- Disabled 1- Enabled

Table 340. 0x04 - next - Next descriptor pointer

31 1	0
addr	last

31 : 1	addr	MSB of 2 byte aligned next descriptor start address. This pointer will be considered for execution when current conditional descriptor has executed successfully.
0	last	Last descriptor in the descriptor queue. 0 - Not last descriptor 1 - Last descriptor

Table 341. 0x08 - fnext - Next descriptor pointer on failure

31			1	0
addr				last
31 : 1	addr	MSB of 2 byte aligned next descriptor start address. This pointer will be considered for execution when current conditional descriptor has failed.		
0	last*	Last descriptor in the descriptor queue. 0 - Not last descriptor		

* *This field should not be set ever. Setting fnext.last field will generate a decode descriptor* error

Table 342. 0x0C - poll - Polling address

31			1	0
addr				trig_ val
31 : 1	addr	MSB of 2 byte aligned address to be polled during descriptor execution if the current descriptor is a polling descriptor.		
0	trig_val	Expected input trigger value if the current descriptor is a triggering descriptor 0- Positive edge/high level based on the ‘ctrl.trtp’ field value of the descriptor. 1 - Negative edge/low level based on the ‘ctrl.trtp’ field value of the descriptor.		

Table 343. 0x10 - sts - Descriptor status word

31			3	2	1	0
Reserved				cpass	err	done
31 : 3	-	Reserved				
2	cpass	Status of condition check. 1 - Conditional descriptor executed successfully. 0 - Conditional descriptor execution failed.				
1	err	Descriptor execution error status. 0 - No error. 1 – Error during execution				
0	done	Descriptor completion without error. 0 - Not completed 1 - Completed				

Table 344. 0x14 - expd - Expected data

31			0
data			
31 : 0	data	Expected data for polling type of descriptor execution. GRDMAC2 compares polling result with this data during descriptor execution.	

Table 345. 0x18 - cond - Conditional Mask

31		0
mask		
31 : 0	mask	Mask for polling type of descriptor execution. GRDMAC2 compares polling result with this mask during descriptor execution. Polling result data bits corresponding to mask bits which are set to one, is only considered for condition check comparison with expected data (‘expd.data’).

29.4 Interrupts

GRDMAC2 provides fine-grained control of interrupt generation. At the highest level, the global Interrupt Enable bit (*‘IE’*) in GRDMAC2 control register(CTRL) can be set to zero to mask every interrupt setting in GRDMAC2. If set to one, interrupt generation depends on the following settings. The Interrupt on Error Enable bit (*‘IER’*) in CTRL register provides a way to generate interrupts in the event of errors. Error generation is discussed further in section 29.5. An interrupt can be also generated by the successful completion of a descriptor, if the Interrupt Enable (*‘irqe’*) bit is set to one in the descriptor’s control field. The Interrupt Mask bit (*‘IM’*) in the CTRL register can be set to one to mask the descriptor completion interrupts. Descriptor completion interrupt will be generated as soon as the successful completion of the descriptor execution. Even before writing back the descriptor’s status in main memory, if the write back is enabled. For both interrupts on error and interrupts on descriptor completion events, a flag will be raised in the interrupt flag bit(*‘IF’*) in GRDMAC2 status register. Interrupt generated on descriptor completion can be masked by configuring *‘IM’* bit even though *‘IE’* bit in CTRL register and ‘*irqe*’ bit in descriptor control word is set to one. However it is not possible to mask the interrupt generated on error when *‘IE’* bit in CTRL register is set to one. As an example of interrupt generation setup, one can perform the following steps. The Interrupt Enable (*‘IE’*) bit in CTRL register must be set to one. The Interrupt Mask (*‘IM’*) bit in CTRL register must be configured as zero. Set up a descriptor queue with a single data descriptor. Configure Interrupt Enable (*‘irqe’*) bit in the control field of the descriptor as one and mark the descriptor as the last descriptor. When the descriptor queue is completed successfully, an interrupt will be generated.

29.5 Status and Errors

GRDMAC2 provides a very clear status display by an APB status register and 8 separate debug capability registers which displays details of the current descriptor that is being executed.

There is a general error flag *‘ERR’* which will be set in any case of error and an individual flag which says the type of error that has occurred. A decode descriptor error(*‘DE’*) can occur during the decoding of a descriptor if the type of descriptor is not a valid value(0 to 3 is only valid) for current version of GRDMAC2. Another situation which can generate this error is when a conditional descriptor has it’s *‘fnext.last’* bit set to one. A read descriptor error(*‘RE’*) will be flagged in GRDMAC2 status register in case when the Bus Master Interface receives an error response during the descriptor read burst memory access. A polling error flag(*‘PE’*) can be set in two cases. One case is when a Bus Master Interface receives an error response during the read access on the address being polled.

Another case is when timeout occurs during a polling descriptor execution and the descriptor is configured to treat timeout as an error. A triggering error(*‘TRE’*) will be generated when a triggering type descriptor encounters a timeout and the descriptor is configured to treat timeout as an error. When the Bus Master Interface receives an error response during the write access on descriptor status word in the memory, a write back error(*‘WBE’*) is flagged in GRDMAC2 status register. If the Bus Master Interface receives an error response during any part of the read accesses performed as part of M2B operation, M2B read data error flag(*‘RDE’*) will be set status register. Similarly, in case the Bus Mas-

ter Interface receives an error response during any part of the write accesses performed as part of B2M operation, B2M write data error

flag(*'WDE'*) will be set in status register. A FIFO error(*'FE'*) is flagged in GRDMAC2 status register only when an uncorrectable error status is reported from the internal FIFO. In such a situation the *'FP'* field will be pointing to the offset in FIFO where the error has been detected. When GRDMAC2 receives a kick request, it reads the next descriptor pointer of the current descriptor again. If the Bus Master Interface receives an error response while this read access, a *'NPE'* flag will be set in APB status register. Timeout status flag *'TO'* will be set in case of a timeout during the conditional descriptor execution. All error flags are cleared on writing 1 to them. However the timeout flag *'TO'* is self clearing. In case of an error, GRDMAC2 pauses the execution and stays idle. Along with appropriate error flags, *'PAU'* flag will be set in GRDMAC2 status register. GRDMAC2 allows the user to clear the errors and kick GRDMAC2 to continue execution from the next descriptor in the queue.

GEDMAC2 has a 5 bit *'ST'* field which always displays the current state of descriptor execution. In case of any error, the *'ST'* field will freeze at the execution state where the error occurred. When GRDMAC2 is performing descriptor execution in a descriptor queue, the '*ONG'* bit in GRDMAC2 status register will be set to one. In case of an error or if GRDMAC2 has completed the execution of the entire queue, GRDMAC2 stays idle and clears the *'ONG'* bit, indicating that no operation is currently in progress. *'CMP'* flag is set to one, only when the descriptor queue is executed completely.

There are two self clearing flags *'KP'* and *'RSP'* which shows the existence of a pending kick request and a pending restart request. *'KP'* flag will be cleared whenever a new descriptor is taken up for execution. In other words, GRDMAC2 handles kick by reading the *'next'* field of the current descriptor again from the memory and processing it. *'RSP'* bit is cleared when the current descriptor execution is completed and DMA operation restarted from the first descriptor. The debug capability registers displays all the fields of the current descriptor that is being executed.

The descriptor pointer debug capability register shows the base address where the current descriptor was read from.

29.6 GRDMAC2 Use case Example

A simple example showing how GRDMAC2 can be used with APBUART is explained below. In this example GRDMAC2 is used to read data, whenever a peripheral APBUART receives data from a host. GRDMAC2 transfers data from the APBUART receiver register to main memory. Assume that the system has SRAM memory at 0x30000000 and UART configuration registers are located at 0x80300000.

Set up a descriptor queue with 3 descriptors. A polling descriptor followed by a data descriptor and a disabled data descriptor. Descriptor queue should be written to a memory location (assuming 0x30000000). The descriptor configuration is listed below.

a) Polling descriptor:

Table 346. GRDMAC2 use case example- Conditional descriptor configuration

Field	Value	Comment
ctrl.en	1	Enabled descriptor
ctrl.type	1	Polling descriptor
ctrl.wb	1	Execution status write back is enabled
ctrl.irqn	-	Irrelevant for polling descriptors
ctrl.trtp	-	Irrelevant for polling descriptors
ctrl.intrv	0xFF	0xFF clock cycles between two polling accesses
ctrl.count	0xFF	Repeat polling condition check for 0xFF iterations before assuming failed execution status
ctrl.bm	1	Bus Master Interface 1 is to be used for UART status register polling
ctrl.errto	0	Timeout is not an error
ctrl.irqe	1	Interrupt enabled on descriptor completion
next.addr	0x3000001C	Pointer to next descriptor
nxt_des.last	0	Not last descriptor
f_nxt_des.addr	0x30000030	Pointer to next descriptor if the current descriptor execution failed.
f_nxt_des.last	0	Mandatory value
poll.trg_val	-	Irrelevant for polling descriptors
poll.addr	0x80300004	APBUART status register address.
expd.data	0x00000001	Expects data ready to be 1. Break, Framing error, Parity error and Overrun bits to be 0.
cond.mask	0x00000079	Consider bit 0, 3, 4, 5 and 6 for comparison using the condition check in table 331

b) Data descriptor 1

This descriptor is written at 0x3000001C

Table 347. GRDMAC2 use case example- Data descriptor configuration

Field	Value	Comment
ctrl.en	1	Enabled descriptor
ctrl.type	0	Data descriptor
ctrl.wb	1	Execution status write back is enabled
ctrl.dstbm	0	Bus master interface 0 is to be used for data write to SRAM area.
ctrl.srcbm	1	Bus master interface 1 is to be used for data fetch from UART RX register.
ctrl.destfix	0	Write data to incrementing destination address offsets
ctrl.srcfix	1	Fetch data always from the same source address
ctrl.irqe	1	Interrupt enabled on descriptor completion
ctrl.size	64	Transfer 64 bytes of data
next.addr	0x30000000	Pointer to next descriptor. This essentially creates a loop of polling- data transfer process
nxt_des.last	0	Not last descriptor
dest.addr	0x30000080	SRAM area where data read from UART is to be written
src.addr	0x80300000	APBUART data register from where data is to be read.

5. Write the same data descriptor described in step **b** at address 0x30000030 with *'ctrl.en'* bit as zero and *'nxt_des.last'* bit as 1. This disabled data descriptor is pointed as the descriptor to be executed on failure for the first polling descriptor.

6. Configure GRDMAC2 '*FPTR'* register as 0x30000000, control register bit field *'TE'* as zero.

7. After configuring descriptor queue and GRDMAC2 registers, start GRDMAC2 operation by setting GRDMAC2 control register(CTRL) *'EN'* bit field as one.

GRDMAC2 should transfer data from UART data register whenever the data ready bit in APBUART status register is set. Transferred data can be read from the destination SRAM area and verified.

29.7 Registers

GRDMAC2 is programmed through registers mapped into APB address space.

Table 348. GRDMAC2 APB registers

APB address offset	Register
0x00	Control register
0x04	Status register
0x08	Timer reset value register
0x0C	Capability register
0x10	First descriptor pointer
0x14	Descriptor control word for debug capability
0x18	Next Descriptor pointer for debug capability
0x1C	fnext/dest.addr for debug capability*
0x20	poll.addr/src.addr for debug capability*
0x24	Descriptor status for debug capability
0x28	expd.data/Null for debug capability*
0x2C	cond.mask /Null for debug capability*
0x30	Current descriptor pointer for debug capability

*Debug capability register field according to the current descriptor type is conditional/data

29.7.1 Control register

Table 349. 0x00 - CTRL - Control register

31	8	7	6	5	4	3	2	1	0
Reserved		TE	IER	IM	IE	RT	KICK	RST	EN
0		0	0	0	0	0	0	0	0
r		rw	rw	rw	rw	rw	rw	rw	rw

31 : 8	-	Reserved
7	TE	Timeout check enable during conditional descriptor execution.
6	IER	Enable interrupt generation on error events
5	IM	Mask interrupt generation on descriptor completion.
4	IE	Enable interrupt generation. 0 - Disable interrupt all type of generation 1- Enable interrupt generation
3	RT	Restart current descriptor queue execution.
2	KICK	Kick GRDMAC2. GRDMAC2 reads the next descriptor pointer word of the current descriptor again when there is a kick request.
1	RST	Reset GRDMAC2. Setting this bit to one resets the core completely.
0	EN	Enable DMA controller.

29.7.2 Status register

Table 350. 0x04 - STS - Status Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NPE	WDE	RDE	TO	WBE	TRE	PE	RE	DE	KP	RSP	IF	PAU	ONG	ERR	CMP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
wc	wc	wc	wc	wc	wc	wc	wc	wc	r	r	wc	r	r	wc	r

31 27	26 17	16
ST	FP	FE
0	0	0
r	r	wc

31 : 27	ST	Current GRDMAC2 operation state
26 :17	FP	FIFO offset at which error encountered.
16	FE	FIFO error. Error reported from GRDMAC2 internal FIFO.
15	NPE	Error during reading next descriptor pointer register, on a kick request
14	WDE	Error while writing data during B2M operation.
13	RDE	Error while reading data during M2B operation.
12	TO	Timeout during conditional descriptor execution.
11	WBE	Write back error.
10	TRE	Triggering descriptor error.
9	PE	Polling error.
8	RE	Read descriptor error.
7	DE	Decode descriptor error.
6	KP	Pending kick request.
5	RSP	Pending restart request.
4	IF	Interrupt flag.
3	PAU	Paused descriptor queue execution.
2	ONG	Ongoing descriptor queue execution.
1	ERR	Error during descriptor queue execution.
0	CMP	Completed execution of the descriptor queue.

29.7.3 Timer reset value register

Table 351. 0x08 - TRST - Timer reset value register

31	0
VAL	
0	
rw	

31 : 0	VAL	Reset value of the timeout counter for timeout check mechanism.

29.7.4 Capability Register

Table 352. 0x0C - CAP - Capability Register

31 28	27 12	11 9	8	7 5	4	3 0
BFDP	Reserved	DW	TE	FT	BM1	VER
4	0	2	1	1	1	0
r	r	r	r	r	r	r

31 : 28	BFDP	Address bits of internal FIFO.
27 : 12	-	Reserved
11 : 9	DW	Bus master interface front end data width is set to 32 bits.
8	TE	Timeout mechanism support.
7 : 5	FT	Fault tolerant support
4	BM1	Second Bus Master Interface support.
3 : 0	VER	Revision of GRDMAC2

29.7.5 First descriptor pointer

Table 353. 0x10 - FPTR - First descriptor pointer

31	0
addr	
0	
rw	

31 : 0	addr	First descriptor pointer register. Points to first descriptor of the descriptor queue.

29.7.6 Descriptor control word for debug capability

Table 354. 0x14 - DCTR - Descriptor control word for debug capability

31	0
ctrl	
0	
r	

31 : 0	ctrl	Current descriptor's control field for debug capability.

29.7.7 Next Descriptor pointer for debug capability

*Table 355.*0x18 - DNXT - Next Descriptor pointer for debug capability

31	0
addr	
0	
r	

31 : 0	addr	Current descriptor's next descriptor field for debug capability

29.7.8 fnext/dest.addr for debug capability

*Table 356.*0x1C - DFNX - fnext/dest.addr for debug capability

31	0
addr	
0	
r	

31 : 0	addr	Current descriptor's next descriptor field on failure for debug capability/ Current descriptor's destination address field for debug capability*

* Debug capability field depends on the descriptor type.

29.7.9 poll.addr/src.addr for debug capability

Table 357. 0x20 - DPOL- poll.addr/src.addr for debug capability

31	0
addr	
0	
r	

31 : 0	addr	Current descriptor's 'poll.addr' for debug capability/Current descriptor's src.addr field for debug capability*

* Debug capability field depends on the descriptor type.

29.7.10 Descriptor status for debug capability

Table 358. 0x24 - DSTS - Descriptor status for debug capability

31	0
sts	
0	
r	

31 : 0	sts	Current descriptor's status word for debug capability

29.7.11 Expected data for debug capability

*Table 359.*0x28 - DDTA- Expected data for debug capability

31	0
data	
0	
r	

31 : 0	data	Current descriptor's expected data(expd.data) for debug capability*

* This field value is null for a data descriptor.

29.7.12 Conditional mask for debug capability

Table 360. 0x2C - DMSK - Conditional mask for debug capability

31	0
mask	
0	
r	

31 : 0	mask	Current descriptor's conditional mask for debug capability*

* This field value is null for a data descriptor.

29.7.13 Current descriptor pointer for debug capability

Table 361. 0x30 - DPTR - Current descriptor pointer for debug capability

31	0
addr	
0	
r	

31 : 0	addr	Pointer from which the current descriptor is read, for debug capability

30 General Purpose I/O Port

The GR716B microcontroller has 2 separate General Purpose I/O port (GRGPIO) units. Each General Purpose I/O port (GRGPIO) units controls its own external pins and has a unique AMBA address described in chapter 2.10.

The General Purpose I/O port (GRGPIO) units are located on APB bus in the address range from 0x8030C000 to 0x8030CFFF and 0x8030D000 to 0x8030DFFF. See General Purpose I/O port (GRGPIO) units connections in the next drawing. The figure shows memory locations and functions used for General Purpose I/O port (GRGPIO) configuration and control.

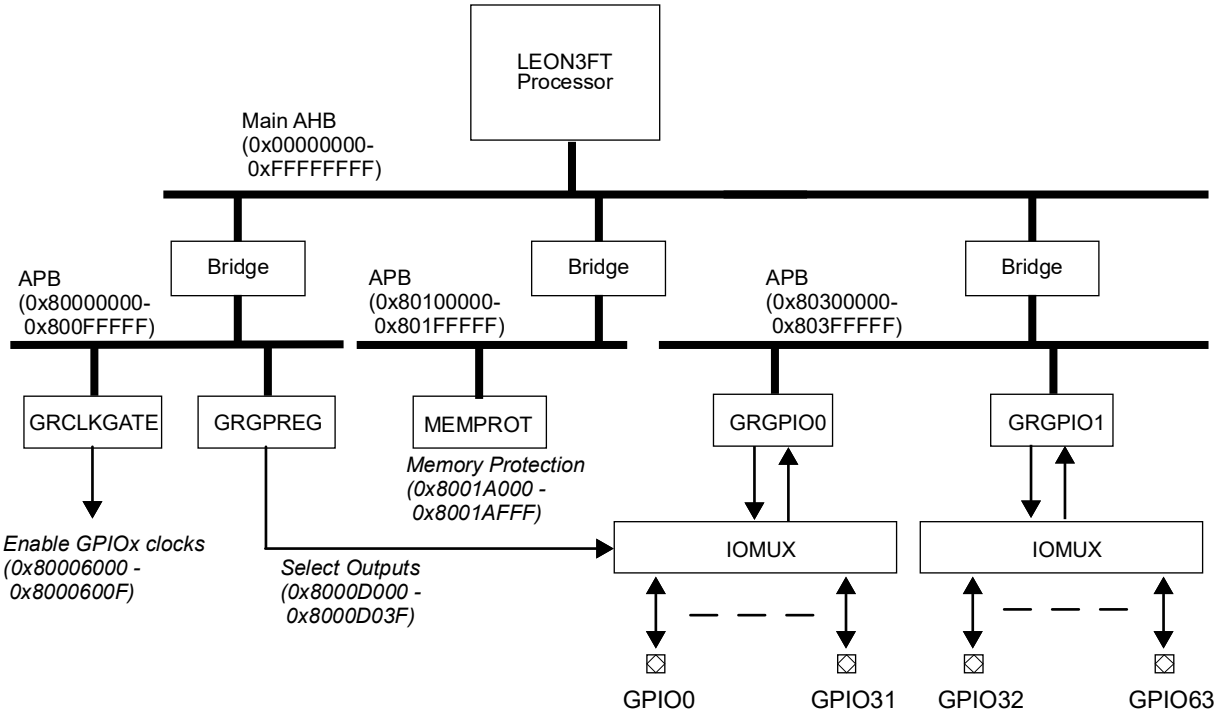


Figure 73. GR716B GRGPIO bus and pin connection

The primary clock gating unit **GRCLKGATE** described in section 27 is used to enable/disable individual General Purpose I/O port (GRGPIO) units. The unit **GRCLKGATE** can also be used to perform reset of individual General Purpose I/O port (GRGPIO) units. Software must enable clock and release reset described in section 27 before General Purpose I/O port (GRGPIO) configuration and transmission can start.

External IO selection per General Purpose I/O port (GRGPIO) unit is made in the system IO configuration register (**GRGPREG**) in the address range from 0x8000D000 to 0x8000D03F. See section 7.1 for further information.

Each **GRGPIOx** unit controls its own external pins and has a unique AMBA address described in chapter 2.10. GRGPIO unit 0 and 1 have identical configuration and status registers. Configuration and status registers are described in section 30.4.

The system can be configured to protect and restrict access to individual General Purpose I/O port (GRGPIO) unit in the **MEMPROT** unit. See section 47 for more information.

30.1 Overview

All 64 external pins can be configured as general purpose I/O. Each external pin in the general purpose mode can be individually set to input or output, and can optionally generate an interrupt. For interrupt generation, the input can be filtered for polarity and level/edge detection.

The GR716B microcontroller comprises two This chapter describes one GPIO unit. The two GPIO units in the GR716B are identical except for the physical external pin connected to GPIO unit 1 and GPIO unit 2. For separation in this document GPIO unit 1 are connected to external pins 0 to 31 . GPIO unit 2 are connected to external pins 32 to 64. Each GPIO unit have a unique AMBA address described in chapter 2.10.

30.2 Operation

All external I/Os have bi-directional buffers with programmable output enable. The input from each buffer is synchronized by two flip-flops in series to remove potential meta-stability. The synchronized values can be read-out from the I/O port data register. The output enable is controlled by the I/O port direction register. A ‘1’ in a bit position will enable the output buffer for the corresponding I/O line. The output value driven is taken from the I/O port output register.

The GPIO interrupts has been implemented to support dynamic mapping of interrupts, each I/O line can be mapped using the Interrupt map register(s) to an interrupt line.

Interrupt generation is controlled by three registers: interrupt mask, polarity and edge registers. To enable an interrupt, the corresponding bit in the interrupt mask register must be set. If the edge register is ‘0’, the interrupt is treated as level sensitive. If the polarity register is ‘0’, the interrupt is active low. If the polarity register is ‘1’, the interrupt is active high. If the edge register is ‘1’, the interrupt is edge-triggered. The polarity register then selects between rising edge (‘1’) or falling edge (‘0’).

The GPIO core includes an Interrupt flag register that can be used to determine if, and which, GPIO pin that caused an interrupt to be asserted.

30.3 Pulse command (TBD)

The pulse command outputs use one of the GR716B microcontroller common counter for establishing the pulse command start and length. The pulse command length defines the logical active part of the pulse. It is possible to select which of the channels shall generate a pulse command. The pulse command outputs are generated in phase with a selected trigger source. To send synchronized pulse commands on multiple outputs simultaneously the same trigger source shall be enabled for the selected outputs.

30.4 Registers

The core is programmed through registers mapped into APB address space.

Table 362. General Purpose I/O Port registers

APB address offset	Register
0x00	I/O port data register
0x04	I/O port output register
0x08	I/O port direction register
0x0C	Interrupt mask register
0x10	Interrupt polarity register
0x14	Interrupt edge register
0x1C	Capability register
0x20 - 0x3C	Interrupt map register(s).
0x40	Interrupt available register
0x44	Interrupt flag register
0x48	Input enable register
0x4C	Pulse register
0x50	Input enable register, logical-OR
0x54	I/O port output register, logical-OR
0x58	I/O port direction register, logical-OR
0x5C	Interrupt mask register, logical-OR
0x60	Input enable register, logical-AND
0x64	I/O port output register, logical-AND
0x68	I/O port direction register, logical-AND
0x6C	Interrupt mask register, logical-AND
0x70	Input enable register, logical-XOR
0x74	I/O port output register, logical-XOR
0x78	I/O port direction register, logical-XOR
0x7C	Interrupt mask register, logical-XOR

30.4.1 I/O Port Data Register

*Table 363.*0x00 - DATA - I/O port data register

31	0
DATA	
*	
r	

31: 0 I/O port input value (DATA) - Data value read from GPIO lines

30.4.2 I/O Port Output Register

*Table 364.*0x04 - OUTPUT - I/O port output register

31	0
DATA	
0	
rw	

31: 0 I/O port output value (DATA) - Output value for GPIO lines

30.4.3 I/O Port Direction Register

*Table 365.*0x08 - DIRECTION - I/O port direction register

31	0
DIR	
0	
rw	

31: 0 I/O port direction value (DIR) - 0=output disabled, 1=output enabled

30.4.4 Interrupt Mask Register

*Table 366.*0x0C - IMASK - Interrupt mask register

31	0
MASK	
0	
rw	

31: 0 Interrupt mask (MASK) - 0=interrupt masked, 1=intrrupt enabled

30.4.5 Interrupt Polarity Register

*Table 367.*0x10 - IPOL - Interrupt polarity register

31	0
POL	
NR	
rw	

31: 0 Interrupt polarity (POL) - 0=low/falling, 1=high/rising

30.4.6 Interrupt Edge Register

*Table 368.*0x14 - IEDGE - Interrupt edge register

31	0
EDGE	
NR	
rw	

31: 0 Interrupt edge (EDGE) - 0=level, 1=edge

30.4.7 Capability Register

*Table 369.*0x1C - CAP - Capability register

31	18	17	16	15	13	12	8	7	5	4	0
RESERVED		PU	IER	IFL	r	IRQGEN		r		NLINES	
0		1	1	1	0	0x4		0		0x1F	
r		r	r	r	r	r		r		r	

31: 19 Reserved and not used

18 PU: Pulse register implemented: If this field is '1' then the core implements the Pulse register.

17 IER: Input Enable register implemented. If this field is '1' then the core implements the Input enable register.

16 IFL: Interrupt flag register implemented. If this field is '1' then the core implements the Interrrupt available and Interrupt flag registers (registers at offsets 0x40 and 0x44).

12 8 IRQGEN: Software can dynamically configure each I/O to drive either of the 4 interrupt lines associated with each GPIO unit (cf. section 2.12).

4: 0 NLINES. Number of pins in GPIO port - 1.

30.4.8 Interrupt Map Register n

*Table 370.*0x20 - 0x3C - IRQMAPRn - Interrupt map register n

31	29	28	24	23	21	20	16	15	13	12	8	7	6	4	0
RESERVED		IRQMAP[4*n]		RESERVED		IRQMAP[4*n+1]		RESERVED		IRQMAP[4*n+2]		RESERVED		IRQMAP[4*n+3]	
0		un+i		0		un+i+1		0		un+i+2		0		uni+i+3	
r		rw		r		rw		r		rw		r		rw	

31: 0 IRQMAP[i] : The field IRQMAP[i] determines to which interrupt I/O line i is connected. If IRQMAP[i] is set to *x*, IO[i] will drive interrupt *pirq+x*. Where *pirq* is the first interrupt assigned to the core (cf. section 2.12). Several I/O can be mapped to the same interrupt.

The core has one IRQMAP field per I/O line. The Interrupt map register at offset 0x20+*4*n* contains the IRQMAP fields for IO[*4*n* : *4*n+3*]. This means that the fields for IO[0:3] are located on offset 0x20, IO[4:7] on offset 0x24, IO[8:11] on offset 0x28, and so on. An I/O line's interrupt generation must be enabled in the Interrupt mask register in order for the I/O line to drive the interrupt specified by the IRQMAP field.

30.4.9 Interrupt Available Register

*Table 371.*0x40 - IAVAIL - Interrupt available register

31	0
IMASK	
*	
r	

31: 0 IMASK: Interrupt mask bit field. If IMASK[n] is 1 then GPIO line n can generate interrupts.

30.4.10 Interrupt Flag Register

Table 372. 0x44 - IFLAG - Interrupt flag register

31	0
IFLAG	
0	
wc	

31: 0 IFLAG : If IFLAG[n] is set to '1' then GPIO line n has generated an interrupt. Write '1' to the corresponding bit to clear. Writes of '0' have no effect.

30.4.11 Input Enable Register

Table 373. 0x48 - IPEN - Input enable register

31	0
IPEN	
0	
rw	

31: 0 IPEN : If IPEN[n] is set to '1' then values from GPIO line n will be visible in the data register. Otherwise the GPIO line input is gated-off to disable input signal propagation.

30.4.12 Pulse Register

Table 374. 0x4C - PULSE - Pulse register

31	0
PULSE	
0	
rw	

31: 0 PULSE : If PULSE[n] is set to '1' then I/O port output register bit n will be inverted whenever selected synchronization source is active. Synchronization source is selected in register SEQSYNC.

30.4.13 Logical-OR/AND/XOR Register

Table 375. 0x50-0x7C - LOR,LAND,LXOR - Logical-OR/AND/XOR registers

31	0
VALUE	
-	
w*	

*Table 375.*0x50-0x7C - LOR,LAND,LXOR - Logical-OR/AND/XOR registers

31:	0	The logical-OR/AND/XOR registers will update the corresponding register according to: New value = <Old value> logical-op <Write data> There exists logical-OR, AND and XOR registers for the Input enable, I/O port output, I/O port direction and Interrupt mask registers. Input enable - Logical OR Address offset 0x50 I/O port output - Logical OR Address offset 0x54 I/O port direction - Logical OR Address offset 0x58 Interrupt mask -Logical OR Address offset 0x5C Input enable - Logical AND Address offset 0x60 I/O port output - Logical AND Address offset 0x64 I/O port direction - Logical AND Address offset 0x68 Interrupt mask -Logical AND Address offset 0x6C Input enable - Logical XOR Address offset 0x70 I/O port output - Logical XOR Address offset 0x74 I/O port direction - Logical XOR Address offset 0x78 Interrupt mask -Logical XOR Address offset 0x7C

30.4.14 Logical-Set&Clear Register

*Table 376.*0x80-0x9C - LSET, LCLEAR - Logical Set & Clear registers

31	0
VALUE	
-	
w*	

31:	0	The logical-Set & Clear registers will update the corresponding register according to: New value = ((<Old value> OR < First write>) AND NOT <Second write>) The first write is used to 'set' bits and second write is used to 'clear' bits. There exists logical-Set & Clear registers for Input enable (Address offset 0x80 - First write, 0x84 - Second write) I/O port output (Address offset 0x88 - First write, 0x8C - Second write) I/O port direction (Address offset 0x90 - First write, 0x94 - Second write) Interrupt mask (Address offset 0x98 - First write, 0x9C - Second write)

31 PacketWire Receiver

Core removed see errata and workaround 57.2.3

32 PacketWire Transmitter

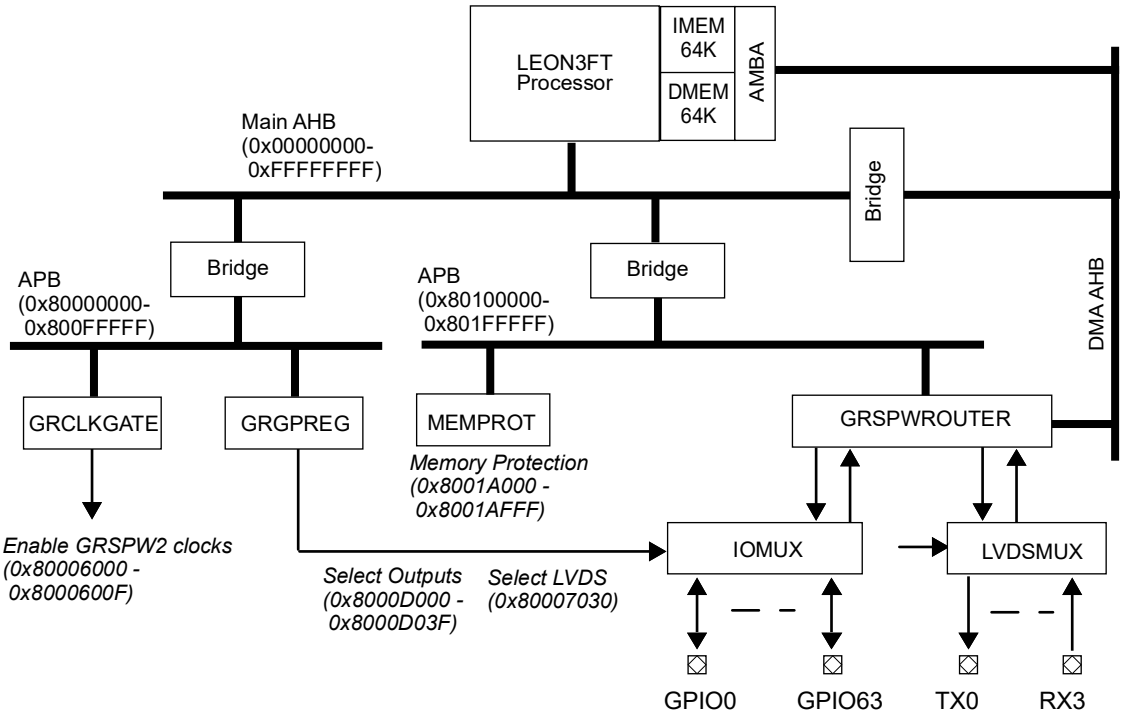
```
Core removed see errata and workaround 57.2.3
```

33 SpaceWire router

The GR716B microcontroller comprises a SpaceWire router with RMAP support (GRSPWROUTER) units. The SpaceWire interface with RMAP controls its own external pins and has a unique AMBA address described in chapter 2.10. The nominal SpaceWire interface is connected via LVDS transceivers to external pins and the redundant interface is connected to external pins via the IOMUX.

The SpaceWire router interface control and status registers are located on APB bus in the address range from 0x80100000 to 0x80100FFF. See GRSPWROUTER unit connections in the next drawing. The figure shows memory locations and functions used for GRSPWROUTER configuration and control.

Figure 74. GR716B 2-port GRSPWROUTER internal bus and external pin connection



The primary clock gating unit **GRCLKGATE** described in section 27 is used to enable/disable the SpaceWire Router interface. The unit **GRCLKGATE** can also be used to perform reset of the SpaceWire router interface Software must enable clock and release reset described in section 27 before configuration and transmission can start.

External IO selection and configuration is made in the system IO and LVDS configuration registers (**GRGPREG**) in the address range from 0x8000D000 to 0x8000D03F and 0x80007030. See section 7.1 for further information.

The system can be configured to protect and restrict access to the SpaceWire router in the **MEMPROT** unit. See section 47 for more information.

33.1 Overview

The SpaceWire router implements a SpaceWire routing switch as defined in the ECSS-E-ST-50-12C standard. The configuration port provides an RMAP target, and an optional AMBA AHB slave interface, both used for accessing internal configuration and status registers.

Among the features supported by the router are: group adaptive routing, packet distribution, system time-distribution, distributed interrupts, port timers to recover from deadlock situations, and SpaceWire-D [SPWD] packet truncation based time-slot violations.

The SpaceWire router in GR716B device implements a 3-port bypass routing mode for support of multiple cascade coupled devices. The 3-port bypass routing mode implements an easy way of filter out packets intended for the GR716B device and to forward all other packets in the hardware. Filtering is enabled by configuring and using logical addresses:

- GR716B router can have one logical address in the range 32 to 223
- Packets received with the logical address of the GR716B device will be routed the AMBA-port
- Packets with logical addressing will be routed according to:
  - Packets incoming on SPW-port 1 and have a different logic address is routed to SPW-port 2
  - Packets incoming on SPW-port 2 and have a different logic address is routed to SPW-port 1
  - Packets incoming on the AMBA-port and have a lower logic address (compared to the configured) is routed to SPW-port 1. A packet with a higher logic address is routed to SPW-port 2

Note: LVDS drivers that are not used in the application can be turned off to save power. This is controlled via the register bank interface. Note that there is no automatic turning off of the LVDS drivers of disabled or inactive SpaceWire links in this device, so this must be managed by the application software.

33.2 Operation

The switch matrix can connect any input port to any output port. Access to each output port is arbitrated using a round-robin arbitration scheme based on the address of the incoming packet. A single routing-table is used for the whole router, where access to the table is arbitrated using a round-robin scheme based on the input port number.

The ports consist of configuration port 0, and two different types of external ports: SpaceWire links and AMBA interfaces. All ports have the same interface to the switch matrix and behave in the same manner. The difference in behavior is on the external side of the port. The SpaceWire ports provide standard SpaceWire link interfaces using on-chip LVDS. The AMBA ports transfer characters from and to an AHB bus using DMA. The different port types are described in further detail in sections 33.3, 33.4 and 33.5.

33.2.1 Port numbering

The router's ports are numbered as follows: configuration port has port number 0, the SpaceWire ports have port numbers 1-2, and the AMBA ports have port numbers 3.

33.2.2 Routing table

A single routing table is provided. The access to this routing table is arbitrated using a round-robin arbiter with each port being of equal priority. The operation is pipelined and one lookup can be done each cycle. This way the maximum latency is equal to the number of ports in the router minus one. The impact on throughput should be negligible provided that packets are not incoming at the same time. The probability for this is higher when the traffic only consist of very small packets sent continuously (the average size being about the same as the number of ports). This should be a very uncommon case. Latency is still bounded and probably negligible in comparison to other latencies in most systems.

Since the latency for the lookup is very small and deterministic there is not much to gain by having configurable priorities for this. Priorities are instead used for arbitrating packets contending for an output port as described in the next section.

The routing table and all the configuration registers are configured through an RMAP target or an AHB slave interface which use the same routing table as the logic handling packet traffic. They do not introduce any extra latency for packets, since the configuration accesses have lower priority than the packet traffic and thus are only allowed access on cycles when no lookup is needed for packets. This can slow down configuration accesses but they are probably mostly done before packet traffic starts and very seldom afterwards. The routing table is split into two parts, one which controls the port mapping for the address (RTR.RTPMAP registers), and one which controls properties for the address, such as priority and header deletion (RTR.RTACTRL registers).

The SpaceWire router has 4 (0 - 3) path address entries, and 224 (32 - 255) logic address entries in the routing table. Path address routes directly to corresponding port, and logic addresses can be used to route packets using single address by configuring the routing table. By default all logic addresses are disabled and sending packets with logical address will result in an individual address error. For the GR716B device only one logical address in the rang 32 to 255 can be configured and used. Hence the routing table for address 32 to 255 has been replaced with a register containing only one mapping (the one to the AMBA port). It shall be noted that all logical entries exists in the memory map to maintain the register compatibility with other SpaceWire router products and software. For detailed information about routing table configuration registers see Table 407 and Table 408.

33.2.2.1 Port mapping

For both physical and logical addresses it is possible to configure which port(s) the incoming packet should be routed to. This is done by programming the corresponding RTR.RTPMAP register. The RTR.RTPMAP registers also controls whether or not group adaptive routing or packet distribution should be used for the incoming packet. The RTR.RTPMAP registers are not initialized after reset / power-up. For physical addresses this has the effect that the incoming packet is routed to the port that matches the address in the packet, without any group adaptive routing or packet distribution. For logical addresses, an uninitialized RTR.RTPMAP register (or if the RTR.RTPMAP.PE field has been written with all zeros) has the effect that the incoming packet is spilled. See table 407 in section 33.6 for more details.

33.2.2.2 Address control

For both physical and logical addresses it is possible to configure the priority, and to enable the spill-if-not-ready feature (explained in section 33.2.8). For logical addresses it is also possible to enable / disable the address, and enable / disable header deletion. Physical addresses are always enabled, and always have header deletion enabled, as specified by ECSS-E-ST-50-12C [SPW]. This configuration for an address is done by programming the corresponding RTR.RTACTRL register. Logical addresses are disabled after reset / power-up. An incoming packet with a disabled logical address is spilled. See table 408 in section 33.6 for details.

33.2.3 Output port arbitration

Each output port is arbitrated individually based on the address of the incoming packet, using two priority levels, with round-robin at each level. Each physical address and logical address can be configured in the routing table (RTR.RTACTRL register) to be either high or low priority. Priority assignments can have large impact on the delays for packets, because packets can be large and the speed of the data consumer and link itself may not be known. This should therefore be considered when assigning priorities.

33.2.4 Group adaptive routing

Group adaptive routing can be used to allow incoming packets to be sent on different output ports depending on which of the output ports that are currently ready. It can be enabled for both physical and logical addresses, and is configured by programming the corresponding RTR.RTPMAP register.

When a packet arrives, and group adaptive routing is enabled for the packet's address, the router looks at the group of ports selected by the corresponding RTR.RTPMAP register and transmits the packet on the port with the lowest index that is currently ready. Ready in this context means that the port's link interface is in run-state and currently not sending any other packet. If none of the selected output ports are ready, the incoming packet will either be spilled or transmitted on the first port that becomes ready. The action taken depends on the setting of the input port's data character timer (see section 33.2.13), the spill-if-not-ready feature for the address (see section 33.2.8), and the link-start-on-request feature for the output ports (see section 33.2.11). See table 407 in section 33.6 for details on how to enable and configure group adaptive routing.

33.2.5 Packet distribution

Packet distribution can be used to implement multicast and broadcast addresses, and can be enabled for both physical and logical addresses. Packet distribution is enabled and configured by programming the corresponding RTR.RTPMAP register.

When a packet arrives, and packet distribution is enabled for the packet's address, the router looks at the group of ports selected by the corresponding RTR.RTPMAP register. If all of the selected ports are ready, the packet is transmitted on all the ports. Ready in this context means that the port's link interface is in run-state and currently not sending any other packet. If one or more of the selected ports are not ready, the incoming packet will either be spilled or transmitted once all ports are ready. The action taken depends on the setting of the input port's data character timer (section 33.2.13), the spill-if-not-ready feature for the address (section 33.2.8), and the link-start-on-request feature for the output ports (section 33.2.11). See table 407 in section 33.6 for details on how to enable and configure packet distribution.

33.2.6 Port disable

A port can be disabled for data traffic by setting the corresponding RTR.PCTRL.DI bit. Incoming packets on a disabled port are silently spilled, and no packets are routed to a disabled port. A disabled port will not be included in any group used for group adaptive routing or packet distribution, even if the corresponding bit in that address' RTR.RTPMAP.PE field is set. When routing packets that are incoming on other ports, the router will simply behave as if the disabled port did not exist. The RTR.PCTRL.DI bit only affects routing of data, thus the transmission and reception of time-codes and distributed interrupt codes are not affected.

The link interface for a SpaceWire port is not affected solely by the RTR.PCTRL.DI bit. However, note that the RTR.PCTRL.DI bit, together with the RTR.PCTRL.LD bit, is used to clock-gate a SpaceWire port (see section).

33.2.7 Static routing

The router supports a feature called static routing, which can be enabled individually per port. When enabled, all incoming packets on the port are routed based on the physical address specified in the port's RTR.PCTRL2.SC field, and the setting of the corresponding RTR.PCTRL2.SC bit, instead of the address in the packets. Header deletion is not used for the incoming packets when static routing is enabled, which means that the first byte of the packets are always sent to the output port as well. Static routing to port 0 is not allowed, and will generate an invalid address error if attempted.

The STATICROUTEEN signal works as a global enable / disable for the static routing feature during reset. The feature is enabled if the signal is high during reset, and disabled if the signal is low during reset. The RTR.RTRCFG.SR bit shows if the static routing feature is globally enabled or disabled.

Note that when static routing is enabled for a port, it is not possible to access the configuration port from the same port.

33.2.8 Spill-if-not-ready

The spill-if-not-ready feature can be enabled individually for each physical and logical address by configuring the corresponding RTR.RTACTRL.SR bit. When enabled, an incoming packet is spilled if the selected output port's link interface is not in run-state. If group adaptive routing is used for the incoming packet then the packet is only spilled if none of the ports in the group is in run-state. If packet distribution is used for the incoming packet then the packet is spilled unless the link interfaces for all selected output ports are in run-state. The spill-if-not-ready feature has priority over the incoming port's data character timer (section 33.2.13) and the output port's link-start-on-request feature (section 33.2.11). This means that if the spill-if-not-ready feature is enabled, the packet is spilled before the timer starts, and the link-start-on-request feature will never be activated.

33.2.9 Self addressing

Self addressing occurs when a selected output port for a packet is the same port as the input port. Whether or not this is allowed is controlled by the RTR.RTRCFG.SA bit. If self addressing is not allowed, the incoming packet is spilled and an invalid address error occurs.

When group adaptive routing is used, and self addressing is not allowed, the input port is still allowed to be in the group of ports configured for the packet. The packet is not spilled until the router actually selects the input port as output port. If the router selects one of the other ports in the group, the packet is not spilled.

When packet distribution is used, and self addressing is not allowed, the input port is not allowed to be in the group of ports configured for the packet, since the packet should be sent to all ports in the group.

33.2.10 Invalid address error

An invalid address error occurs under the conditions listed below.

- When an incoming packet's address corresponds to a non-existing port (physical addresses 20-31).
- When an incoming packet's address is a logical address that is not enabled (RTR.RTACTRL.EN = 0).
- When an incoming packet's address is a logical address for which the corresponding RTR.RTPMAP register is not initialized, or the corresponding RTR.RTPMAP.PE field set to all zeroes.
- When only one output port is selected for an incoming packet, and that port is disabled (RTR.PCTRL.DI = 1).
- When self addressing occurs, and the router is configured not to allow self addressing (RTR.RTRCFG.SA = 0).
- When a packet is routed with the static routing feature, and the physical address programmed in RTR.PCTRL2.SD is 0 (static routing to port 0 is not allowed).

For all the invalid address cases above, the incoming packet is spilled, and the RTR.PSTS.IA bit corresponding to the input port will be set to 1.

33.2.11 Link-start-on-request

The link-start-on-request feature gives the possibility to automatically start a SpaceWire port's link interface when a packet is routed to the port (i.e, port is selected as output port). Each port can have the feature individually enabled by setting the corresponding RTR.PCTRL.LR bit to 1.

If a packet arrives, and the link interface of the selected output port is not in run-state, and the port has the link-start-on-request feature enabled, the router will try to start the link interface under the following conditions:

1. The link interface is not already trying to start (RTR.PCTRL.LS = 0).

2. The link is not disabled (RTR.PCTRL.LD = 0).

3. The spill-if-not-ready feature is not enabled for the packet being routed.

The link will continue to be started until either the RTR.PCTRL.LD bit is set to 1, or until the link is disabled through the auto-disconnect feature, described in section 33.2.12.

The link-start-on-request feature is only available for the SpaceWire ports, since the configuration port and AMBA ports does not have a link interface FSM, and are therefore always considered to be in run-state.

33.2.12 Auto-disconnect

The auto-disconnect feature gives the possibility to automatically disable the link interface of a SpaceWire port if the port has been inactive for a long enough period of time. Each port can have the feature individually enabled by setting their corresponding RTR.PCTRL.AD bit to 1. The amount of time the port needs to be inactive for is decided by the settings of the global prescaler register (RTR.PRESCALER), and the port's individual timer register (RTR.PTIMER). This time period is the same as the timeout period used by the port's data character timer when recovering from deadlock situations (see section 33.2.13). If the auto-disconnect feature is enabled, then a SpaceWire port will automatically disable its link interface under the following conditions:

1. The link interface entered run-state because it was started by the link-start-on-request feature, described in section 33.2.11.

2. The packet that caused the link interface to start has finished (either sent or spilled).

3. Nothing has been transmitted or received on the port for the duration of the time period specified by the RTR.PRESCALER register, and the corresponding RTR.PTIMER register.

4. The port's corresponding RTR.PCTRL.LS bit has not been set to 1.

The auto-disconnect feature is only available for the SpaceWire ports, since the configuration port and AMBA ports does not have a link interface FSM, and are therefore always considered to be in run-state.

33.2.13 Port data character timers

Each port has an individual data character timer, which can be used to timeout an ongoing data transfer in order to recover from a deadlock situation. There are two different timeouts defined: overrun timeout, and underrun timeout. An overrun timeout is when the input port has data available, but the output port(s) can not accept data fast enough. An underrun timeout is when the output port(s) can accept more data, but the input port can not provide data fast enough.

The timeout period for a port is set in its corresponding RTR.PTIMER register, and the timer is enabled through the corresponding RTR.PCTRL.TR bit. Timeouts due to overrun and underrun can also be individually enabled / disabled through the corresponding RTR.PCTRL2.OR and RTR.PCTRL2.UR bits.

It is always the input port's data character timer that is used for timing data transfers. When the timer is enabled, it counts down on every tick from the global prescaler (RTR.PRESCALER register). If a data character is transmitted from the input port to the output port(s), then the timer is restarted. If the timer expires, the ongoing packet is spilled, and an EEP is written to the transmit FIFO of the output port(s).

The range of the timeout period depends on the system clock frequency, and is calculated with the following formula:

<timeout period> = (<clock period> x (RTR.PRESCALER+1)) x RTR.PTIMER

Sub-sections 33.2.13.1 through 33.2.13.4 clarifies the behaviour of the timers for different scenarios that can occur when a packet arrives.

33.2.13.1 Timer disabled

If the data character timer for an input port is disabled, the incoming packet will wait indefinitely for the output port(s) to be ready, unless the spill-if-not-ready feature is enabled for the packet's address (see section 33.2.8).

33.2.13.2 Timer enabled, but output port(s) not in run state

If the spill-if-not-ready feature (see section 33.2.8) is disabled for the incoming packet's address, the input port's data character timer is started when the packet arrives, and if the output port's link interface has not entered run-state when the timer expires, the packet is spilled. When group adaptive routing is used for the incoming packet, it is enough for one of the possible output ports to enter run-state before the timer expires for the packet to be transmitted. If packet distribution is used, all the output ports must enter run-state before the timer expires, otherwise the packet is spilled.

If the link-start-on-request feature is enabled for an output port, that port will try to enter run-state when the packet arrives. However, the input port's timer is unaffected of this, and will still only wait for its configured timeout period, before spilling the packet.

A timeout due to the output port not being in run-state is classified as a overrun timeout, which means that the RTR.PCFG2.OR bit for the input port must be set in order for the packet to be spilled. If the RTR.PCFG2.OR bit is not set, the packet will wait indefinitely (unless spill-if-not-ready is enabled).

33.2.13.3 Timer enabled, output port(s) in run-state but busy with other transmission

The input port's data character timer will not start, and the incoming packet will wait indefinitely until the output port either becomes free or leaves run-state. When group adaptive routing is used for the incoming packet, it is enough for one of the possible output ports to be in run-state to prevent the timer from starting. If packet distribution is used, all the output ports must be in run-state to prevent the timer from starting.

33.2.13.4 Timer functionality when accessing the configuration port

The timer functionality is basically the same for the configuration port as for the other ports. When the command is being received, the configuration port is the output port of the data transfer, and when the reply is being sent, the configuration port is the input port of the data transfer. The differences between the configuration port and the other ports are:

- The configuration port can always accept data fast enough, which means that an overrun timeout will never occur when a command is being received.
- The configuration port can always send data fast enough, which means that an underrun timeout will never occur when a reply is being sent.

33.2.14 Packet length truncation

Packet length truncation monitors the length of an incoming packet, and increases a counter for each received data character. If the counter reaches a value larger than the input port's RTR.MAXPLEN register, and truncation is enabled for the input port (RTR.PCTRL.PL = 1), the rest of the packet is spilled, and an EEP is written to the FIFO of the output port(s). Each port has its own RTR.MAXPLEN register and counter in order to allow different maximum lengths for different ports.

Packet length truncation can also be enabled for port 0. In that case, it is the length of the RMAP / SpaceWire Plug-and-Play reply packet that is monitored.

33.2.15 System time-distribution

The router supports system time distribution through time-codes, as defined in ECSS-E-ST-50-12C [SPW]. It contains a global time-counter register (RTR.TC) where the latest received time-code can be read. Both the SpaceWire ports and the SIST port support time-code transmission and reception.

All incoming time-codes update the RTR.TC register. If the incoming time-code has a time value which is plus one (modulo 64) compared to the old RTR.TC value, the time-code is forwarded to all the other ports. The time-code is not sent out onto the port on which it arrived.

Time-codes can be globally enabled / disabled through the RTR.TC register, as well as individually enabled / disabled per port through respective RTR.PCTRL.TE bit. When time-codes are disabled for a port, all incoming time-codes on that port are discarded, and no time-codes will be forwarded to the port.

The router can be configured to either filter out all incoming time-codes that does not have the two control flags (bit 7:6) set to “00”, or to discard the control flags and allow them to have any value. This configuration is done through the RTR.RTRCFG.TF bit. The value of the control flags for the last received time-code can also be read from the RTR.TC register. Note that if interrupt distribution is globally enabled (RTR.RTRCTRL.IE = 1), only control flags “00” are considered as time-codes, no matter the value of the RTR.RTRCFG.TF bit.

33.2.16 SpaceWire distributed interrupt support

The router supports SpaceWire distributed interrupts. It can be configured to operate in two modes, interrupt with acknowledgment mode and extended interrupt mode. In the interrupt with acknowledgment mode, 32 interrupt numbers are supported, whereas the extended interrupt mode supports 64 interrupt numbers. The operation mode is configured through the RTR.RTRCFG.EE bit.

A distributed interrupt code is a control code that has the control flags (bits 7:6) always set to “10”.

- *Interrupt with acknowledgment mode:* A distributed interrupt code that has bit 5 set to 0 is called an interrupt code, and bits 4:0 specify an interrupt number between 0 and 31. When operating in the interrupt with acknowledgment mode, a distributed interrupt code with bit 5 set to 1 is called an interrupt acknowledgment code, which is used to acknowledge the interrupt with the interrupt number specified by bits 4:0.
- *Extended interrupt mode:* When operating in the extended interrupt mode, a distributed interrupt code with bit 5 set to 1 is called an extended interrupt code, and bits 4:0 specify an interrupt number between 32 and 63. If bit 5 is set to 0, bits 4:0 specify an interrupt number between 0 and 31.

The interrupt codes and extended interrupt codes are generated by the source of the interrupt event, while the interrupt acknowledgment code is sent by the interrupt handler for the corresponding interrupt number.

The router has two 32-bit ISR register (RTR.ISR0 and RTR.ISR1) where each bit corresponds to one interrupt number. A bit in the ISR registers is set to 1 when an interrupt code or extended interrupt code with the corresponding interrupt number is received. A bit in the ISR registers is set to 0 when an interrupt acknowledgment code with the corresponding interrupt number is received. Therefore, the ISR registers reflect the status of all interrupt numbers. Each interrupt number has also its own timer which is used to clear the ISR register bit if an interrupt acknowledgment code is not received before the timer expires (for example if operating in the extended interrupt mode), as well as an optional timer which is used to control how fast a bit in the RTR.ISR register is allowed to toggle. See section 33.2.16.2 for more details on the ISR timers. Note that although it is possible to clear the bits in the ISR registers, these registers should normally only be used for diagnostics and FDIR.

33.2.16.1 Receiving and transmitting distributed interrupt codes

When a distributed interrupt code is received on a port, or the auxiliary time-code / distributed interrupt code interface, the following requirements must be fulfilled in order for the code to be distributed:

1. Interrupt distribution is globally enabled (RTR.RTRCTRL.IE = 1), and enabled for the port that received the code (corresponding RTR.PCTRL.IC = 1).

2. If the received code is an interrupt code, the RTR.PCTRL2.IR bit for the port must be set to 1. If the received code is an interrupt acknowledgement code or extended interrupt code, the RTR.PCTRL2.AR bit for the port must be set to 1.

3. If the code is an interrupt code or extended interrupt code, the interrupt number's corresponding bit in RTR.ISR0 or RTR.ISR1 must be 0. If the code is an interrupt acknowledgement code, the corresponding bit in RTR.ISR0 must be 1.

4. No previous distributed interrupt code with the same interrupt number is waiting to be distributed.

5. The ISR change timers (see section 33.2.16.2) are either globally disabled (RTR.RTRCFG.IC = 0) or the interrupt number's corresponding ISR change timer has expired.

If one of the requirements above is not fulfilled, then the received code is discarded. If all of the requirements above are fulfilled, then the received code is placed in a queue. The queue is then serviced in one of the four following ways, depending on the settings of the RTR.RTRCFG.IS and RTR.RTRCFG.IP bits:

1. All interrupt codes have priority over all interrupt acknowledgement codes / extended interrupt codes (RTR.RTRCFG.IP = 0), and the interrupt numbers are serviced through a round-robin scheme (RTR.RTRCFG.IS = 0). This is the default service scheme after reset / power-up.

2. All interrupt codes have priority over all interrupt acknowledgement codes / extended interrupt codes (RTR.RTRCFG.IP = 0), and the interrupt numbers are serviced with priority to lower interrupt numbers (RTR.RTRCFG.IS = 1).

3. All interrupt acknowledgement codes / extended interrupt codes have priority over all interrupt codes (RTR.RTRCFG.IP = 1), and the interrupt numbers are serviced through a round-robin scheme (RTR.RTRCFG.IS = 0).

4. All interrupt acknowledgement codes / extended interrupt codes have priority over all interrupt codes (RTR.RTRCFG.IP = 1), and the interrupt numbers are serviced with priority to lower interrupt numbers (RTR.RTRCFG.IS = 1).

When a distributed interrupt code has been selected from the queue, it is forwarded to all ports (except the port it was received on) that has interrupt distribution enabled (RTR.PCTRL.IC = 1), and that has enabled transmission of interrupt codes or interrupt acknowledgement codes / extended interrupt codes (RTR.PCTRL2.IT and RTR.PCTRL2.AT respectively).

33.2.16.2 Interrupt distribution timers

Each interrupt number has two corresponding timers, called the ISR timer, and ISR change timer:

The ISR timer is started and reloaded with the value from the RTR.ISRTIMER register each time a received interrupt code / extended interrupt code sets the corresponding RTR.ISR0 / RTR.ISR1 bit to 1. If an interrupt acknowledgement code is received, the corresponding ISR timer is stopped. If the ISR timer expires before an interrupt acknowledgement code is received, the corresponding bit in the RTR.ISR0 or RTR.ISR1 register is cleared. The use of ISR timers is always enabled. In the interrupt with acknowledgement mode, the purpose of the timers is to recover from situations where an interrupt acknowledgement code is lost. In the extended interrupt mode, the purpose of the ISR timers is to limit the rate of which interrupt codes are forwarded. It is important to configure the reload value for the ISR timer correctly. In the interrupt with acknowledgement mode, the reload value must not be less than the worst propagation delay for the interrupt code, plus the maximum delay in the interrupt handler, plus the worst propagation delay for the interrupt acknowledgement code. In the extended interrupt mode, the reload value must not be less than the worst propagation delay for the interrupt code / extended interrupt code.

The ISR change timers are timers that optionally can be used to control the minimum delay between two consecutive changes to the same RTR.ISR0 / RTR.ISR1 bit. The purpose of the timers is to protect against unexpected code occurrences that could occur, for example, due to a network malfunction or a babbling idiot. If the use of ISR change timers is enabled (RTR.RTRCFG.IC = 1), then the ISR

change timer for an RTR.ISR0 / RTR.ISR1 bit is started and reloaded with the value from the RTR.ISRCTIMER register each time a received distributed interrupt code makes the RTR.ISR0 / RTR.ISR1 bit change value. Until the timer has expired, the corresponding RTR.ISR0 / RTR.ISR1 bit is not allowed to change value, and any received distributed interrupt code with that interrupt number is discarded. In the extended interrupt mode, the ISR change timers are not used, and should be disabled.

33.2.16.3 Interrupt code generation

In addition to distributing interrupt codes received on the ports, the router can also generate an interrupt code / extended interrupt code when an internal error event occurs, such that the IRQ pin is set to 1. See section 33.2.20 for information about how to control which errors that set the IRQ pin. In addition to the errors described in 33.2.20 the IRQ pin can also be configured to be set when a SpaceWire port's link interface enters run-state.

Everything in sections 33.2.16.1 and 33.2.16.2 also applies when the distributed interrupt code is generated by the router. The only difference is that a distributed interrupt code generated by the router will not be discarded if it is not allowed to be distributed. Instead, the distributed interrupt code will be distributed later, as soon as it is allowed. The only time a distributed interrupt code generated by the router is not distributed is if the bits in RTR.PIP are cleared by software before the interrupt code is allowed to be sent.

The interrupt code generation is controlled through the RTR.ICODEGEN register, and in addition to the enable / disable bit (RTR.ICODEGEN.EN), the following features are available:

- The interrupt number to use for a generated distributed interrupt code is programmable through the RTR.ICODEGEN.IN field.
- The generated distributed interrupt code can be configured to be either level type, or edge type. Level type means that a new distributed interrupt code will be sent as long as the IRQ pin is set (i.e. as long as any bit in the RTR.PIP register is set). Edge type means that a new distributed interrupt code will only be sent when a new error event occurs (an RTR.PIP bit toggles from 0 to 1). The type is selected by the RTR.ICODEGEN.IT bit.
- A timer can be enabled through the RTR.ICODEGEN.TE bit. This timer controls the minimum time between a received interrupt acknowledgement code and the distribution of a new generated interrupt code. The timer is started and reloaded with the value from the RTR.AITIMER register when an interrupt acknowledgement code is received, if the router was the source of the corresponding interrupt code. Until the timer has expired, a new generated interrupt code will not be distributed. The reload value should not be less than the worst propagation delay for the interrupt acknowledgement code. This timer is unused when operating in the extended interrupt mode.
- Through the RTR.ICODEGEN.AH and RTR.ICODEGEN.UA bits the router can be configured to, upon receiving an interrupt acknowledgement code, or the when the ISR timer expires, automatically clear the RTR.PIP bits that were set when the distributed interrupt code was generated.

33.2.17 Auxiliary time-code / distributed interrupt code interface

The router provides an auxiliary time-code / distributed interrupt code interface that is connected internally to the SpaceWire TDP controller.

The rules that determine whether a distributed interrupt code should be forwarded to the ports are the same as when a distributed interrupt code is received from any other port. However, for time-codes, the value on AUXTIMEIN[7:0] is always forwarded, and the router's internal time-count value is updated. Note that time-codes / distributed interrupt codes received through the auxiliary interface is not forwarded back out onto the interface itself.

Just as for the router ports, the auxiliary interface has enable / disable bits for time-codes (RTR.RTRCFG.AT) and distributed interrupt codes (RTR.RTRCFG.AI), which need to be set high in order for respective code to be transmitted / received.

33.2.18 SpaceWire-D support

33.2.18.1 Time-code / distributed interrupt code truncation

The router supports truncation of packets when it receives a valid time-code / distributed interrupt code (time-code or distributed interrupt code). A time-code is considered valid when the value equals the internal time count plus one (modulo 64). An distributed interrupt code is considered valid if the corresponding ISR bit is flipped due to reception of the code. The feature can be enabled individually for each port by setting the corresponding RTR.PCTRL.TS bit to 1. A filter, allowing only certain time-codes / distributed interrupt codes to spill packets, can also be configured individually for each port (see RTR.PCTRL2 register).

If a packet transfer is ongoing when a valid time-code / distributed interrupt code is received, and the code matches the filter in RTR.PCTRL2, the rest of the packet is spilled, and an EEP is written to the FIFO of the output port(s).

Time-code / distributed interrupt code truncation can also be enabled for port 0. In that case, it is the RMAP / SpaceWire Plug-and-Play reply packet that is spilled.

33.2.19 Character and packet counters

Each port, except port 0, has counters for incoming and outgoing characters and packets. For the character counters (RTR.ICHARCNT / RTR.OCHARCNT registers), only SpaceWire data characters are counted (not EOP/EEP). Characters deleted due to header deletion are counted on the incoming port but not on the outgoing port. For the packet counters (RTR.IPKTCNT / RTR.OPKTCNT registers), each EOP/EEP that is preceded by at least one data character is counted as a packet. The counters wrap around, and signal an overflow, when they reach the maximum value. The counters are accessed through the configuration port. See section 33.6 for details.

33.2.20 Error detection and reporting

The router can detect and report the following errors:

- SpaceWire link errors: parity error, disconnect error, escape error, credit error (see ECSS-E-ST-50-12C [SPW] for definition of these errors).
- Packet spill due to: timeout (see section 33.2.13), packet length truncation (see section 33.2.14), time-code / distributed interrupt code truncation (see section 33.2.18.1), and spill-if-not-ready feature (see section 33.2.8)
- Invalid address error (see section 33.2.10)
- RMAP errors (see section 33.5.1)
- Memory errors in the memory used for the routing table, RMAP command buffers, and port transmit and receive FIFO (see section 33.2.22)
- SpaceWire Plug-and-Play errors (see section 33.6.1)

Each error type has corresponding status bits in respective RTR.PSTS register (RTR.PSTSCFG for the configuration port). Common for all the status bits is that they are set when the error is detected, and stay set until they are cleared manually.

Another way the router can report an error is to asserted the IRQ pin. Whether or not the IRQ pin is set when one of the above mentioned errors occur is controlled by the two mask registers, RTR.IMASK and RTR.IPMASK. When the error occur and both the port's corresponding bit in RTR.IPMASK as well as the error type's corresponding bit in RTR.IMASK, are set, then the port's corresponding bit in the Port interrupt pending register (RTR.PIP) is set. The IRQ pin is high as long as any bit in the RTR.PIP register is set.

The router can also be configured to generate a distributed interrupt code when the IRQ pin gets set. See section 33.2.16 for more details.

33.2.21 Setting link-rate for the SpaceWire ports

The initialization divisor register (RTR.IDIV) determines the link-rate during initialization (all states up to and including the connecting-state) for all SpaceWire ports. The register is also used to calculate the link interface FSM timeouts for all SpaceWire ports (6.4 us and 12.8 us, as defined in the SpaceWire standard). The RTR.IDIV register should always be set so that a 10 Mbit/s link-rate is achieved during initialization. In that case the timeout values will also be calculated correctly. The reset value of the RTR.IDIV.ID field is taken from the IDIVISOR[7:0] signal, see Table 419 for details.

To achieve a 10 Mbit/s link-rate, the RTR.IDIV register should be set according to the following formula:

RTR.IDIV = (<frequency in MHz of internal SpaceWire clock> / 10) - 1

The link-rate in run-state can be controlled individually per SpaceWire port with the run-state divisor located in each port's control register (RTR.PCTRL.RD field). The link-rate in run-state is calculated according to the following formula:

<link-rate in Mbits/s> = <frequency in MHz of internal SpaceWire clock> / (RTR.PCTRL.RD+1)

The value in RTR.PCTRL.RD only affects the link-rate in run-state, and does not affect the 6.4 us or 12.8 us timeouts values. The reset value of the RTR.PCTRL.RD field is taken from the IDIVISOR[7:0] signal, see Table 410 for details.

33.2.22 On-chip memories

There are two memory blocks in the routing table, one for the port setup registers and one for the routing table. The port setup memory bit width is equal to the number of ports including the configuration port with depth 256. The routing table is 256 locations deep and 3 bits wide.

Each port excluding the configuration port also have FIFO memories. The SpaceWire ports have one FIFO per direction (rx, tx) which are 10-bit respective 9-bit wide. The FIFO ports have the exact same FIFO configuration as the SpaceWire ports.

The AMBA ports have one 9-bit wide receiver FIFO and two 32-bit wide AHB FIFOs.

Parity is used to protect the memories and up to four bits per word can be corrected and there is a signal indicating an uncorrectable error.

If a memory error occurs in the port setup table or the routing table the memory error (ME) bit in the router configuration/status register is set and remains set until cleared by the user. If a memory error is detected in any of the ports FIFO memories the memory error (ME) bit in the respective port status register is set and remains set until cleared by the user. The ME bits are only set for uncorrectable errors.

When an uncorrectable error is detected in the port setup or routing table when a packet is being routed it will be discarded. Unocrrectable errors in the FIFO memories are not handled since they only affect the contents of the routed packet not the operation of the router itself. These type of errors should be caught by CRC checks if used in the packet.

The ME bit for the ports is only usable for detecting errors and statistics since there is no need to correct the error manually since the packet has already been routed when it is detected. The ME indication for the routing table and port setup registers can be used for starting a scrubbing operation if detected.

33.3 SpaceWire ports

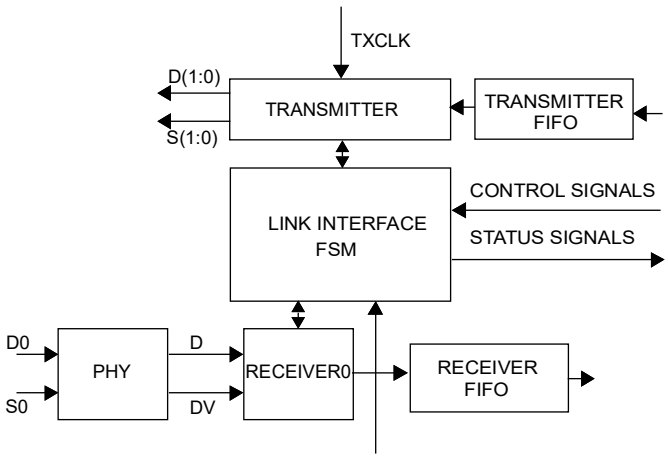


Figure 75. Block diagram

33.3.1 Each SpaceWire port comprises a SpaceWire codec that implements an encoder-decoder compliant to ECSS-E-ST-50-12C [SPW]. The interface to the router's switch matrix consists of a transmit FIFO, receive FIFO, and control and status signals, see Figure 75. The transmit and receive FIFOs can both store up to 64 N-chars. All the configuration parameters and status information for the ports are accessible through the router's configuration port, either through RMAP or AMBA AHB (see section 33.6).

33.3.2 Link-interface FSM

The link-interface FSM controls the link interface (a more detailed description is found in ECSS-E-ST-50-12C [SPW]). The low-level protocol handling (the signal and character level) is handled by the transmitter and receiver while the FSM handles the exchange level.

The link-interface FSM is controlled through the control signals provided in the RTR.PCTRL registers. The link can be disabled through the RTR.PCTRL.LD bit, which depending on the current state, either prevents the link-interface from reaching the started-state, or forces it to the error-reset state. When the link is not disabled, the link interface FSM is allowed to enter the started-state when either the RTR.PCTRL.LS bit is set, or the link-start-on-request feature described in section 33.2.11 is trying to start the port, or when a NULL character has been received and the RTR.PCTRL.AS bit is set.

The current state of the link-interface determines which type of characters are allowed to be transmitted, which, together with the requests made from the host interface, determine what character will be sent.

When the link-interface is in the connecting- or run-state it is allowed to send FCTs. FCTs are sent automatically by the link-interface when possible. This is done based on the maximum value of 56 for the outstanding credit counter and the currently free space in the receive FIFO. FCTs are sent as long as the outstanding counter is less than or equal to 48, and there are at least 8 more empty FIFO entries than the counter value.

N-Chars are sent in the run-state when they are available from the transmit FIFO, and there are credits available. NULLs are sent when no other character transmission is requested, or when the FSM is in a state where no other transmission is allowed.

The credit counter (incoming credits) is automatically increased when FCTs are received, and decreased when N-Chars are transmitted. The credit counter for a SpaceWire port can be read in the corresponding RTR.CREDCNT register.

33.3.3 Transmitter

The state of the FSM, credit counters, possible request to send a time-code / distributed interrupt code, and requests from the transmit FIFO are used to decide the next character to be transmitted. The type of character and the character itself (for N-Chars and time-codes / distributed interrupt codes) to be transmitted are presented to the low-level transmitter, which run on the internal SpaceWire clock. For information on how to change the transmission rate, please see section 33.2.21.

The state of the FSM, credit counters, requests from the time-interface and requests from the transmitter FIFO are used to decide the next character to be transmitted. The type of character and the character itself (for N-Chars and Time-codes) to be transmitted are presented to the low-level transmitter which is located in a separate clock-domain.

33.3.4 Receiver

The receiver is activated as soon as the link-interface leaves the error reset state. Then after a NULL is received it can start receiving any characters. It detects parity, escape and credit errors, which causes the link interface to enter the error-reset state.

Received L-Chars are the handled automatically by link-interface, while all N-Chars are stored in the receive FIFO.

The max receive rate is 1.3 times the frequency of the internal SpaceWire clock and the device has been tested for a receive bit rate of maximum 200 Mbit/s. The system clock must be higher or equal to the receive bit rate divided by eight. For example, if the receive bit rate is 100 Mbit/s then the system clock must be at least 12.5 MHz.

33.3.5 Setting link-rate

The router's Initialization divisor register determines the link-rate during initialization (all states up to and including the connecting-state). The register is also used to calculate the link interface FSM timeouts (6.4 us and 12.8 us, as defined in the SpaceWire standard). The register's ID field should always be set so that a 10 Mbit/s link-rate is achieved during initialization. In that case the timeout values will also be calculated correctly.

To achieve a 10 Mbit/s link-rate, the ID field should be set according to the following formulas:

With single data rate (SDR) outputs:

ID = (<frequency in MHz of internal SpaceWire clock> / 10) - 1

The link-rate in run-state is controlled with the run-state divisor, which is set through the RD field of each port's Port control register. The link-rate in run-state is calculated according to the following formulas:

With SDR outputs:

<link-rate in Mbits/s> = <frequency in MHz of internal SpaceWire clock> / (RD+1)

The value of the RD field only affects the link-rate in run-state, and does not affect the 6.4 us or 12.8 us timeouts values.

An example of clock divisor and resulting link-rate, with a internal SpaceWire frequency of 50 MHz, is shown in the table 377.

Table 377. SpaceWire link-rate example with 50 MHz TXCLK

	Link-rate in Mbit/s
Clock divisor value	**SDR outputs**
0	50
1	25
3	12.5
4	10
9	5

33.4 AMBA ports

The AMBA ports are Frontgrade Gaisler's GRSPW2 controller with the SpaceWire codec removed. Thus, the same drivers that are provided for the GRSPW2 can also be used for an AMBA port of the router. Only one additional driver is needed, which handles the setup of the registers within the configuration port.

33.4.1 Overview

The router's AMBA ports are configured by register accesses through an APB interface. Data is transferred through one to four DMA channels using an AHB master interface.

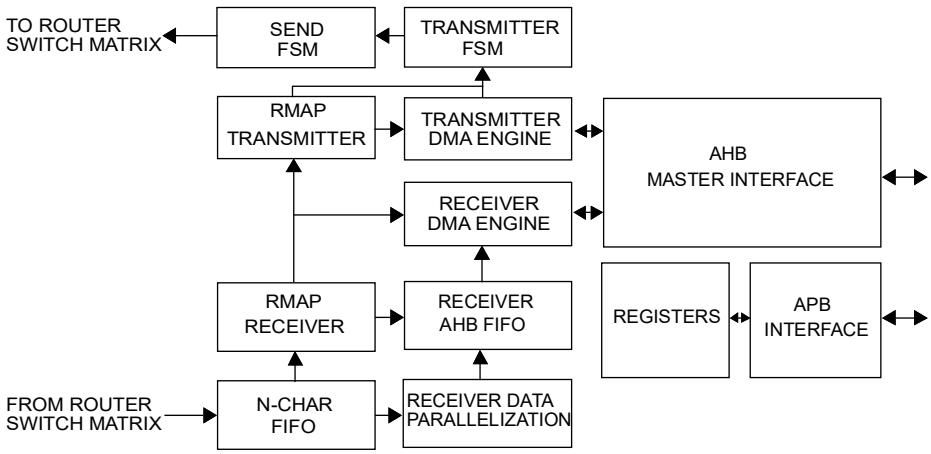


Figure 76. Block diagram of the Router DMA port

33.4.2 Operation

The main sub-blocks of the router AHB interfaces are the DMA engines, the RMAP target and the AMBA interface. A block diagram of the internal structure can be found in figure 76.

The AMBA interface is divided into the AHB master interface and the APB interface. The DMA engines have FIFO interfaces to the router switch matrix. These FIFOs are used to transfer N-Chars between the AMBA bus and the other ports in the router.

The RMAP target handles incoming packets which are determined to be RMAP commands instead of the receiver DMA engine. The RMAP command is decoded and if it is valid, the operation is per-

formed on the AHB bus. If a reply was requested it is automatically transmitted back to the source by the RMAP transmitter.

The AMBA ports are controlled by writing to a set of user registers through the APB interface and a set of signals. The different sub-modules are discussed in further detail in later sections.

33.4.2.1 Protocol support

The AMBA port only accepts packets with a valid destination address in the first received byte. Packets with address mismatch will be silently discarded (except in promiscuous mode which is covered in section 33.4.4.10).

The second byte is sometimes interpreted as a protocol ID a described hereafter. The RMAP protocol (ID=0x1) is the only protocol handled separately in hardware while other packets are stored to a DMA channel. If the RMAP target is present and enabled all RMAP commands will be processed, executed and replied automatically in hardware. Otherwise RMAP commands are stored to a DMA channel in the same way as other packets. RMAP replies are always stored to a DMA channel. More information on the RMAP protocol support is found in section 33.4.6 (note that this RMAP target is different from the one in the configuration port). When the RMAP target is not present or disabled, there is no need to include a protocol ID in the packets and the data can start immediately after the address.

All packets arriving with the extended protocol ID (0x00) are stored to a DMA channel. This means that the hardware RMAP target will not work if the incoming RMAP packets use the extended protocol ID. Note also that packets with the reserved extended protocol identifier (ID = 0x000000) are not ignored by the AMBA port. It is up to the client receiving the packets to ignore them.

When transmitting packets, the address and protocol-ID fields must be included in the buffers from where data is fetched. They are *not* automatically added by the AMBA port DMA engine.

Figure 77 shows the packet types accepted by the port. The port also allows reception and transmission with extended protocol identifiers but without support for RMAP CRC calculations and the RMAP target.

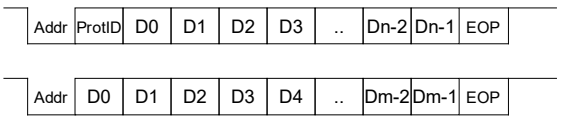


Figure 77. The SpaceWire packet types supported by the port.

33.4.3 Time-Code / distributed interrupt interface

33.4.3.1 Sending and receiving Time-Codes

To transmit a Time-Code through the register interface of an AMBA port, the RTR.AMBACTRL.TI bit should be written to 1. When the bit is written, the AMBA port's current time value (RTR.AMBATC.TIMECNT field) is incremented and a Time-Code consisting of the new time value together with the current control flags (RTR.AMBATC.TCTRL field) is sent. The RTR.AMBACTRL.TI bit will stay high until the Time-Code has been transmitted. Note that whether or not a Time-Code is forwarded to any other port after it has been sent by an AMBA port depends on the settings explained in 33.2.15.

A Time-Code that is received by an AMBA port will be stored in the port's RTR.AMBATC register. There is also a possibility to generate AMBA interrupts and tick-out pulses. This is controlled through the RTR.AMBACTRL and RTR.AMBATC registers. See section 33.4.8 for details about these registers.

33.4.3.2 Sending and receiving distributed interrupts

To transmit a distributed interrupt code through the register interface of an AMBA port, the RTR.AMBAINTCTRL.II bit in should be written to 1. When the bit is written, the value of the RTR.AMBAINTCTRL.TXINT field determine which distributed interrupt code that will be sent. Note that whether or not a distributed interrupt code is forwarded to any other port after it has been sent by an AMBA port depends on the settings of the router and the state of that specific interrupt in the network. See 33.2.16 for details.

A distributed interrupt can also be configured to be generated automatically by the AMBA port upon certain events. This is controlled through the RTR.AMBAINTCTRL and RTR.AMBADMACTRL registers. See section 33.4.8 for details about these registers and features.

Distributed interrupt codes that are received by an AMBA port can be programmed to generate AMBA interrupts as well as tick out-pulses. See section 33.4.8 for details about these features.

33.4.3.3 Sending and receiving Time-Codes and distributed interrupts via signal interface

For all received control codes, Time-Codes or not, the control flags together with the time value are outputted on the TIMEOUT[7:0] signals, and the TICKOUT signal is asserted for one system clock cycle.

Control codes can be sent by asserting the TICKIN signal and place the value of the Time-Code / distributed interrupt to be sent on the TIMEIN[7:0] signals.

33.4.4 Receiver DMA channels

The receiver DMA engine handles reception of data from the SpaceWire network to different DMA channels.

33.4.4.1 Address comparison and channel selection

Packets are received to different channels based on the address and whether a channel is enabled or not. When the receiver N-Char FIFO contains one or more characters, N-Chars are read by the receiver DMA engine. The first character is interpreted as the logical address and is compared with the addresses of each channel starting from 0. The packet will be stored to the first channel with an matching address. The complete packet including address and protocol ID but excluding EOP/EEP is stored to the memory address pointed to by the descriptors (explained later in this section) of the channel.

Each SpaceWire address register has a corresponding mask register. Only bits at an index containing a zero in the corresponding mask register are compared. This way a DMA channel can accept a range of addresses. There is a default address register which is used for address checking in all implemented DMA channels that do not have separate addressing enabled and for RMAP commands in the RMAP target. With separate addressing enabled the DMA channels' own address/mask register pair is used instead.

If an RMAP command is received it is only handled by the target if the default address register (including mask) matches the received address. Otherwise the packet will be stored to a DMA channel if one or more of them has a matching address. If the address does not match neither the default address nor one of the DMA channels' separate register, the packet is still handled by the RMAP target if enabled since it has to return the invalid address error code. The packet is only discarded (up to and including the next EOP/EEP) if an address match cannot be found and the RMAP target is disabled.

Packets, other than RMAP commands, that do not match neither the default address register nor the DMA channels' address register will be discarded. Figure 78 shows a flowchart of packet reception.

At least 2 non EOP/EEP N-Chars needs to be received for a packet to be stored to the DMA channel unless the promiscuous mode is enabled in which case 1 N-Char is enough. If it is an RMAP packet

with hardware RMAP enabled 3 N-Chars are needed since the command byte determines where the packet is processed. Packets smaller than these sizes are discarded.

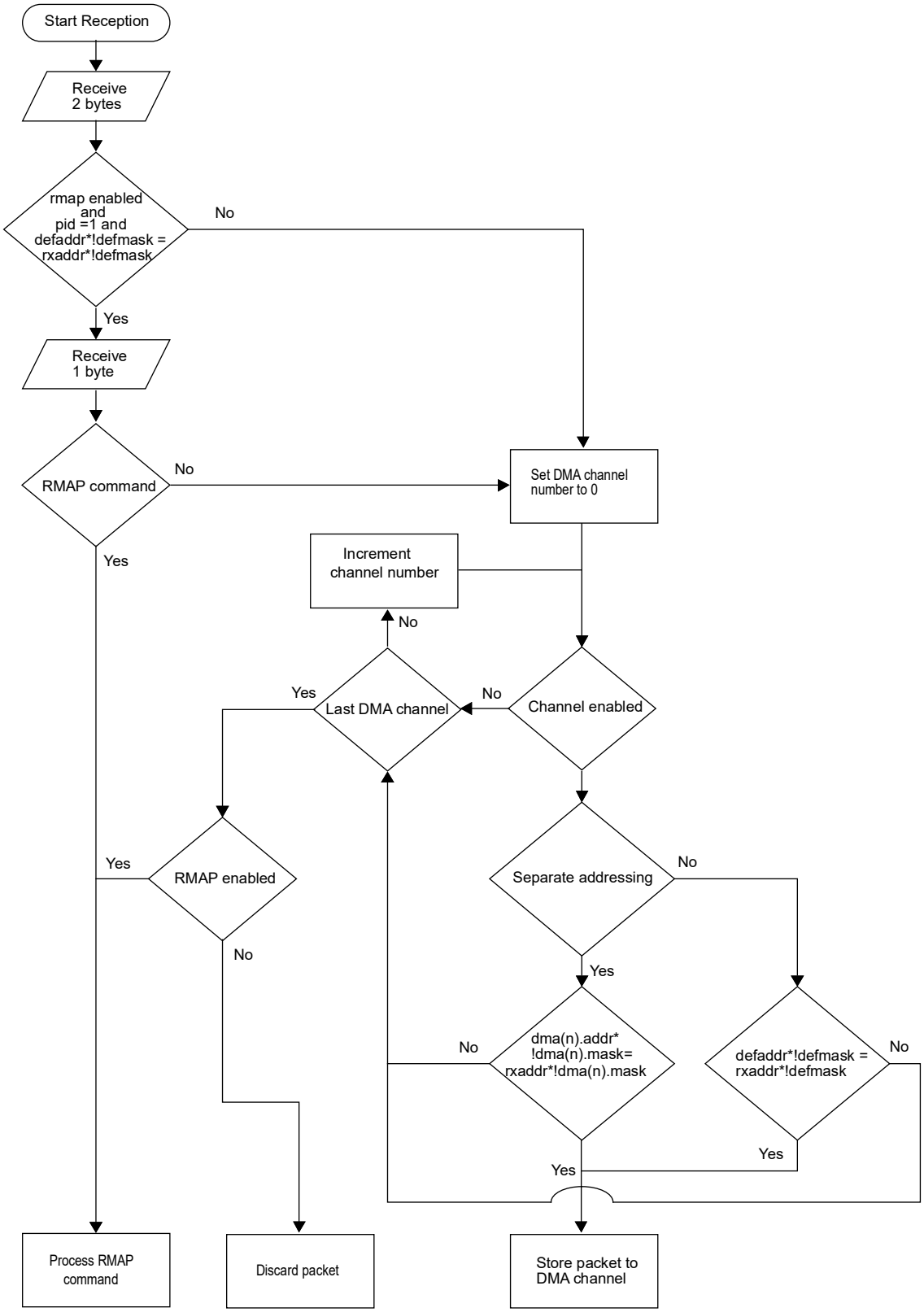


Figure 78. Flow chart of packet reception (promiscuous mode disabled).

33.4.4.2 Basic functionality of a channel

Reception is based on descriptors located in a consecutive area in memory that hold pointers to buffers where packets should be stored. When a packet arrives at the port the channel which should receive it is first determined as described in the previous section. A descriptor is then read from the channels' descriptor area and the packet is stored to the memory area pointed to by the descriptor. Lastly, status is stored to the same descriptor and increments the descriptor pointer to the next one. The following sections will describe DMA channel reception in more detail.

33.4.4.3 Setting up the port for reception

A few registers need to be initialized before reception to a channel can take place. The DMA channel has a maximum length register which sets the maximum packet size in bytes that can be received to this channel. Larger packets are truncated and the excessive part is spilled. If this happens an indication will be given in the status field of the descriptor. The minimum value for the receiver maximum length field is 4 and the value can only be incremented in steps of four bytes up to the maximum value 33554428. If the maximum length is set to zero the receiver will *not* function correctly.

Either the default address register or the channel specific address register (the accompanying mask register must also be set) needs to be set to hold the address used by the channel. A control bit in the DMA channel control register determines whether the channel should use default address and mask registers for address comparison or the channel's own registers. Using the default register the same address range is accepted as for other channels with default addressing and the RMAP target while the separate address provides the channel its own range. If all channels use the default registers they will accept the same address range and the enabled channel with the lowest number will receive the packet.

Finally, the descriptor table and control register must be initialized. This will be described in the two following sections.

33.4.4.4 Setting up the descriptor table address

The port reads descriptors from an area in memory pointed to by the receiver descriptor table address register. The register consists of a base address and a descriptor selector. The base address points to the beginning of the area and must start on a 1024 bytes aligned address. It is also limited to be 1024 bytes in size which means the maximum number of descriptors is 128 since the descriptor size is 8 bytes.

The descriptor selector points to individual descriptors and is increased by 1 when a descriptor has been used. When the selector reaches the upper limit of the area it wraps to the beginning automatically. It can also be set to wrap at a specific descriptor before the upper limit by setting the wrap bit in the descriptor. The idea is that the selector should be initialized to 0 (start of the descriptor area) but it can also be written with another 8 bytes aligned value to start somewhere in the middle of the area. It will still wrap to the beginning of the area.

If one wants to use a new descriptor table the receiver enable bit has to be cleared first. When the rxactive bit for the channel is cleared it is safe to update the descriptor table register. When this is finished and descriptors are enabled the receiver enable bit can be set again.

33.4.4.5 Enabling descriptors

As mentioned earlier one or more descriptors must be enabled before reception can take place. Each descriptor is 8 byte in size and the layout can be found in the tables below. The descriptors should be written to the memory area pointed to by the receiver descriptor table address register. When new descriptors are added they must always be placed after the previous one written to the area. Otherwise they will not be noticed.

A descriptor is enabled by setting the address pointer to point at a location where data can be stored and then setting the enable bit. The WR bit can be set to cause the selector to be set to zero when

reception has finished to this descriptor. IE should be set if an interrupt is wanted when the reception has finished. The DMA control register interrupt enable bit must also be set for an interrupt to be generated.

Table 378. RXDMA receive descriptor word 0 (address offset 0x0)

31	30	29	28	27	26	25	24	0
TR	DC	HC	EP	IE	WR	EN	PACKETLENGTH	

31	Truncated (TR) - Packet was truncated due to maximum length violation.
30	Data CRC (DC) - 1 if a CRC error was detected for the data and 0 otherwise.
29	Header CRC (HC) - 1 if a CRC error was detected for the header and 0 otherwise.
28	EEP termination (EP) - This packet ended with an Error End of Packet character.
27	Interrupt enable (IE) - If set, an interrupt will be generated when a packet has been received if the receive interrupt enable bit in the DMA channel control register is set.
26	Wrap (WR) - If set, the next descriptor used by the GRSPW will be the first one in the descriptor table (at the base address). Otherwise the descriptor pointer will be increased with 0x8 to use the descriptor at the next higher memory location. The descriptor table is limited to 1 kbytes in size and the pointer will be automatically wrap back to the base address when it reaches the 1 kbytes boundary.
25	Enable (EN) - Set to one to activate this descriptor. This means that the descriptor contains valid control values and the memory area pointed to by the packet address field can be used to store a packet.
24: 0	Packet length (PACKETLENGTH) - The number of bytes received to this buffer. Only valid after EN has been set to 0 by the GRSPW.

Table 379. RXDMA receive descriptor word 1 (address offset 0x4)

31	0
PACKETADDRESS	

31: 0	Packet address (PACKETADDRESS) - The address pointing at the buffer which will be used to store the received packet.

33.4.4.6 Setting up the DMA control register

The final step to receive packets is to set the control register in the following steps: The receiver must be enabled by setting the rxen bit in the DMA control register. This can be done anytime and before this bit is set nothing will happen. The rxdescav bit in the DMA control register is then set to indicate that there are new active descriptors. This must always be done after the descriptors have been enabled or the port might not notice the new descriptors. More descriptors can be activated when reception has already started by enabling the descriptors and writing the rxdescav bit. When these bits are set reception will start immediately when data is arriving.

33.4.4.7 The effect to the control bits during reception

When the receiver is disabled all packets going to the DMA-channel are discarded if the packet's address does not fall into the range of another DMA channel. If the receiver is enabled and the address falls into the accepted address range, the next state is entered where the rxdescav bit is checked. This bit indicates whether there are active descriptors or not and should be set by the external application using the DMA channel each time descriptors are enabled as mentioned above. If the rxdescav bit is '0' and the nospill bit is '0' the packets will be discarded. If nospill is one the grspw waits until rxdescav is set and the characters are kept in the N-Char fifo during this time. If the fifo becomes full further N-char transmissions are inhibited by stopping the transmission of FCTs.

When rxdescav is set the next descriptor is read and if enabled the packet is received to the buffer. If the read descriptor is not enabled, rxdescav is set to '0' and the packet is spilled depending on the value of nospill.

The receiver can be disabled at any time and will stop packets from being received to this channel. If a packet is currently received when the receiver is disabled the reception will still be finished. The rxdescav bit can also be cleared at any time. It will not affect any ongoing receptions but no more descriptors will be read until it is set again. Rxdescav is also cleared by the port when it reads a disabled descriptor.

33.4.4.8 Status bits

When the reception of a packet is finished the enable bit in the current descriptor is set to zero. When enable is zero, the status bits are also valid and the number of received bytes is indicated in the length field. The DMA control register contains a status bit which is set each time a packet has been received. The port can also be made to generate an interrupt for this event.

The RMAP CRC calculation is always active for all received packets and all bytes except the EOP/EEP are included. The packet is always assumed to be a RMAP packet and the length of the header is determined by checking byte 3 which should be the command field. The calculated CRC value is then checked when the header has been received (according to the calculated number of bytes) and if it is non-zero the HC bit is set indicating a header CRC error.

The CRC value is not set to zero after the header has been received, instead the calculation continues in the same way until the complete packet has been received. Then if the CRC value is non-zero the DC bit is set indicating a data CRC error. This means that the port can indicate a data CRC error even if the data field was correct when the header CRC was incorrect. However, the data should not be used when the header is corrupt and therefore the DC bit is unimportant in this case. When the header is not corrupted the CRC value will always be zero when the calculation continues with the data field and the behaviour will be as if the CRC calculation was restarted

If the received packet is not of RMAP type the header CRC error indication bit cannot be used. It is still possible to use the DC bit if the complete packet is covered by a CRC calculated using the RMAP CRC definition. This is because the port does not restart the calculation after the header has been received but instead calculates a complete CRC over the packet. Thus any packet format with one CRC at the end of the packet calculated according to RMAP standard can be checked using the DC bit.

If the packet is neither of RMAP type nor of the type above with RMAP CRC at the end, then both the HC and DC bits should be ignored.

33.4.4.9 Error handling

If an AHB error occurs during reception the current packet is spilled up to and including the next EEP/EOP and then the currently active channel is disabled and the receiver enters the idle state. A bit in the channels control/status register is set to indicate this condition.

33.4.4.10 Promiscuous mode

The port supports a promiscuous mode where all received data (excluding EOPs/EEPs) is stored to the first enabled DMA channel regardless of the node address and possible early EOPs/EEPs. The AMBA port DMA RX maximum length register is still checked and packets exceeding this size are truncated.

RMAP commands are still handled by the RMAP hardware target when promiscuous mode is enabled, if the RMAP enable bit in the AMBA port DMA control/status register is set. If the RMAP enable bit is cleared, RMAP commands are stored to a DMA channel instead.

33.4.5 Transmitter DMA channels

The transmitter DMA engine handles transmission of data from the DMA channels to the SpaceWire network. Each receive channel has a corresponding transmit channel which means there can be up to 4 transmit channels. It is however only necessary to use a separate transmit channel for each receive

channel if there are also separate entities controlling the transmissions. The use of a single channel with multiple controlling entities would cause them to corrupt each other's transmissions. A single channel is more efficient and should be used when possible.

Multiple transmit channels with pending transmissions are arbitrated in a round-robin fashion.

33.4.5.1 Basic functionality of a channel

A transmit DMA channel reads data from the AHB bus and stores them in the transmitter FIFO for transmission on the SpaceWire network. Transmission is based on the same type of descriptors as for the receiver and the descriptor table has the same alignment and size restrictions. When there are new descriptors enabled the port reads them and transfer the amount data indicated.

33.4.5.2 Setting up the core for transmission

Four steps need to be performed before transmissions can be done with the port. First the link interface must be enabled and started by writing the appropriate value to the ctrl register. Then the address to the descriptor table needs to be written to the transmitter descriptor table address register and one or more descriptors must also be enabled in the table. Finally, the txen bit in the DMA control register is written with a one which triggers the transmission. These steps will be covered in more detail in the next sections.

33.4.5.3 Enabling descriptors

The descriptor table address register works in the same way as the receiver's corresponding register which was covered in section 33.4.4. The maximum size is 1024 bytes as for the receiver but since the descriptor size is 16 bytes the number of descriptors is 64.

To transmit packets one or more descriptors have to be initialized in memory which is done in the following way: The number of bytes to be transmitted and a pointer to the data has to be set. There are two different length and address fields in the transmit descriptors because there are separate pointers for header and data. If a length field is zero the corresponding part of a packet is skipped and if both are zero no packet is sent. The maximum header length is 255 bytes and the maximum data length is 16 Mbyte - 1. When the pointer and length fields have been set the enable bit should be set to enable the descriptor. This must always be done last. The other control bits must also be set before enabling the descriptor.

The transmit descriptors are 16 bytes in size so the maximum number in a single table is 64. The different fields of the descriptor together with the memory offsets are shown in the tables below.

The HC bit should be set if RMAP CRC should be calculated and inserted for the header field and correspondingly the DC bit should be set for the data field. The header CRC will be calculated from the data fetched from the header pointer and the data CRC is generated from data fetched from the data pointer. The CRCs are appended after the corresponding fields. The NON-CRC bytes field is set to the number of bytes in the beginning of the header field that should not be included in the CRC calculation.

The CRCs are sent even if the corresponding length is zero, but when both lengths are zero no packet is sent not even an EOP.

33.4.5.4 Starting transmissions

When the descriptors have been initialized, the transmit enable bit in the DMA control register has to be set to tell the port to start transmitting. New descriptors can be activated in the table on the fly (while transmission is active). Each time a set of descriptors is added the transmit enable register bit should be set. This has to be done because each time the core encounters a disabled descriptor this register bit is set to 0.

Table 380. TXDMA transmit descriptor word 0 (address offset 0x0)

31 18	17	16	15	14	13	12	11 8	7 0
RESERVED	DC	HC	RE	IE	WR	EN	NONCRCLEN	HEADERLEN

31: 18	RESERVED
17	Append data CRC (DC) - Append CRC calculated according to the RMAP specification after the data sent from the data pointer. The CRC covers all the bytes from this pointer. A null CRC will be sent if the length of the data field is zero.
16	Append header CRC (HC) - Append CRC calculated according to the RMAP specification after the data sent from the header pointer. The CRC covers all bytes from this pointer except a number of bytes in the beginning specified by the non-crc bytes field. The CRC will not be sent if the header length field is zero.
15	RESERVED
14	Interrupt enable (IE) - If set, an interrupt will be generated when the packet has been transmitted and the transmitter interrupt enable bit in the DMA control register is set.
13	Wrap (WR) - If set, the descriptor pointer will wrap and the next descriptor read will be the first one in the table (at the base address). Otherwise the pointer is increased with 0x10 to use the descriptor at the next higher memory location.
12	Enable (EN) - Enable transmitter descriptor. When all control fields (address, length, wrap and crc) are set, this bit should be set. While the bit is set the descriptor should not be touched since this might corrupt the transmission. The GRSPW clears this bit when the transmission has finished.
11: 8	Non-CRC bytes (NONCRCLEN)- Sets the number of bytes in the beginning of the header which should not be included in the CRC calculation. This is necessary when using path addressing since one or more bytes in the beginning of the packet might be discarded before the packet reaches its destination.
7: 0	Header length (HEADERLEN) - Header Length in bytes. If set to zero, the header is skipped.

Table 381. TXDMA transmit descriptor word 1 (address offset 0x4)

31 0
HEADERADDRESS

31: 0	Header address (HEADERADDRESS) - Address from where the packet header is fetched. Does not need to be word aligned.

Table 382. TXDMA transmit descriptor word 2 (address offset 0x8)

31 24	23 0
RESERVED	DATALEN

31: 24	RESERVED
23: 0	Data length (DATALEN) - Length of data part of packet. If set to zero, no data will be sent. If both data- and header-lengths are set to zero no packet will be sent.

Table 383. TXDMA transmit descriptor word 3(address offset 0xC)

31 0
DATAADDRESS

31: 0	Data address (DATAADDRESS) - Address from where data is read. Does not need to be word aligned.

33.4.5.5 The transmission process

When the txen bit is set the port starts reading descriptors immediately. The number of bytes indicated are read and transmitted. When a transmission has finished, status will be written to the first field of the descriptor and a packet sent bit is set in the DMA control register. If an interrupt was requested it will also be generated. Then a new descriptor is read and if enabled a new transmission starts, otherwise the transmit enable bit is cleared and nothing will happen until it is enabled again.

33.4.5.6 The descriptor table address register

The internal pointer which is used to keep the current position in the descriptor table can be read and written through the APB interface. This pointer is set to zero during reset and is incremented each time a descriptor is used. It wraps automatically when the 1024 bytes limit for the descriptor table is reached or it can be set to wrap earlier by setting a bit in the current descriptor.

The descriptor table register can be updated with a new table anytime when no transmission is active. No transmission is active if the transmit enable bit is zero and the complete table has been sent or if the table is aborted (explained below). If the table is aborted one has to wait until the transmit enable bit is zero before updating the table pointer.

33.4.5.7 Error handling

33.4.5.7.1Abort Tx

The DMA control register contains a bit called Abort TX which if set causes the current transmission to be aborted, the packet is truncated and an EEP is inserted. This is only useful if the packet needs to be aborted because of congestion on the SpaceWire network. If the congestion is on the AHB bus this will not help (This should not be a problem since AHB slaves should have a maximum of 16 wait-states). The aborted packet will have its LE bit set in the descriptor. The transmit enable register bit is also cleared and no new transmissions will be done until the transmitter is enabled again.

33.4.5.7.2AHB error

When an AHB error is encountered during transmission the currently active DMA channel is disabled and the transmitter goes to the idle mode. A bit in the DMA channel's control/status register is set to indicate this error condition and, if enabled, an interrupt will also be generated. Further error handling depends on what state the transmitter DMA engine was in when the AHB error occurred. If the descriptor was being read the packet transmission had not been started yet and no more actions need to be taken.

If the AHB error occurs during packet transmission the packet is truncated and an EEP is inserted. Lastly, if it occurs when status is written to the descriptor the packet has been successfully transmitted but the descriptor is not written and will continue to be enabled (this also means that no error bits are set in the descriptor for AHB errors).

The client using the channel has to correct the AHB error condition and enable the channel again. No more AHB transfers are done again from the same unit (receiver or transmitter) which was active during the AHB error until the error state is cleared and the unit is enabled again.

33.4.6 RMAP target

The Remote Memory Access Protocol (RMAP) is used to implement access to resources on the AHB bus via the SpaceWire Link. Some common operations are reading and writing to memory, registers and FIFOs. The port has an optional hardware RMAP target. This section describes the target implementation.

33.4.6.1 Fundamentals of the protocol

RMAP is a protocol which is designed to provide remote access via a SpaceWire network to memory mapped resources on a SpaceWire node. It has been assigned protocol ID 0x01. It provides three operations write, read and read-modify-write. These operations are posted operations which means that a source does not wait for an acknowledge or reply. It also implies that any number of operations can be outstanding at any time and that no timeout mechanism is implemented in the protocol. Time-outs must be implemented in the user application which sends the commands. Data payloads of up to 16 Mb - 1 is supported in the protocol. A destination can be requested to send replies and to verify data before executing an operation. A complete description of the protocol is found in the RMAP standard.

33.4.6.2 Implementation

The port includes a target for RMAP commands which processes all incoming packets with protocol ID = 0x01, type field (bit 7 and 6 of the 3rd byte in the packet) equal to 01b and an address falling in the range set by the default address and mask register. When such a packet is detected it is not stored to the DMA channel, instead it is passed to the RMAP receiver.

The target implements all three commands defined in the standard with some restrictions. Support is only provided for 32-bit big-endian systems. This means that the first byte received is the msb in a word. The target will not receive RMAP packets using the extended protocol ID which are always dumped to the DMA channel.

The RMAP receiver processes commands. If they are correct and accepted the operation is performed on the AHB bus and a reply is formatted. If an acknowledge is requested the RMAP transmitter automatically send the reply. RMAP transmissions have priority over DMA channel transmissions.

There is a user accessible destination key register which is compared to destination key field in incoming packets. If there is a mismatch and a reply has been requested the error code in the reply is set to 3. Replies are sent if and only if the ack field is set to ‘1’.

When a failure occurs during a bus access the error code is set to 1 (General Error). There is predetermined order in which error-codes are set in the case of multiple errors in the core. It is shown in table 384.

Table 384. The order of error detection in case of multiple errors in the AMBA port. The error detected first has number 1.

Detection Order	Error Code	Error
1	12	Invalid destination logical address
2	2	Unused RMAP packet type or command code
3	3	Invalid destination key
4	9	Verify buffer overrun
5	11	RMW data length error
6	10	Authorization failure
7*	1	General Error (AHB errors during non-verified writes)
8	5/7	Early EOP / EEP (if early)
9	4	Invalid Data CRC
10	1	General Error (AHB errors during verified writes or RMW)
11	7	EEP
12	6	Too Much Data
*The AHB error is not guaranteed to be detected before Early EOP/EEP or Invalid Data CRC. For very long accesses the AHB error detection might be delayed causing the other two errors to appear first.		

Read accesses are performed on the fly, that is they are not stored in a temporary buffer before transmitting. This means that the error code 1 will never be seen in a read reply since the header has already been sent when the data is read. If the AHB error occurs the packet will be truncated and ended with an EEP.

Errors up to and including Invalid Data CRC (number 8) are checked before verified commands. The other errors do not prevent verified operations from being performed.

The details of the support for the different commands are now presented. All defined commands which are received but have an option set which is not supported in this specific implementation will not be executed and a possible reply is sent with error code 10.

33.4.6.3 Write commands

The write commands are divided into two subcategories when examining their capabilities: verified writes and non-verified writes. Verified writes have a length restriction of 4 bytes and the address must be aligned to the size. That is 1 byte writes can be done to any address, 2 bytes must be halfword aligned, 3 bytes are not allowed and 4 bytes writes must be word aligned. Since there will always be only on AHB operation performed for each RMAP verified write command the incrementing address bit can be set to any value.

Non-verified writes have no restrictions when the incrementing bit is set to 1. If it is set to 0 the number of bytes must be a multiple of 4 and the address word aligned. There is no guarantee how many words will be written when early EOP/EEP is detected for non-verified writes.

33.4.6.4 Read commands

Read commands are performed on the fly when the reply is sent. Thus if an AHB error occurs the packet will be truncated and ended with an EEP. There are no restrictions for incrementing reads but non-incrementing reads have the same alignment restrictions as non-verified writes. Note that the “Authorization failure” error code will be sent in the reply if a violation was detected even if the length field was zero. Also note that no data is sent in the reply if an error was detected i.e. if the status field is non-zero.

33.4.6.5 RMW commands

All read-modify-write sizes are supported except 6 which would have caused 3 B being read and written on the bus. The RMW bus accesses have the same restrictions as the verified writes. As in the verified write case, the incrementing bit can be set to any value since only one AHB bus operation will be performed for each RMW command. Cargo too large is detected after the bus accesses so this error will not prevent the operation from being performed. No data is sent in a reply if an error is detected i.e. the status field is non-zero.

33.4.6.6 Control

The RMAP target mostly runs in the background without any external intervention, but there are a few control possibilities.

There is an enable bit in the control register of the core which can be used to completely disable the RMAP target. When it is set to ‘0’ no RMAP packets will be handled in hardware, instead they are all stored to the DMA channel.

There is a possibility that RMAP commands will not be performed in the order they arrive. This can happen if a read arrives before one or more writes. Since the target stores replies in a buffer with more than one entry several commands can be processed even if no replies are sent. Data for read replies is read when the reply is sent and thus writes coming after the read might have been performed already if there was congestion in the transmitter. To avoid this the RMAP buffer disable bit can be set to force the target to only use one buffer which prevents this situation.

The last control option for the target is the possibility to set the destination key which is found in a separate register.

Table 385. AMBA port hardware RMAP handling of different packet type and command fields.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Command	Action
Reserved	Command / Response	Write / Read	Verify data before write	Acknow-ledge	Increment Address		
0	0	-	-	-	-	Response	Stored to DMA-channel.
0	1	0	0	0	0	Not used	Does nothing. No reply is sent.
0	1	0	0	0	1	Not used	Does nothing. No reply is sent.
0	1	0	0	1	0	Read single address	Executed normally. Address has to be word aligned and data size a multiple of four. Reply is sent. If alignment restrictions are violated error code is set to 10.
0	1	0	0	1	1	Read incre-menting address.	Executed normally. No restric-tions. Reply is sent.
0	1	0	1	0	0	Not used	Does nothing. No reply is sent.
0	1	0	1	0	1	Not used	Does nothing. No reply is sent.
0	1	0	1	1	0	Not used	Does nothing. Reply is sent with error code 2.
0	1	0	1	1	1	Read-Mod-ify-Write increment-ing address	Executed normally. If length is not one of the allowed rmw val-ues nothing is done and error code is set to 11. If the length was correct, alignment restric-tions are checked next. 1 byte can be rmw to any address. 2 bytes must be halfword aligned. 3 bytes are not allowed. 4 bytes must be word aligned. If these restrictions are violated nothing is done and error code is set to 10. If an AHB error occurs error code is set to 1. Reply is sent.
0	1	1	0	0	0	Write, sin-gle-address, do not verify before writ-ing, no acknowledge	Executed normally. Address has to be word aligned and data size a multiple of four. If alignment is violated nothing is done. No reply is sent.
0	1	1	0	0	1	Write, incre-menting address, do not verify before writ-ing, no acknowledge	Executed normally. No restric-tions. No reply is sent.
0	1	1	0	1	0	Write, sin-gle-address, do not verify before writ-ing, send acknowledge	Executed normally. Address has to be word aligned and data size a multiple of four. If alignment is violated nothing is done and error code is set to 10. If an AHB error occurs error code is set to 1. Reply is sent.

Table 385. AMBA port hardware RMAP handling of different packet type and command fields.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Command	Action
Reserved	Command / Response	Write / Read	Verify data before write	Acknow-ledge	Increment Address		
0	1	1	0	1	1	Write, incre-menting address, do not verify before writ-ing, send acknowledge	Executed normally. No restric-tions. If AHB error occurs error code is set to 1. Reply is sent.
0	1	1	1	0	0	Write, single address, ver-ify before writing, no acknowledge	Executed normally. Length must be 4 or less. Otherwise nothing is done. Same alignment restric-tions apply as for rmw. No reply is sent.
0	1	1	1	0	1	Write, incre-menting address, ver-ify before writing, no acknowledge	Executed normally. Length must be 4 or less. Otherwise nothing is done. Same alignment restric-tions apply as for rmw. If they are violated nothing is done. No reply is sent.
0	1	1	1	1	0	Write, single address, ver-ify before writing, send acknowledge	Executed normally. Length must be 4 or less. Otherwise nothing is done and error code is set to 9. Same alignment restrictions apply as for rmw. If they are vio-lated nothing is done and error code is set to 10. If an AHB error occurs error code is set to 1. Reply is sent.
0	1	1	1	1	1	Write, incre-menting address, ver-ify before writing, send acknowledge	Executed normally. Length must be 4 or less. Otherwise nothing is done and error code is set to 9. Same alignment restrictions apply as for rmw. If they are vio-lated nothing is done and error code is set to 10. If an AHB error occurs error code is set to 1. Reply is sent.
1	0	-	-	-	-	Unused	Stored to DMA-channel.
1	1	-	-	-	-	Unused	Stored to DMA-channel.

33.4.7 AMBA interface

The AMBA interface consists of an APB interface, an AHB master interface and DMA FIFOs. The APB interface provides access to the user registers. The DMA engines have 32-bit wide FIFOs to the AHB master interface which are used when reading and writing to the bus.

The transmitter DMA engine reads data from the bus in bursts which are half the FIFO size in length. A burst is always started when the FIFO is half-empty or if it can hold the last data for the packet. The burst containing the last data might have shorter length if the packet is not an even number of bursts in size.

The receiver DMA works in the same way except that it checks if the FIFO is half-full and then performs a burst write to the bus which is half the fifo size in length. The last burst might be shorter. Byte accesses are used for non word-aligned buffers and/or packet lengths that are not a multiple of four

bytes. There might be 1 to 3 single byte writes when writing the beginning and end of the received packets.

33.4.7.1 APB slave interface

As mentioned above, the APB interface provides access to the user registers which are 32-bits in width. The accesses to this interface are required to be aligned word accesses. The result is undefined if this restriction is violated.

33.4.7.2 AHB master interface

The port contains a single master interface which is used by both the transmitter and receiver DMA engines. The arbitration algorithm between the channels is done so that if the current owner requests the interface again it will always acquire it. This will not lead to starvation problems since the DMA engines always deassert their requests between accesses.

33.4.8 Registers

The port is programmed through registers mapped into APB address space. The addresses in the table below are offsets from each port's base address. The actual AMBA AHB address used to access the port is determined as follows: The AMBA ports' registers are accessed through an APB interface which resides on the APB bus. The APB bus is connected to the AHB bus using an APB controller whose AHB address determines the first base address for the AMBA ports.

.

Table 386. AMBA port registers

APB address offset**	Register name	Acronym
0x00	AMBA port Control	RTR.AMBACTRL
0x40	AMBA port Status	RTR.AMBASTS
0x08	AMBA port Default address	RTR.AMBADEFADDR
0x0C	RESERVED	
0x10	AMBA port Destination key	RTR.AMBADKEY
0x14	AMBA port Time-code	RTR.AMBATC
0x18	RESERVED	
0x20, 0x40, 0x60, 0x80	AMBA port DMA control/status, channels 1 - 4 *	RTR.AMBADMACTRL
0x24, 0x44, 0x64, 0x84	AMBA port DMA RX maximum length, channels 1 - 4 *	RTR.AMBADMAMAXLEN
0x28, 0x48, 0x68, 0x88	AMBA port DMA transmit descriptor table address, channels 1 - 4 *	RTR.AMBADMATXDESC
0x2C, 0x4C, 0x6C, 0x8C	AMBA port DMA receive descriptor table address, channels 1 - 4 *	RTR.AMBADMARXDESC
0x30, 0x50, 0x70, 0x90	AMBA port DMA address, channels 1 - 4 *	RTR.AMBADMAADDR
0x34, 0x54, 0x74, 0x94	RESERVED	
0x38, 0x58, 0x78, 0x98	RESERVED	
0x3C, 0x5C, 0x7C, 0x9C	RESERVED	
0xA0	AMBA port Distributed interrupt control	RTR.AMBAINTCTRL
0xA4	AMBA port Interrupt receive	RTR.AMBAINTRX
0xA8	AMBA port Interrupt acknowledgement / extended interrupt receive	RTR.AMBAACKRX
0xAC	AMBA port Interrupt timeout, interrupt 0-31	RTR.AMBAINTTO0
0xB0	AMBA port Interrupt timeout, interrupt 32-63	RTR.AMBAINTTO1
0xB4	AMBA port Interrupt mask, interrupt 0-31	RTR.AMBAINTMSK0
0xB8	AMBA port Interrupt mask, interrupt 32-63	RTR.AMBAINTMSK1
0xBC - 0xFF	RESERVED	

* One identical register per DMA channel. Register is only described once

Table 387. 0x00 - RTR.AMBACTRL - AMBA port Control

31	30	29	28 27	26 25	24	23	22 21 20 19 18	17	16	15 14 13 12 11 10 9	8	7	6	5	4	3	2 1 0
RA	RX	RC	NCH	RES	DI	ME	RESERVED	RD	RE	RESERVED	TQ	R	RS	PM	TI	IE	RESERVED
1	1	1	0x3	0x0	1	0	0x00	0	1	0x00	0	0	0	0	0	0	0x0
r	r	r	r	r	r	rw	r	rw	rw	r	rw	r	rw	rw	rw	rw	r

31 RMAP available (RA) - Constant value of 1, indicating that the RMAP target is implemented.

30 RX unaligned access (RX) - Constant value of 1, indicating that unaligned writes are available for the receiver.

29 RMAP CRC available (RC) - Constant value of 1, indicating that RMAP CRC is enabled.

28: 27 Number of DMA channels (NCH) - The number of available DMA channels minus one. Constant value of 0x3.

26: 25 RESERVED

24 Distributed interrupt support (DI) - Constant value of 1, indicating that distributed interrupts are supported.

23 Memory error truncation enable (ME) - If set to 1, a packet being transmitted will be truncated with an EEP if an error occur while reading from the AMBA port's TX FIFO.

22: 18 RESERVED

17 RMAP buffer disable (RD) - If set only one RMAP buffer is used. This ensures that all RMAP commands will be executed consecutively.

16 RMAP Enable (RE) - Enable RMAP target.

15: 9 RESERVED

8 Tick-out IRQ (TQ) - Generate interrupt when a valid time-code is received if the time-code also matches the time-code filter specified by the RTR.AMBATIME.TCMSK and RTR.AMBATIME.TCVAL fields.

7 Time-code tick-out enable (TO) - If set to 1, the internal tickout signal is set when a valid time-code is received. if the time-code also matches the time-code filter specified by the RTR.AMBATIME.TCMSK and RTR.AMBA-TIME.TCVAL fields.

6 Reset (RS) - Make complete reset of the SpaceWire node. Self clearing.

5 Promiscuous Mode (PM) - Enable Promiscuous mode.

4 Tick In (TI) - The host can generate a tick by writing a one to this field. This will increment the timer counter and the new value is transmitted after the current character is transferred. A tick can also be generated by asserting the tick_in signal.

3 Interrupt Enable (IE) - If set, an interrupt is generated when bit 8 is set and its corresponding event occurs.

2: 0 RESERVED

Table 388. 0x04 - RTR.AMBASTS - AMBA port Status

31	30 28	27 26	25 24	23 13	12	11 9	8	7	7 0	0
R	NIRQ	NRXD	NTXD	RESERVED	ME	RESERVED	EE	IA	RESERVED	TO
0	0x6	0	0	0x000	0	0x0	0	0	0x00	0
r	r	r	r	r	wc	r	wc	wc	r	wc

31 RESERVED

30: 28 Number of interrupts (NIRQ) - Shows the number of support distributed interrupt, according to the formula 2^{NIRQ}. Constant value of 0x6 = 64 supported interrupts.

27: 26 Number of receive descriptors (NRXD) - Shows the size of the DMA receive descriptor table. Constant value of 0, indicating 128 descriptors.

25: 24 Number of transmit descriptors (NTXD) - Shows the size of the DMA transmit descriptor table. Constant value of 0, indicating 64 descriptors.

23: 13 RESERVED

12 Memory error packet truncation (ME) - This bit is set to one when a transmitted packet is truncated with an EEP due to a memory error in the AMBA ports' TX FIFO.

11: 9 RESERVED

8 Early EOP/EEP (EE) - Set to one when a packet is received with an EOP after the first byte for a non-RMAP packet and after the second byte for an RMAP packet.

7 Invalid Address (IA) - Set to one when a packet is received with an invalid destination address field, i.e it does not match the nodeaddr register.

6: 1 RESERVED

0 Tick Out (TO) - A new time count value was received and is stored in the time counter field.

Table 389. 0x08 - RTR.AMBADEFADDR - AMBA port Default address

31 16	15 8	7 0
RESERVED	DEFMASK	DEFADDR
0x0000	0x00	0xFE
r	rw	rw

31: 8 RESERVED

15: 8 Default mask (DEFMASK) - Default mask used for node identification on the SpaceWire network. This field is used for masking the address before comparison. Both the received address and the DEFADDR field are anded with the inverse of DEFMASK before the address check.

7: 0 Default address (DEFADDR) - Default address used for node identification on the SpaceWire network. Reset value: 254.

Table 390. 0x10 - RTR.AMBADKEY - AMBA port Destination key

31 8	7 0
RESERVED	DESTKEY
NA	0x00
r	rw

31: 8 RESERVED

7: 0 Destination key (DESTKEY) - RMAP destination key.

Table 391. 0x14 - RTR.AMBATC - AMBA port Time-code

31 24	23 16	15 8	7 6	5 0
TCMSK	TCVAL	RESERVED	TCTRL	TIMECNT
0x00	0x00	0x00	0x0	0x00
rw	rw	r	rw	rw

31: 24 Time-code filter mask (TCMSK) - If a bit in this field is set to 1 then the corresponding bit in the RTR.AMBA-TIME.TCVAL field must match the value of the same bit in a received time-code in order for that time-code to generate an AMBA IRQ or a pulse on the internal tickout signals connected to the general purpose timers.

23: 16 Time-code filter value (TCVAL) - For each bit set to 1 in the RTR.AMBATIME.TCMSK, the corresponding bit in this field must match the value of the same bit of a received time-code in order to that time-code to generate and AMBA IRQ, or a pulse on the internal tickout signals connected to the general purpose timers.

15: 8 RESERVED

7: 6 Time control flags (TCTRL) - The current value of the time control flags. Sent with time-code resulting from a tick-in. Received control flags are also stored in this register.

5: 0 Time counter (TIMECNT) - The current value of the system time counter. It is incremented for each tick-in and the incremented value is transmitted. The register can also be written directly but the written value will not be transmitted. Received time-counter values are also stored in this register

Table 392. 0x20,0x40,0x60,0x80 - RTR.AMBADMACTRL - AMBA port DMA control/status

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTNUM						RES		EP	TR	IE	IT	RP	TP	RES		SP	SA	EN	NS	RD	RX	AT	RA	TA	PR	PS	AI	RI	TI	RE	TE
0x00						0x0		0	0	0	0	0	0	0x0		0	0	0	0	0	0	0	0	0	0	0	NR	NR	NR	0	0
rw						r		wc	wc	rw	rw	wc	wc	r		rw	rw	rw	rw	rw	r	rw	wc	wc	wc	wc	rw	rw	rw	rw	rw

31: 26 Interrupt number (INTNUM) - The interrupt number used for this DMA channel when sending a distributed interrupt code that was generated due to any of the events maskable by the RTR.AMBADMACTRL.IE and RTR.AMBADMACTRL.IT bits. The value in this field should be in the range 0 to 31.

25: 24 RESERVED

23 EEP termination (EP) - Set to 1 when a received packet for the corresponding DMA channel ended with an Error End of Packet (EEP) character.

22 Truncated (TR) - Set to 1 when a received packet for the corresponding DMA channel is truncated due to a maximum length violation.

21 Interrupt transmit enable on EEP (IE) - When set to 1, the distributed interrupt code specified in the RTR.AMBADMACTRL.INTNUM field is generated when a received packet on this DMA channel ended with an Error End of Packet (EEP) character.

20 Interrupt-code transmit enable on truncation (IT) - When set to 1, the distributed interrupt code specified in the RTR.AMBADMACTRL.INTNUM field is generated when a received packet on this DMA channel is truncated due to a maximum length violation.

19 Receive packet IRQ (RP) - This bit is set to 1 when an AMBA interrupt was generated due to the fact that a packet was received for the corresponding DMA channel.

18 Transmit packet IRQ (TP) - This bit is set to 1 when an AMBA interrupt was generated due to the fact that a packet was transmitted for the corresponding DMA channel.

17: 16 RESERVED

15 Strip pid (SP) - Remove the pid byte (second byte) of each packet. The address byte (first byte) will also be removed when this bit is set independent of the SA bit.

14 Strip addr (SA) - Remove the addr byte (first byte) of each packet.

13 Enable addr (EN) - Enable separate node address for this channel.

12 No spill (NS) - If cleared, packets will be discarded when a packet is arriving and there are no active descriptors. If set, the GRSPW will wait for a descriptor to be activated.

11 Rx descriptors available (RD) - Set to one, to indicate to the GRSPW that there are enabled descriptors in the descriptor table. Cleared by the GRSPW when it encounters a disabled descriptor:

10 RX active (RX) - Is set to ‘1’ if a reception to the DMA channel is currently active otherwise it is ‘0’.

9 Abort TX (AT) - Set to one to abort the currently transmitting packet and disable transmissions. If no transmission is active the only effect is to disable transmissions. Self clearing.

8 RX AHB error (RA) - An error response was detected on the AHB bus while this receive DMA channel was accessing the bus. The AHB error interrupt (AI) field must be set to ’1’ for the RA field to be set.

7 TX AHB error (TA) - An error response was detected on the AHB bus while this transmit DMA channel was accessing the bus.

6 Packet received (PR) - This bit is set each time a packet has been received. never cleared by the SW-node.

5 Packet sent (PS) - This bit is set each time a packet has been sent. Never cleared by the SW-node.

4 AHB error interrupt (AI) - If set, an interrupt will be generated each time an AHB error occurs when this DMA channel is accessing the bus.

3 Receive interrupt (RI) - If set, an interrupt will be generated each time a packet has been received. This happens both if the packet is terminated by an EEP or EOP.

2 Transmit interrupt (TI) - If set, an interrupt will be generated each time a packet is transmitted. The interrupt is generated regardless of whether the transmission was successful or not.

1 Receiver enable (RE) - Set to one when packets are allowed to be received to this channel.

0 Transmitter enable (TE) - Write a one to this bit each time new descriptors are activated in the table. Writing a one will cause the SW-node to read a new descriptor and try to transmit the packet it points to. This bit is automatically cleared when the SW-node encounters a descriptor which is disabled.

Table 393. 0x24,0x44,0x64,0x84 - RTR.AMBADMAMAXLEN - AMBA port DMA RX maximum length

31 25	24 2	1 0
RESERVED	RXMAXLEN	RES
0x00	N/R	0x0
r	rw	r

31: 25 RESERVED

24: 2 RX maximum length (RXMAXLEN) - Receiver packet maximum length in 32-bit words.

1: 0 RESERVED

Table 394. 0x28,0x48,0x68,0x88 - RTR.AMBADMATXDESC - AMBA port DMA transmit descriptor table address

31 10	9 4	3 0
DESCBASEADDR	DESCSEL	RESERVED
N/R	0x00	0x0
rw	rw	r

31: 10 Descriptor table base address (DESCBASEADDR) - Sets the base address of the descriptor table.

9: 4 Descriptor selector (DESCSEL) - Offset into the descriptor table. Shows which descriptor is currently used by the GRSPW. For each new descriptor read, the selector will increase with 16 and eventually wrap to zero again.

3: 0 RESERVED

Table 395. 0x2C,0x4C,0x6C,0x8C - RTR.AMBADMARXDESC - AMBA port DMA receive descriptor table address

31 10	9 3	2 0
DESCBASEADDR	DESCSEL	RESERVED
N/R	0x00	0x0
rw	rw	r

31: 10 Descriptor table base address (DESCBASEADDR) - Sets the base address of the descriptor table. Not reset.

9: 3 Descriptor selector (DESCSEL) - Offset into the descriptor table. Shows which descriptor is currently used by the GRSPW. For each new descriptor read, the selector will increase with 8 and eventually wrap to zero again. Reset value: 0.

2: 0 RESERVED

Table 396. 0x30,0x50,0x70,0x90 - RTR.AMBADMAADDR - AMBA port DMA address

31 16	15 8	7 0
RESERVED	MASK	ADDR
0x0000	N/R	N/R
r	rw	rw

31: 8 RESERVED

15: 8 Mask (MASK) - Mask used for node identification on the SpaceWire network. This field is used for masking the address before comparison. Both the received address and the ADDR field are anded with the inverse of MASK before the address check.

7: 0 Address (ADDR) - Address used for node identification on the SpaceWire network for the corresponding dma channel when the EN bit in the DMA control register is set.

Table 397. 0xA0 - RTR.AMBAINTCTRL - AMBA port Distributed interrupt control

31 26	25	24	23	22 21	20	19	18	17 16	15	14	13	12 8	7	6	5 0
INTNUM	R	EE	IA	RES	TQ	AQ	IQ	RES	AA	AT	IT	RESERVED	ID	II	TXINT
0x00	0	0	0	0x0	0	0	0	0x0	0	1	1	0x00	0	0	0x00
rw	r	rw	rw	r	rw	rw	rw	r	rw	rw	rw	r	wc	rw*	rw

31: 26 Interrupt number (INTNUM) - The interrupt-number used when sending an interrupt code that was generated due to any of the events maskable by the RTR.AMBAINCTRL.ER and RTR.AMBAINTCTRL.IA bits. Note that when RTR.RTRCFG.EE = 0 (interrupt with acknowledgement mode), this field must no be set to a value greater than 31.

25 RESERVED

24 Interrupt transmit on early EOP/EEP (EE) - If set to 1, a distributed interrupt code with the interrupt number specified in the RTR.AMBAINTCTRL.INTNUM field is sent each time an event occurs such that the STS.EE bit is set to 1 (even if the bit was already set when the event occurred).

23 Interrupt transmit on invalid address (IA) - If set to 1, a distributed interrupt code with the interrupt number specified in the RTR.AMBAINTCTRL.INTNUM field is sent each time an event occurs such that the STS.IA bit is set to 1 (even if the bit was already set when the event occurred).

22: 21 RESERVED

20 Interrupt timeout IRQ enable (TQ) - When set to 1, an AMBA interrupt is generated when a bit in the RTR.AMBAINTTO0 or RTR.AMBAINTTO1 registers is set. Note that the RTR.AMBACTRL.IE bit also must be set for this bit to have any effect.

19 Interrupt acknowledgment / extended interrupt receive IRQ enable (AQ) - When set to 1, an AMBA interrupt is generated when an interrupt acknowledgment code or extended interrupt code is received such that a bit in the RTR.AMBAACKRX register is set to 1 (even if the bit was already set when the code was received).

18 Interrupt-code receive IRQ enable (IQ) - When set to 1, an AMBA interrupt is generated when an interrupt code is received such that a bit in the RTR.AMBAINTRX register is set to 1 (even if the bit was already set when the code was received).

17: 16 RESERVED

15 Handle all interrupt acknowledgment codes (AA) - Is set to 0, only those received interrupt acknowledgment codes that match an interrupt code sent by software are handled. If set to 1, all received interrupt acknowledgment codes are handled.

14 Interrupt acknowledgment / extended interrupt tickout enable (AT) - When set to 1, the internal tickout signal from this AMBA port is set when an interrupt acknowledgment code or extended interrupt code is received such that a bit in the RTR.AMBAACKRX register is set to 1 (even if the bit was already set when the code was received).

13 Interrupt tickout enable (IT) - When set to 1, the internal tickout signal from this AMBA port is set when an interrupt code is received such that a bit in the RTR.AMBAINTRX register is set to 1 (even if the bit was already set when the code was received).

12: 8 RESERVED

7 Interrupt discarded (ID) - This bit is set to 1 when a distributed interrupt code that software tried to send by writing the RTR.AMBAINTCTRL.II bit was discarded by the routers switch matrix. There is a maximum of ten clock cycle delay between the RTR.AMBAINTCTRL.II bit being written and this bit being set.

6 Interrupt-code tick-in (II) - When this field is written to 1 the distributed interrupt code specified by the RTR.AMBAINTCTRL.TXINT field will be sent out from the AMBA port to the routers switch matrix. This bit is automatically cleared and always reads ‘0’. Writing a ‘0’ has no effect.

5: 0 Transmit distributed interrupt code (TXINT) - The distributed interrupt code that will be sent when the register RTR.AMBAINTCTRL.II is written with 1.

Table 398. 0xA4 - RTR.AMBAINTRX - AMBA port Interrupt receive

31	0
RXIRQ	
0x00000000	
wc	

31: 0 Received interrupt code (RXIRQ) - Each bit corresponds to the interrupt number with the same number as the bit index. A position is set to 1 when an interrupt code is received for which the corresponding bit in the RTR.AMBAINTMSK0 register is set to 1.

Table 399. 0xA8 - RTR.AMBAACKRX - AMBA port Interrupt acknowledgment / extended interrupt receive

31	0
RXACK	
0x00000000	
wc	

31: 0 Received interrupt acknowledgment code / extended interrupt code (RXACK) - When operating in extended interrupt mode (RTR.RTRCFG.EE = 1) then each bit corresponds to the interrupt number with the same number as the bit index plus 32, i.e bit 0 corresponds to interrupt number 32, bit 1 to interrupt number 33 etc. This bit gets set to 1 when an extended interrupt code is received for which the corresponding bit in the RTR.AMBAINTMSK1 register is set to 1.

When operating in interrupt with acknowledgment mode (RTR.RTRCFG.EE = 0) then each bit corresponds to the interrupt number with the same number as the bit index. This bit gets set to 1 an interrupt acknowledgment code is received for which the corresponding bit in the RTR.AMBAINTMSK0 register is set, and either if RTR.AMBAINTCTRL.AA is set to 1 or for which the matching interrupt code was sent by software.

Table 400. 0xAC - RTR.AMBAINTTO0 - AMBA port Interrupt timeout, interrupt 0-31

31	0
INTTO	
0x00000000	
wc	

31: 0 Interrupt code timeout (INTTO) - Each bit corresponds to the interrupt number with the same number as the bit index. This bit is set to 1 when an interrupt code that was sent by software doesn't receive an interrupt acknowledgment code for the duration of a timeout period (specified in the RTR.ISRTIMER register), and if the corresponding bit in the RTR.AMBAINTMSK0 register is set.

Table 401. 0xAC - RTR.AMBAINTTO1 - AMBA port Interrupt timeout, interrupt 32-63

31	0
INTTO	
0x00000000	
wc	

31: 0 Extended interrupt code timeout (INTTO) - Each bit corresponds to the interrupt number with the same number as the bit index plus 32, i.e bit 0 corresponds to interrupt number 32, bit 1 to interrupt number 33 etc.. This bit is set to 1 when an extended interrupt code that was sent by software time out, i.e after the duration of a timeout period (specified in the RTR.ISRTIMER register), and if the corresponding bit in the RTR.AMBAINTMSK1 register is set.

Table 402. 0xB0 - RTR.AMBAINTMSK0 - AMBA port Interrupt mask, interrupt 0-31

31	0
MASK	
0x00000000	
rw	

31: 0 Interrupt mask (MASK) - Each bit corresponds to the interrupt number with the same value as the bit index. If a bit is set to 0, all received interrupt codes and interrupt acknowledgment codes with the interrupt identifier corresponding to that bit is ignored. If a bit is set to 1, then the matching distributed interrupt code is handled.

Table 403. 0xB0 - RTR.AMBAINTMSK1 - AMBA port Interrupt mask, interrupt 32-63

31	0
MASK	
0x00000000	
rw	

31: 0 Interrupt mask (MASK) - Each bit corresponds to the interrupt number with the same value as the bit index. plus 32, i.e bit 0 corresponds to interrupt number 32, bit 1 to interrupt number 33 etc. If a bit is set to 0, all received extended interrupt codes with the interrupt identifier corresponding to that bit is ignored. If a bit is set to 1, then the matching distributed interrupt code is handled.

33.5 Configuration port

The configuration port has port number 0. It consists of an RMAP target, AMBA AHB slave interface, SpaceWire Plug-and-Play interface, and a set of configuration and status registers.

33.5.1 RMAP target

33.5.1.1 Overview

The configuration port's RMAP target implements the RMAP protocol, as defined in the RMAP standard [RMAP]. Verified writes and reads up to 128 B, and read-modify-writes of 4 B (8 B if the mask field is included in the count) are supported. Replies from the configuration port are always sent to the port they arrived on, regardless of the values of the RMAP command's Initiator Logical Address field, and Reply Address field. The address space of the configuration port is specified in section 33.6.

Additional requirements on the RMAP commands imposed by the configuration port's RMAP target are:

- The Target Logical Address field must be 0xFE.
- The Address fields must contain a 4 B aligned address.
- The Extended Address field must be 0x00.
- Key field must be 0x00.
- For write and read commands the Data Length fields must contain a value that is a multiple of 4, ranging from 0 to 128 B.
- For read-modify-write commands the Data Length fields must contain a value of 0 or 8.
- For write commands the Verify Data Before Write bit in the Instruction field must be set to 1.

How the RMAP target handles commands that does not meet the above requirement is detailed in sections 33.5.1.2 and 33.5.1.4.

When an RMAP write command larger than 4 bytes is processed (i.e. more than one register written by the same command), the registers will not change value simultaneously. The command is buffered locally, since only verified write commands are allowed, and then read out from the buffer and written to the registers, with one CLK cycle delay between each register write. This needs to be considered when for example updating large parts of the routing table. Data traffic that arrives while the RMAP command is being processed may or may not be routed according to the new values depending on the address of the packet and how much of the RMAP command that has been processed. Note that the RMAP target is blocked while the RMAP write command is being processed, which means that data returned in an RMAP read command is always either the value before or after the complete RMAP write, never in between.

33.5.1.2 RMAP command support

Table 404 lists all possible RMAP commands and shows how the configuration port's RMAP target handles them. An RMAP command will always have bits 7:6 of the command's Instruction field set to "01", and those bits are therefore left out of the table. Bits 1:0 of the command's Instruction field determines the length of the command's Reply Address Field, and does not affect the action taken, so they have been left out of the table as well. The action taken assumes that no errors were detected in the RMAP packet. For handling of RMAP packet error, see section 33.5.1.4.

*Table 404.*RMAP command decoding and handling.

Bit 5	Bit 4	Bit 3	Bit 2		
Write / Read	Verify Data Before Write	Reply	Increment Addr	Function	Action taken
0	0	0	0	Invalid	No operation performed. Error code 0x02 is saved in the RTR.PCTRLCFG.EC field. No reply is sent.
0	0	0	1	Invalid	No operation performed. Error code 0x02 is saved in the RTR.PCTRLCFG.EC field. No reply is sent.
0	0	1	0	Read single address	Read operation performed, if the requirements in section 33.5.1.1 are met.
0	0	1	1	Read incrementing address	Read operation performed, if the requirements in section 33.5.1.1 are met.
0	1	0	0	Invalid	No operation performed. Error code 0x02 is saved in the RTR.PCTRLCFG.EC field. No reply is sent.
0	1	0	1	Invalid	No operation performed. Error code 0x02 is saved in the RTR.PCTRLCFG.EC field. No reply is sent.
0	1	1	0	Invalid	No operation performed. Reply is sent with error code 0x02. Error code is also saved in the RTR.PCTRLCFG.EC field.
0	1	1	1	Read-modify-write incrementing address	Read-modify-write operation performed if the requirements in section 33.5.1.1 are met.
1	0	0	0	Write, single address, don't verify before writing, no reply	No operation performed. Error code 0x0A is saved in the RTR.PCTRLCFG.EC field. No reply sent.

Table 404. RMAP command decoding and handling.

Bit 5	Bit 4	Bit 3	Bit 2		
Write / Read	Verify Data Before Write	Reply	Incre-ment Addr	Function	Action taken
1	0	0	1	Write, incre-menting address, don't verify before writing, no reply	No operation performed. Error code 0x0A is saved in the RTR.PCTRLCFG.EC field. No reply sent.
1	0	1	0	Write, single address, don't verify before write, send reply	No operation performed. Reply is sent with error code 0x0A. Error code is also saved in the RTR.PCTRLCFG.EC field.
1	0	1	1	Write, incre-menting address, don't verify before write, send reply	No operation performed. Reply is sent with error code 0x0A. Error code is also saved in the RTR.PCTRLCFG.EC field.
1	1	0	0	Write, single address, verify before writing, no reply	Write operation performed if the requirements in section 33.5.1.1 are met.
1	1	0	1	Write, incre-menting address, verify before writing, no reply	Write operation performed if the requirements in section 33.5.1.1 are met.
1	1	1	0	Write, single address, verify before writing, send reply	Write operation performed if the requirements in section 33.5.1.1 are met.
1	1	1	1	Write, incre-menting address, verify before writing, send reply	Write operation performed if the requirements in section 33.5.1.1 are met.

33.5.1.3 Access control

After reset / power-up the configuration port's address space can be accessed from all the ports. Configuration port accesses can be individually disabled per port by clearing the corresponding RTR.PCTRL.CE bit. Write commands, and read-modify-write commands to the configuration area can be globally disabled by writing a 0 to the RTR.CFGWE.WE bit.

There is also a CFGLOCK pin which can be used to disable configuration accesses from all ports except port 1 and 2. This signal overrides the setting of the RTR.PCTRL.CE for all ports, enabling configuration port accesses for port 1 and 2, and disabling them for all other ports. The RTR.CFGWE register still affects ports 1 and 2 in this case.

When a correct RMAP command is received but not allowed due to one or more of the access control features being enabled, a reply with Status field set to 0x0A (Authorization failure) is sent (if requested), and the RTR.PSTSCFG.EC field is updated to reflect the error. If a reply is not requested, the RTR.PSTSCFG.EC field is still set. In both cases, the operation is not performed.

33.5.1.4 RMAP Error handling

Table 405 shows the order in which errors in an RMAP command are detected. As soon as an error is detected, the command is discarded. If a reply should be sent, to a command that included an error, the reply is sent as soon as possible after the error is detected. This means that the reply might be sent out before the complete incoming RMAP command has been received. Note that since the complete RMAP command is buffered before it is executed, a command that contains an error is never executed.

*Table 405.*RMAP target error detection order

Detection Order	Error type	RMAP error code	Action taken
1	Wrong Protocol Identifier	N/A	The RTR.PSTSCFG.PT bit is set in order to indicate that the error occurred. No reply is sent.
2	EOP / EEP before completed header	N/A	The RTR.PSTSCFG.EO / RTR.PSTSCFG.EE bit is set in order to indicate that the error occurred. No reply is sent.
3	Header CRC error	N/A	The RTR.PSTSCFG.HC bit is set in order to indicate that the error occurred. No reply is sent.
4	Unused RMAP packet type	N/A	If the packet type (bit 7:6 of the packet's Instruction field) is "10" or "11" then the bit RTR.PSTSCFG.PT is set. For the value "00" (indicating a reply), no bit in RTR.PSTSCFG is set, since the RMAP standard [RMAP] does not specify that such an event should be recorded.
5	EEP immediately after header	N/A	The RTR.PSTSCFG.EE bit is set in order to indicate that the error occurred. No reply is sent.
6	Unused RMAP command code	0x02	RMAP error code is saved in the RTR.PCTRLCFG.EC field. Reply is sent if the Reply bit in the command's Instruction field was set to 1.
7	Invalid Target Logical Address	0x0C	RMAP error code is saved in the RTR.PCTRLCFG.EC field. Reply is sent if the Reply bit in the command's Instruction field was set to 1.
8	Invalid Key	0x03	RMAP error code is saved in the RTR.PCTRLCFG.EC field. Reply is sent if the Reply bit in the command's Instruction field was set to 1.
9	Verify buffer overrun	0x09	RMAP error code is saved in the RTR.PCTRLCFG.EC field. Reply is sent if the Reply bit in the command's Instruction field was set to 1.
10	RMW data length error	0x0B	RMAP error code is saved in the RTR.PCTRLCFG.EC field. Reply is sent if the Reply bit in the command's Instruction field was set to 1.
11	RMAP command not implemented or not authorized.	0x0A	RMAP error code is saved in the RTR.PCTRLCFG.EC field. Reply is sent if the Reply bit in the command's Instruction field was set to 1.
12	Early EOP / early EEP (not immediately after header)	0x05 / 0x07	RMAP error code is saved in the RTR.PCTRLCFG.EC field. Reply is sent if the Reply bit in the command's Instruction field was set to 1.
13	Invalid Data CRC	0x04	RMAP error code is saved in the RTR.PCTRLCFG.EC field. Reply is sent if the Reply bit in the command's Instruction field was set to 1.
14	EEP	0x07	RMAP error code is saved in the RTR.PCTRLCFG.EC field. Reply is sent if the Reply bit in the command's Instruction field was set to 1.
15	Too much data	0x06	RMAP error code is saved in the RTR.PCTRLCFG.EC field. Reply is sent if the Reply bit in the command's Instruction field was set to 1.

Most of the errors listed in table 405 are errors that only occur in one specific way, and they are also explained in the RMAP standard [RMAP]. Authorization failure (error code 0x0A) is however an exception. All the cases that lead to an authorization failure are listed below:

- A read command's Data Length field exceed 128 B.

- A command's (read, write, or read-modify-write) Address field does not contain a 4 B aligned address.
- The access control features described in section 33.5.1.3 prevented the port from accessing the RMAP target.
- The Address field of a command (read, write, or read-modify-write) contains an address that is outside of the configuration port's memory space.
- The Address field of the command (read, write, or read-modify-write) combined with the Length field would generate an access outside of the configuration port's memory space.
- The Length field of a command (read, write, or rea-modify write) is not a multiple of 4.
- A non-verified write command was received.

33.5.2 AMBA AHB slave interface

The configuration port provides an AMBA AHB slave interface, which makes the whole configuration port's address space accessible from the AHB bus. The address offsets are the same as when accessing through RMAP but the base address is different.

The routing table is shared between the ports, RMAP target and AHB slave, so accesses from the AHB slave might be stalled because of accesses from the other sources. The priority order when accessing the routing table, starting from the highest, is: router ports, AHB slave, RMAP target. Note that since the AHB slave has higher priority than the RMAP target, it is possible to read and write to the configuration port's registers in the middle of an RMAP write command. This needs to be considered in order to avoid a mismatch between the expected written value and actual written value.

None of the access control mechanisms mentioned in section 33.5.1.3 have any effect on the AHB slave interface.

33.6 Registers

The configuration port's registers listed in this section can be accessed either through the RMAP target, or the AMBA AHB slave interface. The RMAP addresses specified in table 406. The AHB addresses are determined by adding the RAMP addresses table 406 to the AHB base address for the controller in section 2.10. Registers that exist in several identical copies, corresponding to different addresses or ports, for example the RTR.RTPMAP registers, are only described once.

Only 32-bit single-accesses to the registers through AHB are supported.

*Table 406.*GRSPWROUTER registers

RMAP address	Register name	Acronym
0x00000000	RESERVED *	
0x00000004 - 0x0000000C	Routing table port mapping, physical addresses 1-3	RTR.RTPMAP
0x00000080 - 0x000003FC	Routing table port mapping, logical addresses 32-255	RTR.RTPMAP
0x00000400	RESERVED *	
0x00000404 - 0x0000040C	Routing table address control, physical addresses 1-3	RTR.RTACTRL
0x00000480 - 0x000007FC	Routing table address control, logical addresses 32-255	RTR.RTACTRL
0x00000800	Port control, port 0 (configuration port)	RTR.PCTRLCFG
0x00000804 - 0x0000080C	Port control, port 1-3	RTR.PCTRL
0x00000880	Port status, port 0 (configuration port)	RTR.PSTSCFG
0x00000884 - 0x0000088C	Port status, ports 1-3	RTR.PSTS
0x00000900 - 0x0000090C	Port timer reload, ports 0-3	RTR.PTIMER
0x00000980	Port control 2, ports 0 (configuration port)	RTR.PCTRL2CFG
0x00000984 - 0x0000098C	Port control 2, ports 1-3	RTR.PCTRL2
0x00000A00	Router configuration / status	RTR.RTRCFG
0x00000A04	Time-code	RTR.TC
0x00000A08	Version / instance ID	RTR.VER
0x00000A0C	Initialization divisor	RTR.IDIV
0x00000A10	Configuration write enable	RTR.CFGWE
0x00000A14	Timer prescaler reload	RTR.PRESCALER
0x00000A18	Interrupt mask	RTR.IMASK
0x00000A1C	Interrupt port mask	RTR.IPMASK
0x00000A20	Port interrupt pending	RTR.PIP
0x00000A24	Interrupt code generation	RTR.ICODEGEN
0x00000A28	Interrupt code distribution ISR, interrupt 0-31	RTR.ISR0
0x00000A2C	Interrupt code distribution ISR, interrupt 32-63	RTR.ISR1
0x00000A30	Interrupt code distribution ISR timer reload	RTR.ISRTIMER
0x00000A34	Interrupt code distribution ACK-to-INT timer reload	RTR.AITIMER
0x00000A38	Interrupt code distribution ISR change timer reload	RTR.ISRCTIMER
0x00000A3C	RESERVED	
0x00000A40	SpaceWire link running status	RTR.LRUNSTS

Table 406. GRSPWROUTER registers

RMAP address	Register name	Acronym
0x00000A44	Capability	RTR.CAP
0x00000A48 - 0x00000A4C	RESERVED	
0x00000A50	SpaceWire Plug-and-Play - Device Vendor and Product ID	RTR.PNPVEND
0x00000A54	SpaceWire Plug-and-Play - Unit Vendor and Product ID	RTR.PNPUVEND
0x00000A58	SpaceWire Plug-and-Play - Unit Serial Number	RTR.PNPUSN
0x00000A5C - 0x00000C0C	RESERVED	
0x00000C10, 0x00000C20	Outgoing character counter, ports [1, 2]	RTR.OCHARCNT
0x00000C14, 0x00000C24	Incoming character counter, ports [1, 2]	RTR.ICHARCNT
0x00000C18, 0x00000C28	Outgoing packet counter, ports [1, 2]	RTR.OPKTCNT
0x00000C1C, 0x00000C2C	Incoming packet counter, ports [1, 2]	RTR.IPKTCNT
0x00000D40 - 0x00000DFC	RESERVED	
0x00000E00 - 0x00000E0C	Maximum packet length, ports 0-3	RTR.MAXPLEN
0x00000E84 - 0x00000E88	Credit counter, ports 1-2	RTR.CREDCNT
0x00000F00	RESERVED	
0x00000F04	RESERVED	
0x00000F08 - 0x00000F0C	RESERVED	
0x00000F10	RESERVED	
0x00000F14 - 0x00000FFC	RESERVED	
0x00001000	RESERVED**	
0x00001004 - 0x0000100C	Routing table, combined port mapping and address control, addresses 1-3	RTR.RTCOMB
0x00001080 - 0x000013FC	RESERVED**	
0x00001400 - 0x00001FFC	RESERVED	
0x00002000 - 0x00002FFC	APB address area	RTR.APBAREA

* Physical address 0 (configuration port), and non existing ports 4-31 does not have an RTR.RTPMAP or RTR.RTACTRL register, and are therefore RESERVED.

** Physical address 0 (configuration port), and non existing 4-31 ports does not have an RTR.RTCOMB register, and are therefore RESERVED.

Table 407. 0x00000004-0x0000000C, 0x00000080-0x000003FC - RTR.RTPMAP - Routing table port mapping, addresses 1-3 and 32-255

31 4	3	2	1	0
RESERVED	PE [1) 2) 3) 4)]			PD
0x000	*			0
r	rw*			rw

31: 20 RESERVED

3: 1 Port enable bits (PE) - When set to 1, each bit enables packets with the physical / logical address corresponding to this RTR.RTPMAP register to be sent on the port with the same number as the bit index. For physical addresses, the bit index corresponding to the port with the same number as the physical address itself is always 1 (not possible to write to 0). For logical addresses, at least one bit must be set to 1 in order for a packet with the corresponding address to be routed; otherwise the packet is spilled and an invalid address error generated. Reset value for physical addresses is all zeroes (except for the bit index corresponding to the port with the same number as the address). Reset value for logical addresses is zero.

0 Packet distribution (PD) - When set to 1, packet distribution is used for the physical / logical address corresponding to this RTR.RTPMAP register. When set to 0, group adaptive routing is used. See section 33.2.4 and 33.2.5 for more information.

Note 1 GR716B can only be configured to have 1 logical address. Logical address in range 32 to 255 for GR716B is set via PE bits at selected address routing table position. E.g. to route packets with logical addresses of 64 to the AMBA port of the GR716B device application must set the PE bit in RTR.RTPMAP at the offset 0x160

Note 2 Setting PE bit for a specific logical address will automatically set PE = 0x1 for all lower logical addresses

Note 3 Setting PE bit for a specific logical address will automatically set PE = 0x2 for all higher logical addresses

Note 4 Selected logical address must be deselected before any new logical address can be set. Deselect current logical address by writing 0x0 to selected address routing table position.

Table 408. 0x00000404-0x0000040C, 0x00000480-0x000007FC - RTR.RTACTRL - Routing table address control, addresses 1-3 and 32-255

31 4	3	2	1	0
RESERVED	SR	EN	PR	HD
0x0000000	*	*	*	*
r	rw	rw	rw	rw

31: 4 RESERVED

3 Spill-if-not-ready (SR) - When set to 1, an incoming packet with the corresponding physical / logical address is immediately spilled if the selected output port's link interface is not in run-state. If packet distribution is used for the incoming packet, and this bit is set, the packet is spilled unless all output ports' link interfaces are in run state. For physical addresses, this bit is double mapped in the RTR.PCTRL.SR field. Reset value for physical addresses are taken from the SPILLIFNOTREADY pin. Reset value for logical addresses is N/R.

2 Enable (EN) - Enables the routing table address control entry. Address control entries for physical addresses are always enabled, and this field is constant 1. For logical addresses, this bit must be set to 1 in order for packets with the corresponding logical address to be routed. Reset value for logical addresses is 0.

1 Priority (PR) - Sets the arbitration priority of this physical / logical address. 0 = low priority, 1 = high priority. Used when more than one packet is competing for the same output port. For physical addresses, this bit is double mapped in the RTR.PCTRL.PR field. Reset value for physical addresses is 0. Reset value for logical addresses is N/R.

0 Header deletion (HD) - Enables / disabled header deletion for the corresponding logical address. For physical addresses, header deletion is always enabled, and this bit is constant 1. Reset value for logical addresses is N/R.

Note 1 Select logical address can only be configured after selected routing table position in RTR.RTPMAP has been set.

Note 2 Setting table address control bits for any logical address below selected routing table position in RTR.RTPMAP will affect all lower logical addresses

Note 3: Setting table address control bits for any logical address above selected routing table position in RTR.RTPMAP will affect all higher logical addresses

Note 4: logical address must be deselected in RTR.RTPMAP before new configuration of RTR.RTACTRL can be done

Table 409. 0x00000800 - RTR.PCTRLCFG - Port control, port 0 (configuration port)

31 18	17	16	15 10	9	8 0
RESERVED	PL	TS	RESERVED	TR	RESERVED
0x0000	0	0	0x00	*	0x000
r	rw	rw	r	rw	r

31: 18 RESERVED

17 Packet length truncation (PL) - When set to 1, an RMAP / SpaceWire Plug-and-Play reply is spilled, and an EEP written to the transmit FIFO of the output port, if the total length of the reply packet exceeds the maximum length specified in the RTR.MAXPLEN register for port 0. See section 33.2.14 for more information on packet length truncation.

16 Time-code / distributed interrupt code truncation (TS) - When set to 1, an RMAP / SpaceWire Plug-and-Play reply is spilled, and an EEP written to the transmit FIFO of the output port, if a valid time-code / distributed interrupt code is received and if the code matches the codes selected by the RTR.PCTRL2CFG.SV and RTR.PCTRL2CFG.SM fields. See section 33.2.18 for more information.

15: 10 RESERVED

9 Timer enable (TR) - Enable data character timer for port 0. See section 33.2.13 for details. Reset value set from TIMEEN signal.

8: 0 RESERVED

Table 410. 0x00000804-0x0000080C - RTR.PCTRL - Port control, ports 1-3 SpaceWire ports

31 30 29 24	23 22	21	20	19	18	17	16	15	14 11	10	9	8	7	6	5	4	3	2	1	0
RD	RES	ST	SR	AD	LR	PL	TS	IC	ET	RESERVED	DI	TR	PR	TF	RS	TE	R	CE	AS	LS	LD
0x0 *	0x0	0	*	*	*	0	0	*	0	0x0	*	*	0	0	0	1	0	*	1	0	0
rw	r	rw	rw	rw	rw	rw	rw	rw	rw	r	rw	rw	rw	rw	rw	rw	r	rw	rw	rw	rw

31: 24 Run-state clock divisor (RD) - Clock divisor value used for the corresponding port's link interface when in run-state. Field is only available for the SpaceWire ports. Bits 31:30 have reset value 0x0, while bits 29:24 get their reset value from the IDIVISOR[7:0] pins. For more information about setting the link-rate for SpaceWire ports during run-state see section 33.2.21.

23: 22 RESERVED

21 Static routing enable (ST) - When set to 1, incoming packets on this port are routed based on the physical address specified in the corresponding RTR.PCTRL2.SD field, and the setting of the corresponding RTR.PCTRL2.SC bit, instead of the packet's first byte. Header deletion is not used when static routing is enabled, which means that the first byte of the packet is always sent as well. This bit can only be set to 1 if the RTR.RTRCFG.SR bit is set to 1. Note that when this bit is set to 1 it is not possible to access the configuration port from this port.

20 Spill-if-not-ready (SR) - This bit is double mapping of the RTR.RTACTRL.SR bit. See table 408.

19 Auto-disconnect (AD) - When set to 1, the auto-disconnect feature described in section 33.2.12 is enabled. Reset value taken from the AUTODCONNECT pin. This bit is only available for the SpaceWire ports.

18 Link-start-on-request (LR) - When set to 1, the link-start-on-request feature described in section 33.2.11 is enabled. Reset value taken from the LINKSTARTONREQ pin. This bit is only available for the SpaceWire ports.

17 Packet length truncation (PL) - When set to 1, packets for which this port is the input port will be spilled, and an EEP written to the transmit FIFO of the output port(s), if the packets exceed the maximum length specified in the corresponding RTR.MAXPLEN register. See section 33.2.14 for more information on packet length truncation.

16 Time-code / distributed interrupt code truncation (TS) - When set to 1, packets for which this port is the input port will be spilled, and an EEP written to the transmit FIFO of the output port(s), if a valid time-code / distributed interrupt code is received, and if the code also matches the codes selected by the RTR.PCTRL2.SV and RTR.PCTRL2.SM fields. See section 33.2.18 for more information.

Table 410. 0x00000804-0x0000080C - RTR.PCTRL - Port control, ports 1-3 SpaceWire ports

15	Distributed interrupt code enable (IC) - When set to 0, all incoming distributed interrupt codes on this port are discarded, and no distributed interrupt codes are sent out on the port. When set to 1, the four bits RTR.PCTRL2.IR, RTR.PCTRL2.IT, RTR.PCTRL2.AR, RTR.PCTRL2.AT are used to enable / disable distributed interrupt code transmit and receive. Note that the global distributed interrupt code enable bit, RTRTCFG.IE, also must be set to 1 for distributed interrupt codes to be sent / received. See section 33.2.16 for a description of distributed interrupts. Reset value set from INTERRUPTFWD signal.
14	Enable external time (ET) - When a time-code is received on the port and this bit is set to 0, the router discards the received time-code value and instead increments its internal time-counter value (RTR.TC.TC), and forwards a time-code with the new value to the other ports. If this bit is set to 1 when the time-code is received, the time-code is processed according to the rules described in section 33.2.15. This bit is only available for the SIST port.
13: 11	RESERVED
10	Disable port (DI) - When set to 1, data transfers to and from this port are disabled. See section 33.2.6 for details. Reset value is 0 for SpaceWire ports, and 1 for SIST port. NOTE: If this bit is set in combination with the RTR.PCTRL.LD bit, the link interface for this port will be clock gated, and the LVDS drivers for this port will be powered-down.
9	Packet timer enable (TR) - Enable the data character timer for incoming packets. See section 33.2.13 for details. Reset value set from TIMEREN signal.
8	Priority (PR) - This bit is double mapping of the RTR.RTACTRL.SR bit. See table 408.
7	Transmit FIFO reset (TF) - Resets the transmit FIFO on this port. This means that the FIFO is emptied (counters and pointers set to 0), and an EEP is written to the FIFO to ensure that any incomplete packet is detected by the receiver. If a packet transmission is active (another port is using this port as output port) when this bit is set, the remainder of that packet will be spilled before the EEP is inserted. This bit is self-clearing, and should not be written with 0 while it is 1, since that could abort the ongoing transmit FIFO reset.
6	Receive FIFO spill (RS) - Spills the receive FIFO for this port, meaning that the packet currently being received is spilled. The output port(s) used for the packet will have an EEP written to the transmit FIFO to indicate that the packet was ended prematurely. If no packet is received, setting this bit has no effect. This bit is self-clearing, and should not be written with 0 while it is 1, since that could abort the ongoing receive FIFO spill.
5	Time-code enable (TE) - Enables time-codes to be received and transmitted on this port. When set to 1, received time-codes are processed according to the rules described in section 33.2.15. If this bit is set to 0, all received time-codes on this port are ignored.
4	RESERVED
3	Configuration port access enable (CE) - Enable accesses to the configuration port from this port. If set to 0, incoming packets with physical address 0 will be spilled. Reset value is 1 for SpaceWire ports, and 0 for SIST port.
2	Autostart (AS) - Enable the link interface FSM's autostart feature, as defined in ECSS-E-ST-50-12C [SPW]. This bit is only available for the SpaceWire ports.
1	Link start (LS) - Start the link interface FSM. This bit is only available for the SpaceWire ports.
0	Link disabled (LD) - Disable the link interface FSM. This bit is only available for the SpaceWire ports. NOTE: When this bit is set to 1, the LVDS driver for this SpaceWire port will be powered down. If this bit is set in combination with the RTR.PCTR.DI bit, the link interface will be clock gated.

Table 411. 0x00000880 - RTR.PSTSCFG - Port status, port 0 (configuration port)

31	30	29	28	27	26	25	24	23 20	19	18	17 12	11 7	6 5	4	3 0
EO	EE	PL	TT	PT	HC	PI	CE	EC	R	TS	RESERVED	IP	RES	CP	PC
0	0	0	0	0	0	0	0	0x0	0	0	0x00	0x0	0x0		0x0
wc	wc	wc	wc	wc	wc	wc	rw*	r	r	wc	r	r	r	rw*	r

31 Early EOP (EO) - Set to 1 when an RMAP / SpaceWire Plug-and-Play command with an early EOP was received by the configuration port. See section 33.5.1.4 for error detection order.

30 Early EEP (EE) - Set to one when an RMAP / SpaceWire Plug-and-Play command with an early EEP was received by the configuration port. See section 33.5.1.4 for error detection order.

29 Packet length truncation (PL) - Set to 1 when an RMAP / SpaceWire Plug-and-Play reply packet has been spilled due to a maximum length violation. See section 33.2.14 for details.

28 Time-code / distributed interrupt code tick truncation (TT) - Set to one when an RMAP / SpaceWire Plug-and-Play reply packet has been spilled due to a time-code / distributed interrupt code. See section 33.2.18 for details.

27 Packet type error (PT) - Set to one if an RMAP / SpaceWire Plug-and-Play packet with correct header CRC, but with the packet type bits set to the reserved values “10” or “11”, was received by the configuration port. See section 33.5.1.4 for error detection order.

26 Header CRC Error (HC) - Set to one if a Header CRC error is detected in an RMAP / SpaceWire Plug-and-Play command received by the configuration port. See section 33.5.1.4 for error detection order.

25 Protocol ID Error (PI) - Set to one if a packet received by the configuration port had the wrong protocol ID. Supported protocol ID:s are 0x01 (RMAP), and 0x03 (SpaceWire Plug-and-Play). See section 33.5.1.4 for error detection order.

24 Clear error code (CE) - Write with a 1 to clear the RTR.PCTRLCFG.EC field. This bit is self clearing and always reads 0. Writing 0 has no effect.

23: 20 Error code (EC) - Shows the four least significant bits of the latest non-zero RMAP status code. If zero, no error has occurred.

19 RESERVED

18 Timeout spill (TS) - Set to one when an RMAP reply was spilled due to a packet timeout. See section 33.2.13 for details.

17: 12 RESERVED

11: 7 Input port (IP) - The number of the last port from which a packet was routed to the configuration port. This field is updated even if an operation is not performed, for example due to an incorrect RMAP packet.

6: 5 RESERVED

4 Clear SpaceWire Plug-and-Play error code (CP) - Write with a 1 to clear the RTR.PCTRLCFG.PC field. This bit is self clearing and always reads 0. Writing 0 has no effect.

3: 0 SpaceWire Plug-and-Play Error code (PC) - Shows the four least significant bits of the latest non-zero SpaceWire Plug-and-Play status code. If zero, no error has occurred.

Table 412. 0x00000884-0x0000088C - RTR.PSTS - Port status, ports 1-3

31 30	29	28	27	26	25 23	22	21	20	19	18	17	16	15	14 12	11 7	6	5	4	3	2	1	0
PT	PL	TT	RS	SR	RESERVED	LR	SP	AC	R	TS	R	TF	RE	LS	IP	PR	PB	IA	CE	ER	DE	PE
*	0	0	0	0	0x0	0	0	0	0	0	0	0	1	000	00000	0	0	0	0	0	0	0
r	wc	wc	wc	wc	r	r	r	r	r	wc	r	r	r	r	r	r	r	wc	wc	wc	wc	wc

31: 30 Port type (PT) - The type of this port. Constant value of “00” for the SpaceWire ports, and constant value of “11” for the SIST port.

29 Packet length truncation (PL) - Set to 1 when a packet for which this port was the input port has been spilled due to the packet length truncation feature. See section 33.2.14 for details.

28 Time-code / distributed interrupt code tick truncation (TT) - Set to 1 when a packet for which this port was the input port has been spilled due to the time-code / distributed interrupt code truncation feature. See section 33.2.18 for details.

Table 412. 0x00000884-0x0000088C - RTR.PSTS - Port status, ports 1-3

27	RMAP / SpaceWire Plug-and-Play spill (RS) - Set to 1 when an RMAP / SpaceWire Plug-and-Play command received on this port was spilled by the configuration port.
26	Spill-if-not-ready spill (SR) - Set to 1 when a packet received on this port was spilled due to the spill-if-not-ready feature. See section 33.2.8.
25: 23	RESERVED
22	Link-start-on-request status (LR) - Set to 1 when this port either was started, or currently is trying to start, due to the link-start-on-request feature, described in section 33.2.11. This bit is only available for the SpaceWire ports.
21	Spill status (SP) - This bit is 1 when a packet that is incoming on this port currently is being spilled. Otherwise, this bit is 0.
20	Active status (AC) - Set to 1 when a packet arrives at this port and the port has been given access to the routing table. Cleared when the packet has been transmitted or spilled.
19	RESERVED
18	Timeout spill (TS) - Set to 1 when a packet for which this port was the input port was spilled due to a packet timeout. See section 33.2.13 for details.
17	RESERVED
16	Transmit FIFO full (TF) - Set to 1 when the transmit FIFO on this port is full.
15	Receive FIFO empty (RE) - Set to 1 when the receive FIFO on this port is empty.
14: 12	Link state (LS) - Current link state. 000 = Error reset. 001 = Error wait, 010 = Ready, 011 = Started, 100 = Connecting, 101 = Run state. This field is only available for the SpaceWire ports.
11: 7	Input port (IP) - This field shows the number of the input port for either the currently ongoing packet transfer on this port (if RTR.PSTS.PB = 1), or for the last packet transfer on this port (if RTR.PSTS.PB = 0).
6	Port receive busy (PR) - Set to 1 when this port is the input port of an ongoing packet transfer.
5	Port transmit busy (PB) - Set to 1 when this port is the output port of an ongoing packet transfer.
4	Invalid address (IA) - Set to 1 when an invalid address error occurred on this port. See section 33.2.10 for details.
3	Credit error (CE) - Set to 1 when a credit error has occurred. This bit is only available for the SpaceWire ports.
2	Escape error (ER) - Set to 1 when an escape error has occurred. This bit is only available for the SpaceWire ports.
1	Disconnect error (DE) - Set to 1 when a disconnect error has occurred. This bit is only available for the SpaceWire ports.
0	Parity error (PE) - Set to 1 when a parity error has occurred on. This bit is only available for the SpaceWire ports.

Table 413. 0x00000900-0x0000090C - RTR.PTIMER - Port timer reload, ports 0-3

31	10	9	0
RESERVED		RL	
*			
rw*			

31: 10	RESERVED
9: 0	Timer reload (RL) - Port timer reload value, counted in prescaler ticks. This value is used to reload the corresponding port timer used for packet transfer timeouts, and auto-disconnect. The minimum value of this field is 1. Trying to write 0 will result in 1 being written. Reset value set from RELOADN[31:0] signal.

Table 414. 0x00000980 - RTR.PCTRL2CFG - Port control 2, port 0 (configuration port)

31 24	23 16	15	14 13 12 11 10 9 8 6 5 1	0
SM	SV	OR	RESERVED	
0xC0	0x00	1	0x0000	
rw	rw	rw	r	

31: 24 Time-code / distributed interrupt code truncation mask (SM) - Defines which bits of a time-code / distributed interrupt code that must match the value specified in RTR.PCTRL2CFG.SV in order for an RMAP / SpaceWire Plug-and-Play reply packet to be spilled. If a bit in this field is set to 1, the corresponding bit in RTR.PCTRL2.SV must match the time-code / distributed interrupt code. If a bit in this field is set to 0, the corresponding bit in RTR.PCTRL2.SV does not have to match the time-code / distributed interrupt code.

23: 16 Time-code / distributed interrupt code truncation value (SV) - Defines the value to use together with the RTR.PCTRL2CFG.SM field when checking if a received time-code / distributed interrupt code should spill an ongoing RMAP / SpaceWire Plug-and-Play reply.

15 Overrun timeout enable (OR) - Enables spilling due to overrun timeouts for RMAP / SpaceWire Plug-and-Play replies. See section 33.2.13 for details.

14: 0 RESERVED

Table 415. 0x00000984-0x0000098C - RTR.PCTRL2 - Port control 2, ports 1-3

31 24	23 16	15	14	13	12	11	10	9	8 6	5 1	0
SM	SV	OR	UR	R	AT	AR	IT	IR	RESERVED	SD	SC
0xC0	0x00	1	1	0	1	1	1	1	0x0	0x00	0
rw	rw	rw	rw	r	rw	rw	rw	rw	r	rw	rw

31: 24 Time-code / distributed interrupt code truncation mask (SM) - Defines which bits of a time-code / distributed interrupt code that must match the value specified in RTR.PCTRL2.SV in order for a packet, for which this port is the input port, to be spilled. If a bit in this field is set to 1, the corresponding bit in RTR.PCTRL2.SV must match the time-code / distributed interrupt code. If a bit in this field is set to 0, the corresponding bit in RTR.PCTRL2.SV does not have to match the time-code / distributed interrupt code.

23: 16 Time-code / distributed interrupt code truncation value (SV) - Defines the value to use together with the RTR.PCTRL2.SM field when checking if a time-code / distributed interrupt code should spill a packet for which this port is the input port.

15 Overrun timeout enable (OR) - Enables spilling due to overrun timeouts for packets for which this port is the input port. See section 33.2.13 for details.

14 Underrun timeout enable (UR) - Enables spilling due to unerrun timeouts for packets for which this port is the input port. See section 33.2.13 for details.

13 RESERVED

12 Interrupt acknowledgement code / extended interrupt code transmit enable (AT) - Enables the transmission of interrupt acknowledgement codes / extended interrupt codes on this port. If set to 0, no interrupt acknowledgement codes / extended interrupt codes will be forwarded to this port.

11 Interrupt acknowledgement code / extended interrupt code receive enable (AR) - Enabled the reception of interrupt acknowledgement codes / extended interrupt codes on this port. If set to 0, all received interrupt acknowledgement codes / extended interrupt codes on this port will be silently discarded.

10 Interrupt code transmit enable (IT) - Enables the transmission of interrupt codes on this port. If set to 0, no interrupt codes will be forwarded to this port.

9 Interrupt code receive enable (IR) - Enabled the reception of interrupt codes on this port. If set to 0, all received interrupt codes on this port will be silently discarded.

8: 6 RESERVED

5: 1 Static route destination (SD) - When RTR.PCTRL.ST is set to 1, incoming packets on this port will be routed based on the value of this field, and the setting of RTR.PCTRL2.SC, instead of the packet's first byte.

0 Static route configuration (SC) - When this bit is set to 1, the RTR.RTPMAP register corresponding to the physical address specified by the RTR.PCTRL2.SD field will be used when routing packets, if RTR.PCTRL.ST is set to 1.

Table 416. 0x00000A00 - RTR.RTRCFG - Router configuration / status

31 27	26 22	21 17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SP	RESERVED	FP	R	SR	PE	IC	IS	IP	AI	AT	IE	RE	EE	AA	SA	TF	R	TA	PP
0x12	0x00	0x01	0	*	*	0	0	0	*	1	*	0	*	*	1	*	0	1	1
r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	r	rw	rw	r	r	r

31: 27 SpaceWire ports (SP) - Set to the number of SpaceWire ports in the router. Constant value of 0x12.

26: 22 RESERVED

21: 17 FIFO ports (FP) - Set to the number of FIFO ports. Constant value of 0x01.

16 RESERVED

15 Static routing enable (SR) - This read-only bit specifies if the router's static routing feature is enabled (1) or disabled (0). See section 33.2.7 for details. The value is set from the STATICROUTEEN signal at reset.

14 SpaceWire Plug-and-Play enable (PE) - This read-only bit specifies if the router's SpaceWire Plug-and-Play features are enabled (1) or disabled (0). See section 33.6.1 for details. The value is set from the PNPEN pin at reset.

13 ISR change timer enable (IC) - If set to 1, the router will wait for the time period specified by the RTR.IRCTIMER register after an ISR bit change value, before it allows an incoming distributed interrupt code to change the value of the same ISR bit. If set to 0, the ISR change timers are not used, and an ISR bit is allowed to change value again as soon as the previous distributed interrupt code has been distributed.

12 Distributed interrupt code selection routine (IS) - If set to 0, the router uses round-robin on the interrupt numbers when deciding which distributed interrupt code to distribute next. If set to 1, the router gives priority to lower interrupt numbers when deciding which distributed interrupt code to distribute. See section 33.2.16.1.

11 Distributed interrupt code priority (IP) - When set to 0, all interrupt codes have priority over all interrupt acknowledgement codes / extended interrupt codes, and will be distributed first. When set to 1, all interrupt acknowledgement codes have priority over all interrupt codes. See section 33.2.16.1.

10 Auxiliary distributed interrupt codes enable (AI) - If set to 1, distributed interrupt codes can be sent and received on the auxiliary time-code / distributed interrupt code interface. If set to 0, all distributed interrupt codes received on the auxiliary interface are silently discarded, and no distributed interrupt codes will be transmitted on the interface. Reset value set from INTERRUPTFWD signal.

9 Auxiliary time-code enable (AT) - If set to 1, time-codes can be sent and received on the auxiliary time-code / distributed interrupt code interface. If set to 0, all time-codes received on the auxiliary interface are silently discarded, and no time-codes will be transmitted on the interface.

8 Distributed interrupt codes enable (IE) - Global enable/disable for distributed interrupt codes. If set to 0, all received distributed interrupt codes will either be silently discarded (if RTRCFG.TF = 1), or handled as time-codes (if RTRCFG.TF = 0). When set to 1, whether or not distributed interrupt codes are received or transmitted on a port depends on the setting of the register bits RTR.PCTRL.IC, RTR.PCTRL2.IR, RTR.PCTRL2.IT, RTR.PCTRL2.AR, and RTR.PCTRL2.AT. Reset value taken from INTERRUPTCODEEN signal.

7 Reset (RE) - Resets the complete router when written with a 1. When this bit is written through RMAP, an RMAP reply will not be sent, even if the reply bit in the RMAP commands Instruction field is set to 1. This bit is self-clearing.

6 Enable extended distributed interrupts (EE) - If set to 0, all distributed interrupt codes with bit 5 set to 1 are handled as interrupt acknowledgement codes. If set to 1, all distributed interrupt codes with bit 5 set to 1 are handled as extended interrupt code. Reset value taken from INTERRUPTMODE signal. See section 33.2.16.

5 Asynchronous auxiliary interface (AA) - This read-only bit specifies whether the inputs of the auxiliary time-code / distributed interrupt code interface, described in section 33.2.17, are handled as synchronous or asynchronous to CLK. A value of 0 means that the inputs are handles as synchronous to CLK, while 1 means asynchronous.

4 Self addressing enable (SA) - If set to 1, ports are allowed to send packets to themselves. If set to 0, packets with the same input port as output port are spilled, and an invalid address error is asserted for that port.

3 Time-code control flag mode (TF) - When set to 0, all received time-codes / distributed interrupt codes are handled as time-codes, no matter the value of the control flags (bits 7:6 of the code). When set to 1, the time-code control flags must have value "00" to be considered valid time-codes. Note that the RTRCFG.IE bit has priority over this bit, which means that if RTRCFG.IE is 1, then setting this bit to 0 has no impact. Reset value taken from TIMECODEFILT.

2 RESERVED

Table 416. 0x00000A00 - RTR.RTRCFG - Router configuration / status

1 Timers available (TA) - Constant value 1. Indicates that the router has support for timers, as described in section 33.2.13.

0 SpaceWire Plug and Play available (PP) - Constant value 1. Indicates that the router support SpaceWire Plug and Play, as described in section 33.6.1.

Table 417. 0x00000A04 - RTR.TC - Time-code

31 10	9	8	7 6	5 0
RESERVED	RE	EN	CF	TC
0x000000	0	*	0x0	0x00
r	rw*	rw	r	r

31: 10 RESERVED

9 Reset time-code (RE) - When this field is written to 1, the RTR.TC.CF and RTR.TC.TC fields are reset. This bit is self-clearing, and always reads 0. Writing 0 has no effect.

8 Enable time-codes (EN) - When set to 1, received time-codes are handled by the router according to the rules described in 33.2.15. When set to 0, all received time-codes are silently discarded. Reset value set through TIMECODEREGEN signal.

7: 6 Time-control flags (CF) - The current value of the router's time-code control flags (bits 7:6 of the latest valid time-code received).

5: 0 Time-counter (TC) - Current value of the router's time counter.

Table 418. 0x00000A08 - RTR.VER - Version / instance ID

31 24	23 16	15 8	7 4	3 0
MA	MI	PA	ID	
0x01	0x03	0x00	*	
r	r	r	rw	

31: 24 Major version (MA) - Holds the major version number of the router. Constant value 0x01.

23: 16 Minor version (MI) - Holds the minor version number of the router. Constant value 0x03.

15: 8 Patch (PA) - Holds the patch number of the router. Constant value 0x00.

7: 0 Instance ID (ID) - Holds the instance ID number of the router. Reset value is set through INSTANVEID[7:0] signal.

Table 419. 0x00000A0C - RTR.IDIV - Initialization divisor

31 8	7 0
RESERVED	ID
0x000000	*
r	rw

31: 8 RESERVED

7: 0 Initialization clock divisor (ID) - Clock divisor value used by all the SpaceWire links to generate the 10 Mbit/s rate during initialization. Reset value from the IDIVISOR[7:0] signal. For more information about setting the link-rate for SpaceWire ports during initialization see section 33.2.21.

Table 420. 0x00000A10 - RTR.CFGWE - Configuration port write enable

31	1	0
RESERVED		WE
0x00000000		1
r		rw

31: 1 RESERVED

0 Configuration port write enable (WE) - When set to 1, write accesses to the configuration port area are allowed. When set to 0, write accesses are only allowed to this register. RMAP write and RMAP read-modify-write commands will be replied to with the Status field set to 0x0A (authorization failure), if a reply was requested. The value of this bit has no effect for SpaceWire Plug-and-Play commands.

Table 421. 0x00000A14 - RTR.PRESCALER - Timer prescaler reload

31	16	15	0
RL			
*			
rw*			

31: 0 Timer prescaler reload (RL) - Global prescaler reload value used for generating a common tick for the data character timers, auto-disconnect timers, and distributed interrupt code timers. The prescaler runs on the system clock, and a tick is generated every RTR.PRESCALER.RL+1 CLK cycle. The minimum value of this field is 49. Trying to write a value less than that will result in 49 being written. Reset value is set through RELOAD[31:0] signal.

Table 422. 0x00000A18 - RTR.IMASK - Interrupt mask

31	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED		PE	SR	RS	TT	PL	TS	AC	RE	IA	LE	R
0x00000		0	0	0	0	0	0	0	0	0	0	0
r		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r

31: 11 RESERVED

10 SpaceWire Plug-and-Play error (PE) - Generate an interrupt when a SpaceWire Plug and Play error has been detected in the configuration port. The different errors are described in 33.6.1.

9 Spill-if-not-ready (SR) - Generate an interrupt when a packet has been spilled because of the spill-if-not-ready feature described in section 33.2.8.

8 Run-state entry (RS) - Generate an interrupt when a SpaceWire link enters run-state.

7 Time-code / distributed interruptcode tick truncation (TT) - Generate an interrupt when a packet has been spilled because of the time-code / distributed interrupt code truncation feature described in section 33.2.18.1.

6 Packet length truncation (PL) - Generate an interrupt when a packet has been spilled due to the packet length truncation feature described in section 33.2.14.

5 Timeout spill (TS) - Generate an interrupt when a packet has been spilled due to the timeout mechanism.

4 Auxiliary configuration port error (AC) - Generate an interrupt when either a header CRC error, protocol ID error, packet type error, early EOP, or early EEP has been detected in the configuration port.

3 RMAP error (RE) - Generate an interrupt when an error has been detected in the configuration port for an RMAP command such that the PSTS.EC field is set to a non-zero value.

2 Invalid address (IA)- Generate an interrupt when an invalid address error has occurred on a port. See RTR.PSTS:IA bit and section 33.2.10 for a definition of invalid address.

1 Link error (LE) - Generate an interrupt when a link error has been detected on a SpaceWire port.

0 RESERVED

Table 423. 0x00000A1C - RTR.IPMASK - Interrupt port mask

31 20	19 0
RESERVED	IE
0x000	0x00000
r	rw

31: 20 RESERVED

19: 0 Port interrupt enable (IE) - Set a bit to 1 to enable interrupts to be generated for an error detected in the port with the same number as the bit index. An interrupt is signaled through the IRQ pin, and optionally through a distributed interrupt code.

Table 424. 0x00000A20 - RTR.PIP - Port interrupt pending

31 20	19 0
RESERVED	IP
0x000	0x00000
r	wc

31: 20 RESERVED

19: 0 Interrupt pending (IP) - When a bit is set to 1, the port with the same number as the bit index was the source of an interrupt. A bit in this field will only be set to 1 for a generated interrupt if the port's corresponding bit in RTR.IPMASK is set, as well as the error types corresponding bit in RTR.IMASK, are set.

Table 425. 0x00000A24 - RTR.ICODEGEN - Interrupt code generation

31 21	20	19	18	17	16	15 6	5 0
RESERVED	UA	AH	IT	TE	EN	RESERVED	IN
0x000	0	0	0	1	0	0x000	0x00
r	rw	rw	rw	rw	rw	r	rw

0

31: 21 RESERVED

20 Interrupt code generation un-acknowledge mode (UA) - If this bit is set to 1, an ISR timeout for a distributed interrupt that was generated by the router will clear the bits in the RTR.PIP register that were set when the interrupt was generated. If this bit is set to 0, no extra handling is done on an ISR timeout event, and the bits in RTR.PIP will stay set. See section 33.2.16.

19 Interrupt acknowledgement code handling (AH) - When set to 1, and the router has generated an interrupt code, a received interrupt acknowledgement code with the interrupt number matching the RTR.ICODEGEN.IN field will clear the bits in the RTR.PIP register that were set when the interrupt code was generated. If set to 0, no extra handling of a received interrupt acknowledgement code is done and the bits in RTR.PIP will stay set. This bit is unused when the distributed interrupts are operating in the extended interrupt mode. See section 33.2.16.

18 Interrupt type (IT) - 0 = Level. 1 = Edge. When set to 0, a new interrupt code is distributed as long as RTR.PIP register is non zero. When set to 1, a new interrupt code is distributed only when a bit in RTR.PIP toggles from 0 to 1. See section 33.2.16.

Table 425. 0x00000A24 - RTR.ICODEGEN - Interrupt code generation

17	Interrupt acknowledgement code to interrupt code timer enable (TE) - If set to 1, the router will wait for the time period specified by the RTR.AITIMER register after the reception of an interrupt acknowledgement code (for which the router generated the corresponding interrupt code) until a new interrupt code is allowed to be generated. If set to 0, the timer is not used, and a new interrupt code is allowed to be generated as soon as the interrupt acknowledgement code has been distributed. This bit is unused when the distributed interrupts are operating in the extended interrupt mode.
16	Interrupt code generation enable (EN) - When 1, distributed interrupt code generation is enabled, and an interrupt code / extended interrupt code can be generated when an internal error event occurs. See section 33.2.16.
15: 6	RESERVED
5: 0	Interrupt number (IN) - Sets the interrupt number of the distributed interrupt code that will be generated when the interrupt code generation feature is enabled (RTR.ICODEGEN.EN = 1). Note that when the distributed interrupts are operating in interrupt with acknowledgements mode, this field must not be set to a value larger than 31, since that would specify an interrupt with acknowledgement code. See section 33.2.16.

Table 426. 0x00000A28 - RTR.ISR0 - Interrupt code distribution ISR register, interrupt 0-31

31	0
IB	
0x00000000	
wc	

31: 0	Distributed interrupt code ISR bits (IB) - The current value of the distributed interrupt code ISR register for interrupt numbers 0 to 31. Each bit index corresponds to the ISR bit value for the corresponding interrupt number. A bit value of 1 indicates that an interrupt code with the corresponding interrupt number has been received, but not yet acknowledged. A bit value of 0 indicates either that no interrupt code with the corresponding interrupt number has been received, or that the previous interrupt code was either acknowledged or timed out. This register should be normally only be used for diagnostics and / or FDIR.

Table 427. 0x00000A28 - RTR.ISR1 - Interrupt code distribution ISR register, interrupt 32-63

31	0
IB	
0x00000000	
wc	

31: 0	Distributed interrupt code ISR bits (IB) - The current value of the distributed interrupt code ISR register for interrupt numbers 32 to 63. Each bit index + 32 corresponds to the ISR bit value for the corresponding interrupt number. A bit value of 1 indicates that an extended interrupt code with the corresponding interrupt number has been received. A bit value of 0 indicates either that no extended interrupt code with the corresponding interrupt number has been received, or that the previous interrupt cod has timed out. Note that if the distributed interrupts are operating in interrupt with acknowledgements mode, this register is unused. This register should be normally only be used for diagnostics and / or FDIR.

Table 428. 0x00000A30 - RTR.ISRTIMER - Interrupt code distribution ISR timer reload

31	0
RL	
*	
rw	

31: 0	Interrupt code distribution ISR timer reload (RL) - Interrupt code distribution ISR timer reload value, counted in prescaler ticks. Each ISR bit has its own timer, which is started and reloaded with the value of this field when an interrupt code / extended interrupt code with the corresponding interrupt number is received (or generated by the router). Reset value is set through IRQTIMEOUTRELOAD[31:0] pins. See section 33.2.16 for details on interrupt code distribution.

Table 429. 0x00000A34 - RTR.AITIMER - Interrupt code distribution ACK-to-INT timer reload

31	0
RL	
*	
rw	

31: 0 Interrupt acknowledgement code to interrupt code timer reload (RL) - Interrupt acknowledgement code to interrupt code timer reload value, counted in prescaler ticks. When an interrupt acknowledgement code is received - for which the router generated the corresponding interrupt code - the interrupt acknowledgement code to interrupt code timer is started and reloaded with the value of this field. Reset value is set through IRQGENRELOAD[31:0] signal. This register is unused when the distributed interrupts are operating in the extended interrupt mode. See section 33.2.16 for details on interrupt code distribution.

Table 430. 0x00000A38 - RTR.ISRCTIMER - Interrupt code distribution ISR change timer reload

31	0
RL	
0	
rw	

31: 0 Interrupt code distribution ISR change timer reload (RL) - Interrupt code distribution ISR change timer reload value, counted in prescaler ticks. Each time an ISR bit change value, the corresponding ISR change timer is started and reloaded with the value of this field. See section 33.2.16 for details on interrupt code distribution.

Table 431. 0x00000A40 - RTR.LRUNSTAT - Link running status

31 19	18 1	0
RESERVED	LR	R
0x0000	0x00000	0
r	r	r

31: 19 RESERVED

18: 1 Link running status (LR)- Each bit is set to 1 when the link interface for the SpaceWire port with the same number as the bit index is in run-state. If the link interface is not in run-state, the bit is set to 0.

0 RESERVED

Table 432. 0x00000A44 - RTR.CAP - Capability

31 26	25	24	23	22 20	19	18 16	15	14	13	12	11	10	9 5	4 0
RESERVED				PF	R	RM	R	AA	AX	R	ID	SD	PC	CC
0x000				0x2	0	0x5	0	1	1	0	1	1	0x1F	0x1F
r				r	r	r	r	r	r	r	r	r	r	r

31: 23 RESERVED

22: 20 Port N-char FIFO size (PF) - The number of entries in the port FIFOs can be determined by the value of this field, according to the formula: Entries = $2^{(RTR.CAP.PF+4)}$. Constant value of 0x2 = 64 entries.

19 RESERVED

18: 16 RMAP maximum data length (RM) - This field specifies the maximum data length in an RMAP read / write command that the configuration port can handle. The length can be determined according to the formula: Length = $2^{(RTR.CAP.RM+2)}$. Constant value of 0x5 = 128 bytes.

15 RESERVED

Table 432. 0x00000A44 - RTR.CAP - Capability

14	Asynchronous axiliary time-code / distributed interrupt code support (AA) - Specifies that the router has support for the auxiliary time-code / distributed interrupt code interface inputs to be asynchronous to CLK. See section 33.2.15. Constant value of 1.
13	Auxiliary time-code / distributed interrupt code support (AX) - Specifies that the router has support for the auxiliary time-code / distributed interrupt code feature described in 33.2.15. Constant value of 1.
12	RESERVED
11	Distributed interrupt code support (ID) - Specifies that the router has support for the interrupt distribution scheme, described in 33.2.16. Constant value of 1.
10	SpaceWire-D support (SD) - Specifies that the router has support for the SpaceWire-D, described in section 33.2.18. Constant value of 1.
9: 5	Port packet counter bits (PC) - Specifies the number of bits in the port's incoming / outgoing packet counters. Constant value of 0x1F = 31 bits
4: 0	Port character counter bits (CC) - Specifies the number of bits in the port's incoming / outgoing character counters. Constant value of 0x1F = 31 bits

Table 433. 0x00000A50 - RTR.PNPVEND - SpaceWire Plug-and-Play - Device Vendor and Product ID

31 16	15 0
VI	PI
0x0003	0x0718
r	r

31: 16	SpaceWire Plug-and-Play Vendor ID (VI) - Double mapping of the VEND bits from the SpaceWire Plug-and-Play Device Vendor and Product ID field. See table 446.
25: 0	SpaceWire Plug-and-Play Product ID (PI) - Double mapping of the PROD bits from the SpaceWire Plug-and-Play Device Vendor and Product ID field. See table 446.

Table 434. 0x00000A54 - RTR.PNPUVEND - SpaceWire Plug-and-Play - Unit Vendor and Product ID

31 16	15 0
VI	PI
0x0000	0x0000
rw	rw

31: 16	SpaceWire Plug-and-Play Unit vendor ID (VI) - Double mapping of the VEND bits from the SpaceWire Plug-and-Play Unit Vendor and Product ID field (see table 455).
25: 0	SpaceWire Plug-and-Play Unit product ID (PI) - Double mapping of the PROD bits from the SpaceWire Plug-and-Play Unit Vendor and Product ID field (see table 455).

Table 435. 0x00000A58 - RTR.PNPUSN - SpaceWire Plug-and-Play - Unit Serial Number

31 8	7 0
SN	
0x000000	*
rw	

31: 0	SpaceWire Plug-and-Play Unit serial number (SN) - Double mapping of the SpaceWire Plug-and-Play Unit Serial Number field (see table 456). Reset value for bits 3:0 is set through INSTANCEID[7:0] signal.

Table 436. 0x00000C10,0x00000C20...0x00000D30 - RTR.OCHARCNT - Outgoing character counter, ports 1-19

31	30 ... 0
OR	CC
0	0x00000000
wc	rw*

31 Counter overrun (OR) - This bit is set to 1 when the character counter (RTR.OCHARCNT.CC) overflows. A write with a 1 to this field will clear the whole character counter (including this bit)

30: 0 Character counter (CC) - Number of data characters (EOP, EEP, time-codes, distributed interrupt codes are not included) that have been transmitted on the corresponding port. When the counter reaches its maximum value, it sets the RTR.OCHARCNT.OR bit to 1 and continue counting from zero. A write to this field where bit 30 is set to 1 will reset the RTR.OCHARCNT.OR bit. A write to this field where bit 30 is set to 0 has no effect.

Table 437. 0x00000C14,0x00000C24...0x00000D34 - RTR.ICHARCNT - Incoming character counter, ports 1-19

31	30 ... 0
OR	CC
0	0x00000000
wc	rw*

31 Counter overrun (OR) - This bit is set to 1 when the character counter (RTR.ICHARCNT.CC) overflows. A write with a 1 to this field will the whole character counter (including this bit)

30: 0 Character counter (CC) - Number of data characters (EOP, EEP, time-codes, distributed interrupt codes are not included) that have been received on the corresponding port. When the counter reaches its maximum value, it sets the RTR.ICHARCNT.OR bit to 1 and continue counting from zero. A write to this field where bit 30 is set to 1 will reset the RTR.ICHARCNT.OR bit. A write to this field where bit 30 is set to 0 has no effect.

Table 438. 0x00000C18,0x00000C28...0x00000D38 - RTR.OPKTCNT - Outgoing packet counter, ports 1-19

31	30 ... 0
OR	PC
0	0x00000000
wc	rw*

31 Counter overrun (OR) - This bit is set to 1 when the packet counter (RTR.OPKTCNT.PC) overflows. A write with a 1 to this field will reset the whole character counter (including this bit).

30: 0 Packet counter (PC) - Number of packets that have been transmitted on the corresponding port. When the counter reaches its maximum value, it sets the RTR.OPKTCNT.OR bit to 1 and continue counting from zero. A write to this field where bit 30 is set to 1 will reset the RTR.OPKTCNT.OR bit. A write to this field where bit 30 is set to 0 has no effect.

Table 439. 0x00000C1C,0x00000C2C...0x00000D3C - RTR.IPKTCNT - Incoming packet counter, ports 1-19

31	30 ... 0
OR	PC
0	0x00000000
wc	rw*

Table 439. 0x00000C1C,0x00000C2C...0x00000D3C - RTR.IPKTCNT - Incoming packet counter, ports 1-19

31 Counter overrun (OR) - This bit is set to 1 when the packet counter (RTR.IPKTCNT.PC) overflows. A write with a 1 to this field will reset the whole character counter (including this bit).

30: 0 Packet counter (PC) - Number of packets that have been received on the corresponding port. When the counter reaches its maximum value, it sets the RTR.IPKTCNT.OR bit to 1 and continue counting from zero. A write to this field where bit 30 is set to 1 will reset the RTR.IPKTCNT.OR bit. A write to this field where bit 30 is set to 0 has no effect.

Table 440. 0x00000E00-0x00000E4C - RTR.MAXPLEN - Maximum packet length, ports 0-19

31 25	24 0
RESERVED	ML
0x00	0x000000
r	rw

31: 25 RESERVED

24: 0 Maximum packet length (ML) - Maximum length of packets for which the corresponding port is the input port. This field is only used when the RTR.PCTRL.PL bit (RTR.PCTRLCFG.PL for port 0) is set to 1. See section 33.2.18 for details.

Table 441. 0x00000E84-0x00000EC8 - RTR.CREDCNT - Credit counter, ports 1-18

31 12	11 6	5 0
RESERVED	OC	IC
0x00000	0	0
r	r	r

31: 12 RESERVED

11: 6 Out credit counter (OC) - Number of outgoing credits. For each credit, the other end of the link is allowed to send one N-Char.

5: 0 In credit counter (IC) - Number of incoming credits. For each credit, the port is allowed to transmit one N-Char.

Table 442. 0x00001004-0x000013FC - RTR.RTCOMB - Routing table, combined port mapping and address control, addresses 1-255

31	30	29	28	27 20	19 1	0
SR	EN	PR	HD	RESERVED	PE	PD
N/R	0	N/R	N/R	0x00	N/R	N/R
rw	rw	rw	rw	r	rw	rw

31 Spill-if-not-ready (SR) - This bit is a double mapping of the RTR.RTACTRL.SR bit. See table 408.

30 Enable (EN) - This bit is a double mapping of the RTR.RTACTRL.EN bit. See table 408.

29 Priority (PR) - This bit is a double mapping of the RTR.RTACTRL.PR bit. See table 408.

28 Header deletion (HD) - This bit is a double mapping of the RTR.RTACTRL.HD bit. See table 408.

27: 20 RESERVED

19: 1 Port enable bits (PE) - This field is a double mapping of the RTR.RTPMAP.PE field. See table 407.

0 Packet distribution (PD) - This field is a double mapping of the RTR.RTPMAP.PD field. See table 407.

NOTE: See note for RTR.RTPMAP (table 407).

33.6.1 SpaceWire Plug-and-Play interface

The configuration port supports parts of the SpaceWire Plug-and-Play protocol described in [SPW-PNP]. The supported fields are listed in table 445, and explained in more detail in tables 446 through 460.

The SpaceWire Plug-and-Play protocol uses standard RMAP commands and replies with the same requirements as presented in section 33.5.1, but with the following differences:

- Protocol Identifier field of a command shall be set to 0x03.
- A command's address fields shall contain a word address. The SpaceWire Plug-and-Play addresses are encoded as shown in table 443.
- The increment bit in the command's instruction field shall be set to 1, otherwise a reply with Status field set to 0x0A (authorization failure) is sent.
- RMAP Read-modify-write command is replaced by a compare-and-swap operation. The command's data fields shall contain the new data to be written, while the mask fields shall contain the value that the current data must match in order for the new data to be written. If there is a mismatch, a reply with Status field set to 0x0A (authorization failure) is sent.
- The reply packet's Status field can contain the additional status codes described in table 444.

Table 443. SpaceWire Plug-and-Play address encoding

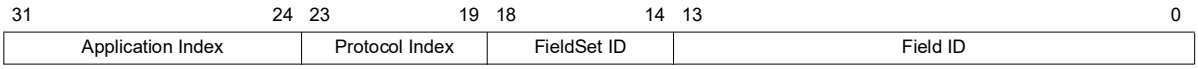


Table 444. SpaceWire Plug-and-Play status codes

Value	Description
0xF0	Unauthorized access - A write, or compare-and-swap command arrived either when the router was not configured (Device ID field = 0), or the command did not match the owner information saved in the Link Information field and Owner Address fields.
0xF1	Reserved field set - A read, write, or compare-and-swap command's address field points to a non existing field set.
0xF2	Read-only field - A write, or compare-and-swap command's address points to a read-only field.
0xF3	Compare-and-swap-only-field - A write command's address points to a compare-and-swap-only field.

Note that it is not possible to access the SpaceWire Plug-and-Play fields through the AHB slave interface, except for the fields that are double mapped into the configuration port's address space (see section 33.6).

An access (read, write, or compare-and-swap) made either to a field outside the Device Information service, or to a field in an undefined field set within the Device Information service, will generate a reply with the Status field set to 0xF1. An access (read, write, or compare-and-swap) to an undefined or unsupported field in one of the defined field sets, within the Device Information service, is not treated as an error, and the Status field of the reply will be 0x00. Possible write-data for such an access is discarded, and possible read-data returned is always 0.

Please reference the [SPWPNP] for additional details to what is presented in this section.

*Table 445.*SpaceWire Plug-and-Play support

SpW PnP Address	Register name	Acronym	Service - Field set - Field
0x00000000	SpaceWire Plug-and-Play - Device Vendor and Product ID	RTR.PNPVEND	Device Information - Device Identification - Device Vendor and Product ID
0x00000001	SpaceWire Plug-and-Play - Version	RTR.PNPVER	Device Information - Device Identification - Version
0x00000002	SpaceWire Plug-and-Play - Device Status	RTR.PNPDEVSTS	Device Information - Device Identification - Device Status
0x00000003	SpaceWire Plug-and-Play - Active Links	RTR.PNPACTLNK	Device Information - Device Identification - Active Links
0x00000004	SpaceWire Plug-and-Play - Link Information	RTR.PNPLNKINFO	Device Information - Device Identification - Link Information
0x00000005	SpaceWire Plug-and-Play - Owner Address 0	RTR.PNPOA0	Device Information - Device Identification - Owner Address 0
0x00000006	SpaceWire Plug-and-Play - Owner Address 1	RTR.PNPOA1	Device Information - Device Identification - Owner Address 1
0x00000007	SpaceWire Plug-and-Play - Owner Address 2	RTR.PNPOA2	Device Information - Device Identification - Owner Address 2
0x00000008	SpaceWire Plug-and-Play - Device ID	RTR.PNPDEVID	Device Information - Device Identification - Device ID
0x00000009	SpaceWire Plug-and-Play - Unit Vendor and Product ID	RTR.PNPUVEND	Device Information - Device Identification - Unit Vendor and Product ID
0x0000000A	SpaceWire Plug-and-Play - Unit Serial Number	RTR.PNPUSN	Device Information - Device Identification - Unit Serial Number
0x00004000	SpaceWire Plug-and-Play - Vendor String Length	RTR.PNPVSTRL	Device Information - Vendor / Product String - Vendor String Length
0x00006000	SpaceWire Plug-and-Play - Product String Length	RTR.PNPPSTRL	Device Information - Vendor / Product String - Product String Length
0x00008000	SpaceWire Plug-and-Play - Protocol Count	RTR.PNPPCNT	Device Information - Protocol Support - Protocol Count
0x0000C000	SpaceWire Plug-and-Play - Application Count	RTR.PNPACNT	Device Information - Application Support- Application Count

Table 446. 0x00000000 - RTR.PNPVEND - SpaceWire Plug-and-Play - Device Vendor and Product ID

31 16	15 0
VEND	PROD
0x0003	0x0716
r	r

31: 16 Vendor ID (VEND) - SpaceWire vendor ID assigned to Frontgrade Gaisler. Constant value of 0x0003.

15: 0 Product ID (PROD) - Product ID assigned to GR716B products. Constant value of 0x0716

Table 447. 0x00000001 - RTR.PNPVER - SpaceWire Plug-and-Play - Version

31 24	23 16	15 8	7 0
MAJOR	MINOR	PATCH	RESERVED
0x01	0x03	0x00	0x00
r	r	r	r

31: 24 Major version number (MAJOR) - Constant value of 0x01.

23: 16 Minor version number (MINOR) - Constant value of 0x03.

15: 8 Patch / Build number (PATCH) - Constant value of 0x00.

7: 0 RESERVED

Table 448. 0x00000002 - RTR.PNPDEVSTS - SpaceWire Plug-and-Play - Device Status

31 8	7 0
RESERVED	STATUS
0x000000	0x00
r	r

31: 8 RESERVED

7: 0 Device status (STATUS) - Constant value of 0x00.

Table 449. 0x00000003 - RTR.PNPACTLNK - SpaceWire Plug-and-Play - Active Links

31 20	19 1	0
RESERVED	ACTIVE	R
0x000	0x00000	0
r	r	r

31: 20 RESERVED

19: 1 Link active (ACTIVE) - If set to 1, the port with the same number as the bit index is running. If set to 0, the port is not running. For the SpaceWire ports (ports 1-18), the corresponding bit will be set to 1 if the link interface is in run-state and the port is not disabled through the Port Control register (RTR.PCTRL.DI = 0). For the SIST port (port 19), the bit is set to 1 if RTR.PCTRL.DI = 0.

0 RESERVED

Table 450. 0x00000004 - RTR.PNPLNKINFO -SpaceWire Plug-and-Play - Link Information

31 24	23 22	21	20 16	15 13	12 8	7	6	5	4 0
OLA	OAL	R	OL	RES	RL	T	U	R	LC
0x00	0x0	0	0x0	0x0	0x0	1	0	0	0x13
r	r	r	r	r	r	r	r	r	r

31: 24 Owner logical address (OLA) - Shows the value of the Initiator Logical Address field from the last successful compare-and-swap command that set the Device ID field.

23: 22 Owner address length (OAL) - Shows how many of the three Owner Address fields that contain valid data.

21 RESERVED

20: 16 Owner link (OL) - Shows the number of the port which was used for the last successful operation to set the value of the Device ID field.

15: 13 RESERVED

Table 450. 0x00000004 - RTR.PNPLNKINFO -SpaceWire Plug-and-Play - Link Information

12: 8	Return link (RL) - Shows the number of the port through which the reply to the current read command will be transmitted.
7	Device type (T) - Constant value of 1, indicating that this device is a router.
6	Unit information (U) - Indicates if the unit identification information (Unit Vendor and Product ID field, and Unit Serial Number field) are valid. 0 = invalid, 1 = valid. This bit will be 0 after reset / power-up. Once the Unit Vendor and Product ID field has been written with a non-zero value, this bit will be set to 1.
5	RESERVED
4: 0	Link count (LC) - Shows the number of router ports. Constant value of 0x13.

Table 451. 0x00000005 - RTR.PNPOA0 - SpaceWire Plug-and-Play - Owner Address 0

31	0
RA	
0x00000000	
r	

31: 0	Reply address (RA) - Shows byte 0-3 of the Reply Address from the last successful compare-and-swap command that set to the Device ID field. If there was no Reply Address, then this field is zero.

Table 452. 0x00000006 - RTR.PNPOA1 - SpaceWire Plug-and-Play - Owner Address 1

31	0
RA	
0x00000000	
r	

31: 0	Reply address (RA) - Shows byte 4-7 of the Reply Address from the last successful compare-and-swap command that set to the Device ID field. If the Reply Address was four bytes or less, then this field is zero.

Table 453. 0x00000007 - RTR.PNPOA2 - SpaceWire Plug-and-Play - Owner Address 2

31	0
RA	
0x00000000	
r	

31: 0 Reply address (RA) - Shows byte 8-11 of the Reply Address from the last successful compare-and-swap command that set to the Device ID field. If the Reply Address was eight bytes or less, then this field is zero.

Table 454. 0x00000008 - RTR.PNPDEVID - SpaceWire Plug-and-Play - Device ID

31	0
DID	
0x00000000	
cas	

31: 0 Device ID (DID) - Shows the device identifier. After reset / power-up, or when this field is written to zero, the router is not considered to have an owner. The same applies to the case when the port indicated by the OL bits in the Link Information field is either disconnected, or disabled by setting the RTR.PCTRL.DI bit to 1. This field is only writable through a compare-and-swap operation.

Table 455. 0x00000009 - RTR.PNPUVEND - SpaceWire Plug-and-Play - Unit Vendor and Product ID

31	16	15	0
VEND		PROD	
0x0000		0x0000	
r		r	

31: 16 Unit vendor ID (VEND) - Shows the unit vendor identifier. This field is read-only through the SpaceWire Plug-and-Play protocol, however it is writable through RMAP and AHB (see section33.6). When this field, ot the PROD field, is written with a non-zero value, the U bit in the Link Information field is set to 1.

15: 0 Unit product ID (VEND) - Shows the unit product identifier. This field is read-only through the SpaceWire Plug-and-Play protocol, however it is writable through RMAP and AHB (see section 33.6). When this field, or the VEND field, is written with a non-zero value, the U bit in the Link Information field is set to 1.

Table 456. 0x0000000A - RTR.PNPUSN - SpaceWire Plug-and-Play - Unit Serial Number

31	0
USN	
0x00000000	
r	

31: 0 Unit serial number (USN) - Shows the unit serial number. This field is read-only through the SpaceWire Plug-and-Play protocol, however it is writable through RMAP and AHB (see section 33.6).

Table 457. 0x00004000 - RTR.PNPVSTRL - SpaceWire Plug-and-Play - Vendor String Length

31 15	14 0
RESERVED	LEN
0x00000	0x0000
r	r

31: 15 RESERVED

14: 0 Vendor string length (LEN) - Constant value of 0, indicating that no vendor string is present.

Table 458. 0x00006000 - RTR.PNPPSTRL - SpaceWire Plug-and-Play - Product String Length

31 15	14 0
RESERVED	LEN
0x00000	0x0000
r	r

31: 15 RESERVED

14: 0 Product string length (LEN) - Constant value of 0, indicating that no product string is present.

Table 459. 0x00008000 - RTR.PNPPCNT - SpaceWire Plug-and-Play - Protocol Count

31 5	4 0
RESERVED	PC
0x0000000	0x00
r	r

31: 5 RESERVED

4: 0 Protocol count (PC) - Constant value of 0, indicating that no protocols can be managed by using SpaceWire Plug-and-Play.

Table 460. 0x0000C000 - RTR.PNPACNT - SpaceWire Plug-and-Play - Application Count

31 8	7 0
RESERVED	AC
0x000000	0x00
r	r

31: 8 RESERVED

7: 0 Application count (AC) - Constant value of 0, indicating that no applications can be managed by using SpaceWire Plug-and-Play.

34 SpaceWire - Time Distribution Protocol

34.1 Overview

This core provides basic time keeping functions such as Elapsed Time counter according to the CCSDS Unsegmented Code specification. It provides support for setting and sampling the Elapsed Time counter. It also includes a frequency synthesizer with which a binary frequency is generated to drive the Elapsed Time counter. This interface implements the SpaceWire - Time Distribution Protocol (TDP). The protocol provides capability to transfer time values and synchronise them between onboard users of SpaceWire network. The time values are transferred as CCSDS Time Codes and synchronisation is performed through SpaceWire Time-Codes. The core also provides datation services. The AMBA APB bus is used for configuration, control and status handling.

34.2 Protocol

The initiator and target maintain their own time locally. The Time Distribution Protocol provides the means for transferring time of initiator to targets and for providing a synchronization point in time. The time is transferred by means of an RMAP write command carrying a CCSDS Time Code (time message). The synchronization event is signaled by means of transferring a SpaceWire Time-Code. The transfer of the SpaceWire Time-Code is synchronized with time maintained by the initiator. To distinguish which SpaceWire Time-Code is to be used for synchronization, the value of SpaceWire Time-Code is transferred from initiator to target by means of an RMAP write command prior to actual transmission of SpaceWire Time-Code itself. When there is more than one target the CCSDS Time Code need to be transferred to each individual target separately [SPWCUC].

34.3 Functionality

The block diagram below shows how the controller is connected to the system.

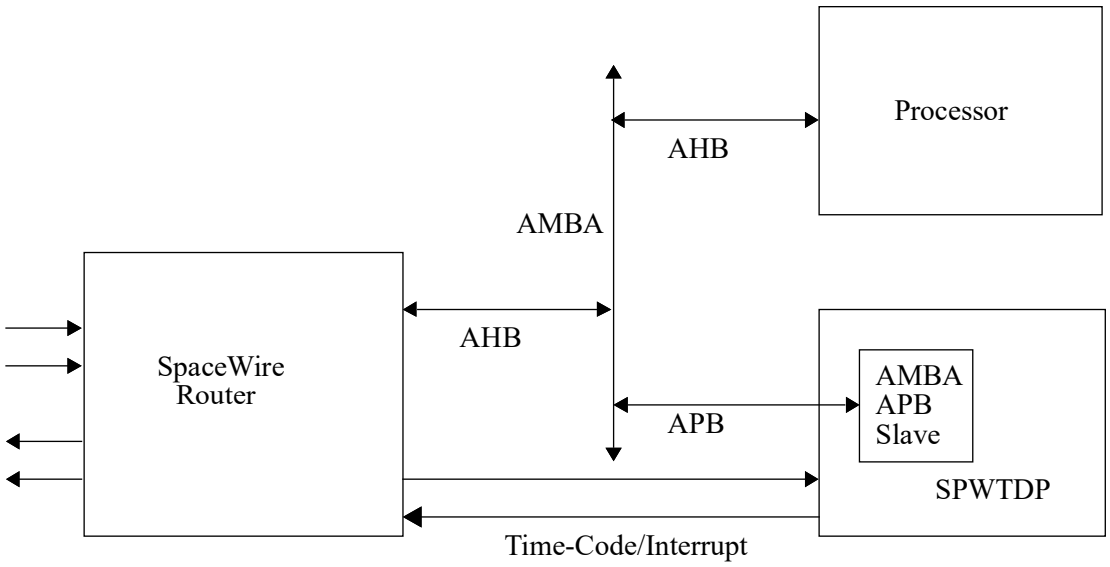


Figure 79. Block diagram

The foreseen usage of this core is to distribute and synchronise time between an initiator SPWTDP core and one or more target SPWTDP (slave) cores using the SpaceWire interface for communication between them.

The system can act as initiator (time master) and target being able to send and receive SpaceWire Time-Codes. The initiator requires SpaceWire link interface implements an RMAP initiator (in

GR716B the SpW router AMBA ports with the help of software can act as RMAP initiator). The Target requires SpaceWire link interface implements an RMAP target (in GR716B the SpW router AMBA ports can handle RMAP packets in HW as a target). The SPWTDP component is a part of this system providing SpaceWire Time-Codes, CCSDS Time Codes, datation, time-stamping of distributed interrupts, support for transmission of CCSDS Time Codes through RMAP and support for latency measurement and correction. In this implementation the CCSDS Time Codes carried between the SpaceWire network is based on CCSDS Unsegmented Code format (CUC) which is explained below [CCSDS]. The table below shows an example Preamble Field (P-Field) which corresponds to 32 bits of coarse time and 24 bits of fine time.

34.3.1 CCSDS Unsegmented Code: Preamble Field (P-Field)

*Table 461.*CCSDS Unsegmented Code P-Field definition

Bit	Value		Interpretation
0	"0"		Extension flag, P-Field extended with 2nd octet
1-3	"010"	Agency-defined epoch (Level 2)	Time code identification
4 - 5	"11"	(number of octets of coarse time) + 1	Detail bits for information on the code
6 - 7	"11"	(number of octets of fine time)	
8	"0"		Extension flag, P-Field not extended with 3rd octet
9-10	"00"	Number of additional octets of the coarse time.	added to octet 1
11-13	"000"	Number of additional octets of the fine time.	added to octet 1
14-15			RESERVED

34.3.2 CCSDS Unsegmented Code: Time Field (T-Field)

For the unsegmented binary time codes described herein, the T-Field consists of a selected number of contiguous time elements, each element being one octet in length. An element represents the state of 8 consecutive bits of a binary counter, cascaded with adjacent counters, which rolls over at a modulo of 256.

*Table 462.*Example CCSDS Unsegmented Code T-Field with 32 bit coarse and 24 bit fine time

CCSDS Unsegmented Code											
Preamble Field	Time Field										
	Coarse time								Fine time		
-	2^{31}	2^{24}	2^{23}	2^{16}	2^{15}	2^{8}	2^{7}	2^{0}	2^{-1}	2^{-8}	2^{-9}	2^{-15}	2^{-16}	2^{-24}
0:15	0						31		32					55

The basic time unit is the second. The T-Field coarse time (seconds) can be maximum 56 bits and minimum 8 bits. The T-Field fine time (sub seconds) can be maximum 80 bits and minimum of 0 bits.

The number of bits representing coarse and fine time implemented in this core can be obtained by reading the DPF bits of Datation Preamble Field register.

The coarse time code elements are a count of the number of seconds elapsed from the initial time value. This code is not UTC-based and leap second corrections do not apply according to CCSDS.

34.3.3 Time generation

The core consist of time generator which is the source for time in this system. The core may act as initiator or a target but both have their respective time generator. The Elapsed Time (ET) counter is implemented complying with the CUC T-Field. The number of bits representing coarse and fine time

of a ET counter implemented in a design can be obtained by reading the DPF bits of Datation Preamble Field register.

The ET counter can be incremented either using an internal frequency synthesizer or by using an external enable signal. The External ET Increment Enable bit in Configuration 0 register must be enabled if external inputs are to be used.

Increment ET using internal frequency synthesizer:

The counter is incremented on the system clock only when enabled by the frequency synthesizer. The binary frequency required to determine the counter increment is derived from the system clock using a frequency synthesizer (FS). The frequency synthesizer is incremented with a pre-calculated increment value, which matches the available system clock frequency. The frequency synthesizer generates a tick every time it wraps around, which makes the ET time counter to step forward with the pre-calculated increment value. The output of frequency synthesizer is used for enabling the increment of ET counter. The increment rate of the ET counter and frequency synthesizer counter should be set according to the system clock frequency. The ET counter increment rate is set by providing values to ETINC bits in Configuration 2 register and frequency synthesizer counter is set by providing values to FSINC bits in Configuration 1 register. The following table specifies some example ETINC and FSINC values for some frequencies. The below values are also obtained for Coarse time width 32, Fine time width 24 and Frequency synthesizer width of 30. To calculate for other frequencies and configuration refer the spreadsheet [SPWTDP].

Table 463. Example values of ETINC and FSINC for corresponding frequencies

Frequency MHz	**ETINC**	**FSINC**
100	0	180143985
50	0	360287970
33333333	2	135107990

Increment ET using external input:

The EP register in Configuration 0 specify whether to increment the ET counter based on rising or falling edge of the external enable signal. Also the ETINC bits in Configuration 2 register specify from which bit the ET counter must increment.

The following section describes the cores capabilities if it configured as initiator or target.

34.3.4 Initiator

An initiator is a SpaceWire node distributing CCSDS Time Codes and SpaceWire Time-Codes. It is also an RMAP initiator, capable of transmitting RMAP commands and receiving RMAP replies. There is only one active initiator in a SpaceWire network during a mission phase.

The initiator performs the following tasks

- Transmission of SpaceWire Time-Codes. The SpaceWire Time-Codes are provided by this component and transmission of those codes to targets should be performed by a SpaceWire interface. (in GR716B the SpW router SpW ports perform the transmission of time codes).
- Transmission of CCSDS Time Codes through RMAP. The SPWTDP core provides the required CCSDS Time Codes, the formation of RMAP packets should be done in SW and the RMAP packets can be transmitted using the SpW router AMBA port.
- Datation, time-stamping and latency measurement

34.3.5 Target

A target is a SpaceWire node receiving CCSDS Time Codes and SpaceWire Time-Codes. A target is also an RMAP target, capable of receiving RMAP commands and transmitting RMAP replies. There can be one or more targets in a SpaceWire network.

The target performs the following tasks

- Reception of SpaceWire Time-Codes. The SpaceWire Time-Codes sent from initiator are received by SpaceWire interface (in GR716B SpW router SpW ports receives time codes) and provided to this component in target.
- Reception of CCSDS Time Codes through RMAP. In GR716B, the RMAP packets are received by the SpW router AMBA ports and handles by its RMAP target.
- Qualification of received time messages (CCSDS Time Codes) using SpaceWire Time-Codes
- Initialization and Synchronisation of received CCSDS Time Codes with Elapsed Time counter available in this component
- Datation, time-stamping and latency correction

34.3.6 Configuring initiator and target

The core is interfaced via an AMBA Advanced Peripheral Bus (APB) slave interface, providing a register view that is compatible with the Time Distribution Protocol (TDP). The core must be configured according to the requirement either as initiator or target.

- Initializing initiator

The initiator transmits the SpaceWire Time-Codes out of the core only when the Transmit Enable TE bit in Configuration 0 register is enabled. The ET counter in initiator can be initialized (to provide any initial value). Initialization is done by writing a time value into the Command Elapsed Time registers available in the command field, the NC bit in the Control register of command field should be enabled to initialize the time value stored in the Command Elapsed Time registers to be the local time (Transmit Enable TE bit in Configuration 0 register must be enabled). The NC bit in the Control register will disable itself when the time is initialized. The INSYNC bit in Status 0 register will enable when initialization is performed. The MAPPING bits in Configuration 0 register determines the interval between SpaceWire Time-Code transmissions which is explained in detail in the section below.

The target time must be configured with time values from the initiator. The targets register space must be configured and controlled through RMAP by an initiator to achieve time synchronisation. The target time synchronisation is explained in detail under the section initialization and synchronisation of target through RMAP.

34.3.7 SpaceWire Time-Code

SpaceWire Time-Codes are continuously transmitted from an initiator node (time master) to all slave nodes. The transmission of the SpaceWire Time-Code is synchronized with the ET counter in the initiator node. The six bits of the Time-Code time information correspond to six bits of the local ET counter (MAPPING bits in Configuration 0 register determines its exact mapping and interval between SpaceWire Time-Code transmissions). Value of 0b00000 for MAPPING bits in Configuration 0 register will send SpaceWire Time-Code at every Second. When the value is 0b00001 SpaceWire Time-Codes are sent at every 0.5 Seconds interval and so on (maximum value of MAPPING can be 0b11111 but this value cannot be more than the number of bits implemented as fine time). The ET bits with lower weights than the size bits mapped to Time Codes time information bits are all zero at time of SpaceWire Time-Codes transmission. The Table below shows an example Local ET

counter and Mapping. If the Coarse time is 32 bits and Fine time is 24 bits and mapping value is 6 then 0 to 31 is coarse(32 bits), 32 to 55 is fine time and mapped SpaceWire Time-Code is 32 to 37.

*Table 464.*Example Local ET counter with Mapping values

0															25	26	27	28	29	30	31	32	33	34	35	36	37	38				55
Mapping Values																					0	1	2	3	4	5	6	7				24
If the Mapping value is 6 then the mapped SpaceWire Time-Codes is 32 to 37																						32	33	34	35	36	37					
If the Mapping value is 0 then the mapped SpaceWire Time-Codes is 26 to 31																26	27	28	29	30	31											
If the Mapping value is 5 then the mapped SpaceWire Time-Codes is 31 to 36																					31	32	33	34	35	36					
If the Mapping value is 7 then the mapped SpaceWire Time-Codes is 33 to 38																							33	34	35	36	37	38				

34.3.8 Initialization and synchronisation of target through RMAP

An initiator must provide the time values and set the target in order to get the time synchronized. The below text explains how an initiator can synchronise the target.

The SPWTC in Control register of initiator core component should be configured initially with a SpaceWire Time-Code value at which the time message needed to be transferred. When the SpaceWire Time-Code generated internally using the ET counter matches the SPWTC in Control register a Time Message TM interrupt will be generated (TME bit Time Message Enable should be enabled in the Interrupt Enable register). Based on this interrupt the local time (ET counter) in initiator should be accessed from the Datation registers and used to calculate the time message needed to be transmitted.

- Time message generation

The Time message transmitted using RMAP should be an exact mapping of the Command field (explained under Registers section). The Time message transmitted should write the Command field available in target. Control register available in Command field specify weather the target should be initialized or synchronized, at which SpaceWire Time-Codes it should happen (synchronization event) and details of coarse and fine time available in the time message. The New code NC bit available in Control register should be enabled and if the target should be initialized then Init Sync IS bit in Control register must be enabled otherwise target will be synchronized.

The Command Elapsed Time in time message are calculated from the local time (ET counter) available in the initiator. The local time can be obtained by reading the Datation Field of initiator component. While reading the Datation registers always the total implemented coarse time and fine time must be read in order (from 0 till the implemented Datation Elapsed Time registers). The DPF of Datation Preamble Field register gives the coarse and fine time implemented which gives the total local ET counter (coarse + fine width).

For example if the implementation has 32 bit coarse and 24 bit fine time then it is enough to access the first two Datation Elapsed Time registers (0 and 1). The 32 bits of Datation Elapsed Time 0 and only the most significant 24 bits (31 to 8) of Datation Elapsed Time 1 registers (32 + 24 =56 bits) represents the local time. These 56 bits only be used for Command Elapsed time (time message) calculation.

The SpaceWire Time-Codes at which the Time Message interrupt generated is embedded in the local ET counter. The Command Elapsed time which is transmitted as time message should be an incremented time value of this SpaceWire Time-Code and Command Elapsed time bits with lower weights than the size bits mapped to SpaceWire Time-Code time information bits are all must be zero.

The incremented time value is to make the initialization or synchronisation of time message in target will happen after the reception of qualifying SpaceWire Time-Codes. The qualifying SpaceWire Time-Code is embedded in the Command Elapsed time (part of time message) sent from initiator. This qualifying SpaceWire Time-Code value should also be written in the SPWTC in Control section of the time message.

- Time qualification in target

In target, the Command field will contain the time message when it is written by the initiator through RMAP. When the SPWTC of Control register in Command field matches with a received SpaceWire Time-Code then initialization or synchronization will occur (according to NC bit and IS bit in the Control register) to the local ET counter of the target SPWTDP component. When the local ET counter is initialized or synchronized the NC bit in the control register will disable itself. The INSYNC bit in Status 0 register will enable when initialization is performed specifying the target is initialized. Initialization completely writes time message values into the implemented local Elapsed time counter and synchronisation verifies whether the time message Command Elapsed Time and local Elapsed Time counter matches till the mapped SpaceWire Time-Code level (with a tolerance of previous value) and only modifies the local Elapsed Time if their is a mismatch. Since the GR716B target is not implemented with a jitter and mitigation unit the synchronisation forces the target time (ET counter) with the time message received.

For example, the initiator can create time message exactly at 0x00000001 coarse time and 0x040000 fine time (32 bit coarse time and 24 bit fine time, mapping value of 6 i.e. 64 SpaceWire Time-Codes per second, time message is generated at 0b000001 SpaceWire Time-Code), the value in the time message to be sent to the target can be coarse time 0x00000002 and 0x040000 fine time, (32 bit coarse time and 24 bit fine time, mapping value of 6, time message is qualified at the next reception of 0b000001 SpaceWire Time-Code, i.e. after a second). Both SPWTC in Control registers available in the initiator and target can be 0b000001 for this example. The time is synchronized after a second in this example. Depending on the frequency of SpaceWire Time-Codes and data link rate several different combination of ways to achieve time synchronisation is possible.

34.3.9 Latency measurement using Time-Stamps

The incoming and outgoing SpaceWire Distributed Interrupts are time stamped in initiator and target. The initiator calculates latency based on these time stamp values. The time stamped values in target are accessed from initiator through RMAP. The Latency Enable LE bit in Configuration 0 register must be enabled between the two nodes in the SpaceWire network for which the latency is to be calculated. The core supports 32 distributed interrupts and acknowledgement (Interrupt and acknowledgement numbers 0 to 31). The distributed interrupt transmission from initiator (which is the origin for latency calculation) is controlled by a mask register STM available in Configuration 3 register and SpaceWire time code register TSTC available in Time-Stamp SpaceWire Time-Code and Preamble Field Tx register, these registers specifies how often and at which time code distributed interrupt is transmitted and time stamping is performed.

The time stamping can be performed in two methods (only Interrupts or Interrupts and Acknowledgement), the DI bit in Configuration 3 register of SPWTDP component in target should be configured to specify which type of method is used. If only distributed interrupts (no acknowledgement) are used then DI bit should be 0. The transmitted and received distributed interrupts INTX and INRX in the Configuration 0 registers of both initiator and target must be configured with the interrupt number which will be used for the latency measurement. For example if the INTX in initiator Configuration 0 is configured with 0b00100 then the target INRX should be configured with the same value. Similarly if the INTX in target Configuration 0 is configured to be 0b00101 then the initiator INRX should be configured with the same value. Initially initiator sends a distributed interrupt when the conditions are matched (STM and TSTC registers match) and when the target received this distributed interrupt it will send another interrupt which will be received by the initiator. At each end transmission and reception is time stamped (current local time is stored in Time Stamp registers) and interrupt transmitted is INTX and received interrupt is checked whether it received INRX.

If both distributed interrupts and acknowledgement method is to be used then DI bit should be 1. The transmitted and received distributed interrupts INTX and INRX in the Configuration 0 registers of both initiator and target can have the same interrupt number (the acknowledgement number for a particular interrupt will be same as interrupt number). Similar to the previous method at each end transmission and reception is time stamped which will be used for latency calculations.

The Latency calculation can be started in initiator based on DIR (distributed interrupt received) interrupt available in Interrupt Status register (the interrupt should be enabled in the Interrupt Enable register). The latency is calculated form the time stamp registers based on the equation explained below

Latency = ((initiator time stamp Rx - initiator time stamp Tx) - (target time stamp Tx - target time stamp Rx)) /2

By calculating the Latency value repeatedly (at least for about 128 times, more number of times provides increased accuracy) and taking an average of it will provide the final latency value. The initiator should transfer the latency correction information to the Latency Field registers in the target by means of RMAP transfer. When the latency values are written it will be adjusted to local time in the target and the LC bit in Status 0 register is enabled (set to '1'), this status register can be disabled by writing '1' into the corresponding field.

34.3.10 External Datation

The core provides external datation services, there are four external datation services implemented which can time stamp the Elapsed Time counter when the conditions for a respective event (time stamping) occurs. The event on which time stamp must occur is configurable individually (using the respective mask registers EDMx and also a dedicated mask bit is available for each of the input events) for all the external datation services.

Each of the four external datation services implemented has its own mask EDMx, status EDS and time EDxETx registers. (here the x suffix represent 0, 1, 2 and 3 respect to individual registers available)

All the external datation services share the same event inputs (32 inputs).

The table below describes the inputs connected.

Table 465. Input Events on which time stamp occurs.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
irq 31	irq 30	irq 29	irq 28	irq 27	irq 26	irq 25	irq 24	irq 23	irq 22	irq 21	irq 20	irq 19	irq 18	irq 17	irq 16	irq 15	irq 14	irq 13	irq 12	irq 11	LS	irq 9	irq 8	irq 7	irq 6	irq 5	irq 4	irq 3	irq 2	irq 1	irq 0

	31:	11		The Interrupt lines (11 to 31) are connected as input events. The respective mask bit must be enabled to time stamp on this event and when the condition matches (Interrupt occurs) the time stamp values are available in respective EDxETx registers.
	10			Latch-Save (LS): Input 10 is special case. MIL-STD-1553B controller RTSYNC event. The Elapsed Time is continuously latched when a valid command is detected by the controller (the corresponding mask register bit 10 must be enabled). When RTSYNC event is reported by the MIL-STD-1553B controller the latched value will remain same (saved at when the previous latch condition met) and all the mask bit previously enabled will be cleared. The RTSYNC event is necessary to finalize the condition match.
	9:	0		The Interrupt lines (0 to 9) are connected as input events. The respective mask bit must be enabled to time stamp on this event and when the condition matches (Interrupt occurs) the time stamp values are available in respective EDxETx registers.

Any condition match for a particular external datation service will clear its respective mask register EDMx (clears all the mask bits and must be set again in order to achieve an another time stamp). The condition match can also invert GPIO lines. The condition match on external datation service 0 can invert the GPIO line 7, similarly for services 1, 2 and 3 the respective GPIO lines are 8, 9 and 10. When the corresponding GPIO Pulse register is enabled and external datation event occurs then the respective GPIO line is inverted. The EDS bit in Status Register 0 will go high when the condition matches and cleared when the latched elapsed time is read. The purpose of this status register is to ensure that all the implemented coarse and fine time are read. Reading the lowest implemented fine time makes the status register to go low.

34.3.11 Pulses

The core provides eight external outputs used for clock pulse distribution. The timing of each pulse output is individually derived from the Elapsed Time counter. It is possible to program for each pulse output individually the following parameters:

- periodicity pulse
- width of pulse
- polarity of pulse
- enable/disable

The pulse has two parts, the active and the inactive part. The active part always starts the pulse, followed by the inactive part. The polarity or logical level of the active part is programmable. The inactive part takes the logical inversion of the active pulse, and is the default output from the generator when the pulse is not issued or the overall generation is disabled.

The periodicity of the pulse corresponds to one of the ET bits that can be selected in the range 27 to 2-8 seconds, providing a range from 128 seconds to 3,91 ms, i.e. 0,0078 to 256 Hz frequency. See register definition for details.

The width of the active part of the pulse corresponds to one of the ET bits that can be selected in the range 26 to 2-9 seconds, providing a range from 64 seconds to 1,95 ms. See register definition for details.

It is possible to generate a pulse that has a duty cycle of 50%. It is also possible to generate a pulse for which the active part is as short as 2-9 seconds, and its period is as high as 27 seconds. The effective duty cycle can be as low as 2-9/27 for the longest period, up to 50% for the shortest period of 2-8 sec onds = 256 Hz. The duty cycle choice becomes more restricted as the frequency increases. Note that it is only possible to reduce the duty cycle in one direction: 50%/50%, 25%/75%... 1%/99%. The active part of the pulse can thus never be more than 50% of the cycle. It should be noted that the active pulse width must be at most 50% of the pulse period.

The pulse outputs are guaranteed to be spike free. If a pulse output is disabled by means of writing to the corresponding register (PDRx) (i.e. writing a zero to the Pulse Enable bit (PE)), the pulse output will be immediately driven to the inversion of the Pulse Level bit (PL), which corresponds to the level of the inactive part of the pulse. It is thus possible to modify immediately the pulse output by dis abling it using the PE bit and then changing the PL bit, since the output will always drive the inversion of the PL bit while disabled.

An ongoing pulse output will be immediately disabled (the pulse output will be immediately driven to the inversion of the Pulse Level bit) if any external modification of the ET counter is triggered (for both master and slave) due to initialization/synchronisation or set using registers.

34.3.12 Set Elapsed Time using external input

The ET counter can be set using an external enable signal (configurable rising or falling edge, see register SP in Configuration 0 register). To set the ET counter the SE bit in configuration register must be enabled, the value to be loaded into the ET counter must be written into the Command Elapsed Time registers. The ARM bit field in the Status 0 register will set itself to ‘1’ when the first Command Elapsed Time register is written. After the occurrence of the external enable signal the value will be loaded into the ET counter and the ARM bit field in the Status 0 register will set itself to ‘0’. An interrupt can also be generated when the ET counter is loaded, the corresponding interrupt (Set ET External Interrupt Enable) must be enabled.

34.3.13 Synchronisation of target using SpaceWire Time-Codes

It is possible to synchronise the target only using SpaceWire Time-Codes. A master sending SpaceWire Time-Codes (using its Elapsed time counter) at regular interval can synchronise the Elapsed

time in the target. The frequency of Time-code transmission in the master and the frequency of (when to expect a) Time-code in the target must match, this can be achieved by setting the Mapping fields in the Configuration 0 register. The incoming SpaceWire Time-Codes and the Time-code position mapped in the target Elapsed Time is compared, if they match the bits available after the compared bits are made zero, if the local time map is less than one (External Time-code arrived early) then the bits available after the compared bits are made zero and the other part (including the mapped part) is incremented by one. If the above two cases occurred then the target time is in sync with the master time and the Insync bit in Status 0 register is enabled. If the incoming Time-code is in out of order (Non consecutive) then the synchronisation is stopped, the Insync bit in Status 0 register is disabled (but the local time keeps running) and an interrupt is generated if corresponding interrupt (Non consecutive SpaceWire Time-code Interrupt) bit is enabled.

34.4 Registers

The core is programmed through registers mapped into AMBA APB address space.

*Table 466.*Registers

APB address offset	Register
0x000-0x00F	Configuration Field
0x000	Configuration 0
0x004	Configuration 1
0x008	Configuration 2
0x00C	Configuration 3
0x010 - 0x01F	Status Field
0x010	Status 0
0x014	Status 1
0x018	RESERVED
0x01C	RESERVED
0x020 - 0x03F	Command Field
0x020	Control
0x024	Command Elapsed Time 0
0x028	Command Elapsed Time 1
0x02C	Command Elapsed Time 2
0x030	Command Elapsed Time 3
0x034	Command Elapsed Time 4
0x038	RESERVED
0x03C	RESERVED
0x040 - 0x05F	Datation Field
0x040 - 0x05F	Datation Field
0x040	Datation Preamble Field
0x044	Datation Elapsed Time 0
0x048	Datation Elapsed Time 1
0x04C	RESERVED
0x050	RESERVED
0x054	RESERVED
0x058	RESERVED
0x05C	RESERVED
0x060 - 0x09F	Time-Stamp Field
0x060	Time-Stamp Preamble Field Rx
0x064	Time-Stamp Elapsed Time 0 Rx
0x068	Time-Stamp Elapsed Time 1 Rx
0x06C	RESERVED
0x070	RESERVED
0x074	RESERVED
0x078	RESERVED
0x07C	RESERVED
0x080	Time-Stamp SpaceWire Time-Code and Preamble Field Tx
0x084	Time-Stamp Elapsed Time 0 Tx

0x088	Time-Stamp Elapsed Time 1 Tx
0x08C	RESERVED
0x090	RESERVED
0X094	RESERVED
0x098	RESERVED
0x09C	RESERVED
0x0A0-0x0BF	Latency Field
0x0A0	Latency Preamble Field
0x0A4	Latency Elapsed Time 0
0x0A8	Latency Elapsed Time 1
0x0AC	RESERVED
0x0B0	RESERVED
0x0B4	RESERVED
0x0B8	RESERVED
0x0BC	RESERVED
0x0C0	Interrupt Enable
0x0C4	Interrupt Status
0x0C8	Delay Count
0x0CC	Disable Sync
0x0D0-0x0FF	RESERVED
0x100-0x18F	External Datation Field
0x100	External Datation 0 Mask
0x104	External Datation 1 Mask
0x108	External Datation 2 Mask
0x10C	External Datation 3 Mask
0x110-0x12F	External Datation 0 Time
0x110	External Datation 0 Preamble Field
0x114	External Datation 0 Elapsed Time 0
0x118	External Datation 0 Elapsed Time 1
0x11C	External Datation 0 Elapsed Time 2
0x120	External Datation 0 Elapsed Time 3
0x124	External Datation 0 Elapsed Time 4
0x128	RESERVED
0x12C	RESERVED
0x130-0x14F	External Datation 1 Time
0x150-0x16F	External Datation 2 Time
0x170-0x18F	External Datation 3 Time
0x190-0x19F	RESERVED
0x1A0-1B8	Pulse Definition Register 0 to 5
0x1C0-0x1FF	RESERVED

Table 467. 0x000 - CONF0 - Configuration 0

31 25	24	23 22	21	20	19	18	17	16	15	14 13	12 8	7	6	5 4	3	2	1	0
RESERVED	R	RES	ST	EP	ET	SP	SE	LE	AE	RES	MAPPING	TD	R	SEL	R	RE	TE	RS
0	0	0	0	1	0	1	0	0	0	0	0b00110	0	0	0	0	0	0	0
r	rw	r	rw	rw	rw	rw	rw	rw	rw	r	rw	rw	r	rw	r	rw	rw	rw

31: 25 RESERVED

24 RESERVED

23: 22 RESERVED

21: Synchronisation using SpaceWire Time-Code Enable (only for target).

20: External ET Increment Polarity (EP) - To select the rising or falling edge of the external enable signal to increment the Elapsed time. Value ‘1’ Rising edge. Value ‘0’ Falling edge.

19: External ET Increment Enable.(ET) - To increment the Elapsed Time based on external signal. When disabled the internal frequency synthesizer is used to increment the Elapsed Time counter.

18: Set ET External Polarity (SP) - To select the rising or falling edge of the external enable signal to load the Elapsed Time with the contents of the command field register.

17: Set ET External Enable (SE) - Based on the external enable signal load the Elapsed Time with the contents of the command field register.

16: Latency Enable (LE) - To calculate latency between an initiator and target this bit must be enabled in both of them.

15: AMBA Interrupt Enable (AE) - The interrupts (explained in interrupt registers) in this core will generate an AMBA interrupt only when this bit is enabled.

14 13 RESERVED

12: 8 Mapping (MAP) - Defines mapping of SpaceWire Time-Codes versus CCSDS Time-code.

Value 0b00000 will send SpaceWire Time-Codes every Second,

Value 0b00001 will send SpaceWire Time-Codes every 0.5 Second,

Value 0b00010 will send SpaceWire Time-Codes every 0.25 Second,

Value 0b00011 will send SpaceWire Time-Codes every 0.125 Second

The maximum value it can take is 0b11000.

7: Enable TDP (TD) - Enable to indicate that the TDP provides SpaceWire Time-codes and Distributed interrupts to the SpW router. Internally this signal is connected to the SpW router auxiliary time input enable signal which enables the routers auxiliary interface.

6: RESERVED

5: 4 Select (SEL) - Select for SpaceWire Time-Codes and Distributed Interrupt transmission and reception, one of 0 through 3, (must always be 0b00 in this implementation).

3: RESERVED

2: Receiver Enable (RE) - Enabling this will make the core to act as target.

1 Transmit Enable (TE) - Enabling this will make the core to act as initiator.

The core can act only as an initiator or target, both TE and RE cannot be enabled at the same time.

0 Reset (RS) - Reset core. Makes complete reset when enabled, self clears itself (to disable).

Table 468. 0x004 - CONF 1 - Configuration 1

31 30 29	0
-	FSINC
0	0
r	rw

Table 468. 0x004 - CONF 1 - Configuration 1

31: 30 RESERVED

29: 0 Frequency synthesizer (FSINC) - Increment value of the Frequency Synthesizer which is added to the counter every system clock cycle. It defines the frequency of the synthesized reference time.

Refer the spreadsheet [SPWTDP]

Table 469. 0x008 - CONF 2 - Configuration 2

31 ... 8	7 ... 0
R	ETINC
0	0
r	rw

31: 8 RESERVED

7: 0 Elapsed Time Increment (ETINC) - Value of the Elapsed Time counter is to be incremented each time when the Frequency Synthesizer wraps around.

Refer the spreadsheet [SPWTDP]

Table 470. 0x00C - CONF3 - Configuration 3

31 ... 22	21 ... 16	15	14	13	12	11	10	9 ... 5	4 ... 0
-	STM	-		DI64R	DI64T	DI64	DI	INRX	INTX
0	0	0		0	0	0	0	0	0
r	rw	r		rw	rw	rw	rw	rw	rw

31: 22 RESERVED

21: 16 SpaceWire Time-Code Mask (STM) - Mask For TSTC register available at Time-Stamp SpaceWire Time-Code and Preamble Field Tx register.

Value all bits zero will send Distributed interrupts at all SpaceWire Time-Codes irrespective of any values in TSTC register.

Value all ones will send Distributed interrupts at complete match of SpaceWire Time-Code with TSTC register.

(only for initiator)

15: 14 RESERVED

13: DI64R - The MSb for received Distributed Interrupt when interrupt numbers 32 to 63 is used. Possible only for DI = '0' (only interrupt mode) and DI64 is enabled.

12: DI64T - The MSb for transmitted Distributed Interrupt when interrupt numbers 32 to 63 is used. Possible only for DI = '0' (only interrupt mode) and DI64 is enabled.

11: Enable Distributed Interrupts 64 (DI64) - when set all 64 Distributed interrupt numbers can be used for latency calculation. Possible only for DI = '0' (only interrupt mode).

10: Distributed Interrupt (DI) - Distributed Interrupt method, when set interrupt and acknowledge mode else only interrupt mode. (only for target)

9: 5 Interrupt Received (INRX) - The distributed interrupt number received by initiator or target.

4: 0 Interrupt Transmitted (INTX) - The distributed interrupt number transmitted by initiator or target.

Table 471. 0x010 - STAT 0 - Status Register 0

31	30 28	27 ... 24	23	22 ... 16	15	14 ... 13 ... 8	7 ... 4	3	2	1	0
MA	-	EDS	-	FW	-	CW	-	ARM	LC	TCQ	INSYNC
0	0	0	0	0	0	0	0	0	0	0	0
r	r	r	r	r	r	r	r	r	wc	r	r

31: Mitigation available (MA) - Mitigation unit available

0 Drift and Jitter mitigation unit not available.

30: 28 RESERVED

Table 471. 0x010 - STAT 0 - Status Register 0

27: 24	External Datation Status (EDS) - When conditions matched for external datation this bit will go high. This bit will go low when all the implemented time values are read. 24: External Datation 0 Status bit 25: External Datation 1 Status bit 26: External Datation 2 Status bit 27: External Datation 3 Status bit
23	RESERVED
22: 16	Fine Width (FW) - Fine width of command CCSDS Time Code received. Calculated from Preamble field of Command Register.
15: 14	RESERVED
13: 8	Coarse Width (CW) Coarse width of command CCSDS Time Code received, calculated from Preamble field of Command Register.
7: 4	RESERVED
3:	Armed (ARM) - This field is enabled when the command field register is written with the value to be loaded into the Elapsed time. The Set ET External Enable SE bit in the Configuration 1 must be enabled. When an external enable signal occurred and the command field register contents are loaded into the Elapsed time then this bit will get disabled.
2	Latency Corrected (LC) - Goes high when the latency value is written into latency registers in target (only for target).
1	Time Message Qualified (TCQ)- Time message is qualified by SpaceWire Time-Codes.
0	In Sync (INSYNC) - In Synchronization at Time code level, enabled when time values are Initialized or Synchronized.

Table 472. 0x014 - STAT 1 - Status Register 1

31	30	29 ... 0
-		IV
0		0
r		r

31: 30	RESERVED
29: 0	Increment Variation (IV) - (not usable in this implementation)

Table 473. 0x020 - CTRL - Control

31	30	29 ... 24	23 ... 16	15 ... 0
NC	IS	-	SPWTC	CPF
0	0	0	0	0
rw	rw	r	rw	rw

31:	New Command (NC) - New command is set to provide a new time value.
30:	Init or Sync (IS) -'1' Initialization of received time message 0' Synchronisation of received time message (only for target).
29: 24	RESERVED
23: 16	Spacewire Time-code (SPWTC) - Spacewire Time-code value used for initialization and synchronisation. In initiator the SpaceWire Time-Codes generated internally using the local ET counter matches this register a Time Message TM interrupt will be generated which is used to send Time message over the SpaceWire network. In target this register should match the received SpaceWire Time-code for time qualification.
15: 0	Command Preamble Field (CPF) - The number of coarse and fine time available in Command Elapsed Time registers should be mentioned in this field. Based on this preamble field the target will initialize or synchronise the local ET counter (only for target).

Table 474. 0x024 - CET0 - Command Elapsed Time 0

31	0
CET0	
0	
rw	

31: 0 Command Elapsed Time 0 (CET0) - Initialize or Synchronise local ET counter value (0 to 31).

Table 475. 0x028 - CET1 - Command Elapsed Time 1

31	0
CET1	
0	
rw	

31: 0 Command Elapsed Time 1 (CET1) - Initialize or Synchronise local ET counter value (32 to 63).

*Table 476.*0x02C - CET2 - Command Elapsed Time 2

31	0
CET2	
0	
rw	

31: 0 Command Elapsed Time 2 (CET2) - Initialize or Synchronise local ET counter value (64 to 95).

*Table 477.*0x030 - CET3 - Command Elapsed Time 3

31	0
CET3	
0	
rw	

31: 0 Command Elapsed Time 3 (CET3) - Initialize or Synchronise local ET counter value (96 to 127).

Table 478. 0x034 - CET4 - Command Elapsed Time 4

31 24	23 0
CET4	RESERVED
0	0
rw	r

31: 24 Command Elapsed Time 4 (CET4) - Initialize or Synchronise local ET counter value (128 to 135).

23: 0 RESERVED

*Table 479.*0x040 - DPF - Datation Preamble Field

31 16	15 0
RESERVED	DPF
0	0x2f00
r	r

31: 16 RESERVED

15: 0 Datation Preamble Field (DPF) - The number of coarse and fine time implemented can be obtained from this Preamble Field.

*Table 480.*0x044 - DET0 - Datation Elapsed Time 0

31	0
DET0	
0	

*Table 480.*0x044 - DET0 - Datation Elapsed Time 0

r

31: 0 Datation Elapsed Time 0 (DET0) - CCSDS Time Code value (0 to 31) of local ET counter value.

*Table 481.*0x048 - DET1 - Datation Elapsed Time 1

31	0
DET1	
0	
r	

31: 0 Datation Elapsed Time 1 (DET1) - CCSDS Time Code value (32 to 63) of local ET counter value.

Table 482. 0x060 - TRPFRX - Time-Stamp Preamble Field Rx

31	16	15	0
RESERVED		TRPF	
0		0x2f00	
r		r	

31: 16 RESERVED

15: 0 Time stamp Preamble Field (TRPF) - The number of coarse and fine time implemented can be obtained from this Preamble Field.

Table 483. 0x064 - TR0 - Time Stamp Elapsed Time 0 Rx

31	0
TR0	
0	
r	

31: 0 Time Stamp Elapsed Time 0 Rx (TR0) - Time stamped local ET value (0 To 31) when distributed interrupt received.

Table 484. 0x068 - TR1 - Time Stamp Elapsed Time 1 Rx

31	0
TR1	
0	
r	

31: 0 Time Stamp Elapsed Time 1 Rx (TR1) - Time stamped local ET value (32 to 63) when distributed interrupt received.

Table 485. 0x080 - TTPFTX - Time-Stamp SpaceWire Time-Code and Preamble Field Tx

31	24	23	16	15	0
TSTC		RESERVED		TTPF	
0		0		0x2f00	
rw		r		r	

31: 24 Time stamp time code (TSTC) - Time stamp on this time-code value, used for time stamping when this register matched with SpaceWire Time-Codes. The mask for this matching is available in configuration register 3.(only for initiator)

23: 16 RESERVED

15: 0 Time stamp Preamble Field (TTPF) - The number of coarse and fine time implemented can be obtained from this Preamble Field.

Table 486. 0x084 - TT0 - Time Stamp Elapsed Time 0 Tx

31	0
TT0	
0	
r	

31: 0 Time Stamp Elapsed Time 0 Tx (TT0) - Time stamped local ET value (0 to 31) when distributed interrupt transmitted.

Table 487. 0x088 - TT1 - Time Stamp Elapsed Time 1 Tx

31	0
TT1	
0	
r	

31: 0 Time Stamp Elapsed Time 1 Tx (TT1) - Time stamped local ET value (32 to 63) when distributed interrupt transmitted.

Table 488. 0x0A0 - LPF- Latency Preamble Field

31	16	15	0
RESERVED		LPF	
0		0x2f00	
r		r	

31: 16 RESERVED

15: 0 Latency Preamble Field (LPF) - The number of coarse and fine time implemented can be obtained from this Preamble Field.(only for target)

Table 489. 0xA4 - LE0 -Latency Elapsed Time 0

31	0
LE0	
0	
rw	

31: 0 Latency Elapsed Time Value 0 (LE0) - Latency Value (0 to 31) written by initiator.(only for target)

Table 490. 0xA8 - LE1 -Latency Elapsed Time 1

31	0
LE1	
0	
rw	

31: 0 Latency Elapsed Time Value 1 (LE1) - Latency Value (32 to 63) written by initiator.(only for target)

Table 491. 0x0C0 - IE - Interrupt Enable

31 20	19	18 11	10	9	8	7	6	5	4	3	2	1	0
-	NCTCE	-	SETE	EDIE3	EDIE2	EDIE1	EDIE0	DITE	DIRE	TTE	TME	TRE	SE
0	0	0	0	0									
r	rw	r	rw	rw									

31: 20 RESERVED

19: Non consecutive SpaceWire Time-Code received Interrupt Enable (NCTCE)

18: 11 RESERVED

10 Set ET External Interrupt Enable (SETE)

9 External Datation Interrupt Enable 3 (EDIE3)

8 External Datation Interrupt Enable 2 (EDIE2)

7 External Datation Interrupt Enable 1 (EDIE1)

6 External Datation Interrupt Enable 0 (EDIE0)

5 Distributed Interrupt Transmitted Interrupt Enable (DITE)

4 Distributed interrupt Received Interrupt Enable (DIRE)

3 Time-Code Transmitted Interrupt Enable (TTE) - SpaceWire Time-Code Transmitted Interrupt Enable (only for initiator)

2 Time Message transmit Interrupt Enable (TME) - (only for initiator)

1 Time-Code Received Interrupt Enable (TRE) - SpaceWire Time-Code Received Interrupt Enable (only for target)

0 Sync Interrupt Enable (SE) (only for target)

Table 492. 0xC4 - IS -Interrupt Status

31 20	19	18 11	10	9	8	7	6	5	4	3	2	1	0
RESERVED	NCTC	-	SET	EDI3	EDI2	EDI1	EDI0	DIT	DIR	TT	TM	TR	S
0	0	0	0	0									
r	wc	r	wc	wc									

31: 20 RESERVED

19: Generated when Non consecutive SpaceWire Time-Code is received (NCTC)

18: 11 RESERVED

10 Generated when Elapsed Time is loaded with contents of the Command Field register based on external enable signal (SET).

9 External Datation Interrupt 3 (EDI3) - Generated when conditions for External Datation 3 is matched.

8 External Datation Interrupt 2 (EDI2) - Generated when conditions for External Datation 2 is matched.

7 External Datation Interrupt 1 (EDI1) - Generated when conditions for External Datation 1is matched.

6 External Datation Interrupt 0 (EDI0) - Generated when conditions for External Datation 0 is matched.

5 Distributed Interrupt Transmitted (DIT) - Generated when distributed interrupt is transmitted (Latency calculation should be enabled)

4 Distributed interrupt Received (DIR) - Generated when distributed interrupt is Received (Latency calculation should be enabled)

3 Time-Codes Transmitted (TT) - Generated when SpaceWire Time-Codes is transmitted (only for initiator)

Table 492. 0xC4 - IS -Interrupt Status

2	Transmit Time Message (TM) - Generated when the conditions for transmitting time message occurred, based on this time message should be transmitted from initiator (only for initiator)
1	Time-Code Received (TR) - Generated when SpaceWire Time-Code is received (only for target)
0	Target initialized or synchronized (S) - Generated when the target is initialized or synchronized with initiator (only for target)

Table 493. 0xC8 - DC - Delay Count

31 15	14 0
-	DC
0	0x7FFF
r	rw

31: 15	RESERVED
14: 0	Delay Count (DC) - Delay induced between SpaceWire Time-Codes and Distributed Interrupt transmission in system clock units. The delay introduced is the value in this register multiplied by the system clock. (only for initiator)0x7FFF

Table 494. 0xCC - DS - Disable Sync

31	30 24	23 0
EN	-	CD
0	0	0xFFFFFF
rw	r	rw

31:	Enable for Configurable delay (EN)
30: 24	RESERVED
23: 0	Configurable delay (CD) to capture missing SpaceWire Time-Code (only for target) The INSYNC bit in the Status 0 register will disable itself when an expected SpaceWire Time-Code is not arrived after the delay mentioned in this register. The delay corresponds to the fine time of Elapsed Time counter and should not overlap with the MAPPING register. Any Overlapping register must also be set to Zero.

Table 495. 0x100 - EDM0 - External Datation 0 Mask

31 0
EDM0
0x00000000
rw

31: 0	External Datation Mask (EDM0) - External datation can be enabled by writing '1' into the bit for that corresponding external input. When conditions are matched the Elapsed Time will be latched. The latched values are available at External Datation 0 Time Register. All the mask bits will go low after any one of the conditions with respect to the enabled mask bits.are matched.

Table 496. 0x110 - EDPF0 - External Datation 0 Preamble Field

31 16	15 0
-	EDPF0
0	0x2f00
r	r

31: 16	RESERVED
15: 0	External Datation Preamble Field (EDPF0) - The number of coarse and fine time implemented can be obtained from this Preamble Field.

*Table 497.*0x114 - ED0ET0 - External Datation 0 Elapsed Time 0

31	0
ED0ET0	
0	
r	

31: 0 External Datation Elapsed Time 0 (ED0ET0) - Latched CCSDS Time Code value (0 to 31) of local ET counter.

*Table 498.*0x118 - ED0ET1 - External Datation 0 Elapsed Time 1

31	0
ED0ET1	
0	
r	

31: 0 External Datation Elapsed Time 1 (ED0ET1) - Latched CCSDS Time Code value (32 to 63) of local ET counter.

*Table 499.*0x1A0-0x1B4 - PDR0 to PDR5 - Pulse Definition Register 0 to 5

31 24	23 20	19 16	15 11	10	9 2	1	0
RESERVED	PP	PW	RESERVED	PL	RESERVED	PE	R
0	0	0	0	1	0	0	0
r	rw	rw	r	rw	r	rw	r

31: 24 RESERVED

23: 20 PP Pulse Period
Value ‘0000’ = 2^{7} seconds
Value ‘0001’ = 2^{6} seconds
...
Value ‘1110’ = 2^{-7} seconds
Value ‘1111’ = 2^{-8} seconds
Period = $2^{(7-PP)}$
Frequency = $2^{-(7-PP)}$

19: 16 PW Pulse Width
Value ‘0000’ = 2^{6} seconds
Value ‘0001’ = 2^{5} seconds
...
Value ‘1110’ = 2^{-8} seconds
Value ‘1111’ = 2^{-9} seconds
Width = $2^{(6-PW)}$

15: 11 RESERVED

10: PL Pulse Level: Defines logical level of active part of pulse output. ‘0’ = Low, ‘1’ = High

9: 2 RESERVED

1: PE Pulse Enable: ‘0’ = disabled, ‘1’ = enabled

0: RESERVED

Note:

- Reserved register fields should be written as zeros and masked out on read.
- The registers which are not mentioned either as only for initiator or target are used in both initiator and target.
- The Definition of External Datation 1 Mask, External Datation 2 Mask and External Datation 3 Mask registers are exactly same as External Datation 0 Mask Register.
- The Definition of External Datation 1 Time, External Datation 2 Time and External Datation 3 Time registers are exactly same as External Datation 0 Time Registers (i.e. External Datation 0 Preamble Field and External Datation 0 Elapsed Time 0,1,2,3,4).

35 General Purpose Timer Unit with Watchdog

35.1 Overview

The General Purpose Timer Unit provides a common prescaler and 7 decrementing timers. The unit is capable of asserting interrupts on timer underflow. The timer also provides the system with watchdog functionality. The watchdog is of window-watchdog type i.e. the watchdog timer can be configured to have a lower boundary and for how often the watchdog timer can be triggered or reloaded by the software.

.

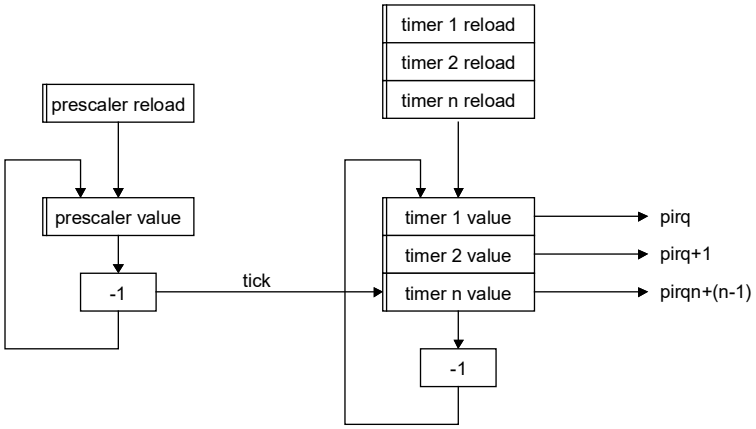


Figure 80. General Purpose Timer Unit block diagram

35.2 Operation

The prescaler is clocked by the system clock and decremented on each clock cycle. When the prescaler underflows, it is reloaded from the prescaler reload register and a timer tick is generated.

The operation of each timers is controlled through its control register. A timer is enabled by setting the enable bit in the control register. The timer value is then decremented on each prescaler tick. When a timer underflows, it will automatically be reloaded with the value of the corresponding timer reload register if the restart bit in the control register is set, otherwise it will stop at -1 and reset the enable bit.

The shared interrupt will be raised when any of the timers with interrupt enable bit underflows. The timer unit will signal an interrupt on appropriate line when a timer underflows (if the interrupt enable bit for the current timer is set). The interrupt pending bit in the control register of the underflown timer will be set and remain set until cleared by writing '1'.

To minimize complexity, timers share the same decrementer. This means that the minimum allowed prescaler division factor is *8* (reload register = *7*) where *7* is the number of timers. By setting the chain bit in the control register timer *n* can be chained with preceding timer *n*-1. Timer *n* will be decremented each time when timer *n*-1 underflows.

Each timer can be reloaded with the value in its reload register at any time by writing a 'one' to the load bit in the control register. The last timer acts as a watchdog, asserting the external RESET_OUT_N output signal when expired. The watchdog timer also implements a window functionality. This enables a decrementing counter which reloads each time the timer is reloaded. If the timer is reloaded and the window counter hasn't reach zero, this will also assert the RESET_OUT_N output.

Each timer can be configured to latch its value to a dedicated register when an event is detected on the interrupt. All timers can be forced to reload when an event is detected on the interrupt bus A dedicated mask register is provided to filter the interrupts.

Simultaneous start of multiple timers are supported via timer configuration register CONFIG.TIMEREN. To simultaneously start two or more counters set the corresponding bits in the register CONFIG.TIMEREN.

At reset, all timer are disabled except the watchdog timer. The prescaler value and reload registers are set to all ones, while the watchdog timer 7 is set to 0xFFF. All other registers are uninitialized except for the WDOGDIS and WDOGNMI fields that are reset to '0'.

35.2.1 Window-watchdog

The watchdog has an optional lower boundary for reloading the watchdog timer. Application can optionally specify the least number of system clock cycles between 2 reload events of the watchdog timer.

When a watchdog window is programmed, an early watchdog reload is also treated as a watchdog event. This allows preventing situations where a system failure may still reload the watchdog. For example, application code could be stuck in an interrupt service that contains a watchdog feed. Setting the window such that this would result in an early reload will generate a watchdog event, allowing for system recovery.

The watchdog window functionality is enabled when the bit field WDOGWINC is greater than 0x0 and smaller or equal to 0xFFFF. The programmed number specify the number of system clock cycles between 2 watchdog reload events.

35.3 Registers

The core is programmed through registers mapped into APB address space. The number of implemented registers depend on the number of implemented timers.

*Table 500.*General Purpose Timer Unit registers

APB address offset	Register
0x80003000	Scaler value
0x80003004	Scaler reload value
0x80003008	Configuration register
0x8000300C	Timer latch configuration register
0x80003010	Timer 1 counter value register
0x80003014	Timer 1 reload value register
0x80003018	Timer 1 control register
0x8000301C	Timer 1 latch register
0x80003020	Timer *2* counter value register
0x80003024	Timer *2* reload value register
0x80003028	Timer *2* control register
0x8000302C	Timer *2* latch register
0x80003030	Timer *3* counter value register
0x80003034	Timer *3* reload value register
0x80003038	Timer *3* control register
0x8000303C	Timer *3* latch register
0x80003040	Timer *4* counter value register
0x80003044	Timer *4* reload value register
0x80003048	Timer *4* control register
0x8000304C	Timer *4* latch register
0x80003050	Timer *5* counter value register
0x80003054	Timer *5* reload value register
0x80003058	Timer *5* control register
0x8000305C	Timer *5* latch register
0x80003060	Timer *6* counter value register
0x80003064	Timer *6* reload value register
0x80003068	Timer *6* control register
0x8000306C	Timer *6* latch register
0x80003070	Timer *7* counter value register
0x80003074	Timer *7* reload value register
0x80003078	Timer *7* control register
0x8000307C	Timer *7* latch register

35.3.1 Scaler Value Register

*Table 501.*0x00 - SCALER - Scaler value register

31 16	16-1 0
RESERVED	SCALER
0	all 1
r	rw

16-1: 0 Scaler value. This value will also be set by writes to the Scaler reload value register.
Any unused most significant bits are reserved. Always reads as ‘000...0’.

35.3.2 Scaler Reload Value Register

*Table 502.*0x04 - SRELOAD - Scaler reload value register

31 16	16-1 0
RESERVED	SCALER RELOAD VALUE
0	all 1
r	rw

16-1: 0 Scaler reload value. Writes to this register also set the scaler value.
Any unused most significant bits are reserved. Always read as ‘000...0’.

35.3.3 Configuration Register

*Table 503.*0x08 - CONFIG - Configuration register

31 23	22 16	15 14	13	12	11	10	9	8	7 3	2 0
RESERVED	TIMEREN	R	EV	ES	EL	EE	DF	SI	RESERVED	TIMERS
0	0	0	0	0	0	0	0	1	*	7
r	rw	r	rw	rw	rw	rw	rw	r	r	r

31: 23 Reserved. Always reads as ‘000...0’.

22: 16 Enable bits for each timer. Writing ‘1’ to one of this bits sets the enable bit in the corresponding timers control register. Writing ‘0’ has no effect to the timers. bit[16] corresponds to timer0, bit[17] to timer 1,...

15: 14 Reserved

13 External Events (EV). If EV is set to 0 then the latch and set events are taken from the least significant 32 bit of the interrupt bus, otherwise they are from some of the most significant ones and some external signals (see table 35.3.4)

12 Enable set (ES). If set, on the next matching interrupt, the timers will be loaded with the corresponding timer reload values. The bit is then automatically cleared, not to reload the timer values until set again.

11 Enable latching (EL). If set, on the next matching interrupt, the latches will be loaded with the corresponding timer values. The bit is then automatically cleared, not to load a timer value until set again.

10 Enable external clock source (EE). If set the prescaler is clocked from the external clock source.

9 Disable timer freeze (DF). If set the timer unit can not be freezed, otherwise signal GPTI.DHALT freezes the timer unit.

8 Separate interrupts (SI). Reads ‘1’ if the timer unit generates separate interrupts for each timer, otherwise ‘0’. Read-only.

7: 3 Reserved

2: 0 Number of implemented timers. Read-only.

35.3.4 Timer Latch Configuration Register

*Table 504.*0x0C - CATCHCFG - Timer latch configuration register

31 0
LATCHSEL
0
rw

31: 0 This field specifies which bits of the interrupt bus or of the external signals (depending on EV field in table 35.3.3) cause the set and latch events. If EV is 0, the latching is done based on events on the 31:0 bits of the interrupt bus with a direct mapping. If the EV field is ‘1’, the bits 29:0 correspond to the 61:32 bits of the interrupt bus, while the bit 30 corresponds to the TICKOUT signal from the SpaceWire Interface (see chapter 33) and the bit 31 corresponds to the rtsync signal from the MIL-STD-1553B / AS15531 Interface (see chapter 23).

35.3.5 Timer N Counter Value Register

*Table 505.*0xn0, when n selects the times - TCNTVALn - Timer n counter value register

32-1 0
TCVAL
0
rw

32-1: 0 Timer Counter value. Decremented by 1 for each prescaler tick.

Any unused most significant bits are reserved. Always reads as ‘000...0’.

35.3.6 Timer N Reload Value Register

*Table 506.*0xn4, when n selects the times - TRLDVALn - Timer n reload value register

32-1	0
TRCDUAL	
*	
rw	

32-1: 0 Timer Reload value. This value is loaded into the timer counter value register when '1' is written to load bit in the timers control register or when the RS bit is set in the control register and the timer underflows.

Any unused most significant bits are reserved. Always reads as '000...0'.

35.3.7 Timer N Control Register

*Table 507.*0xn8, when n selects the times - TCTRLn - Timer n control register

31	16	15	9	8	7	6	5	4	3	2	1	0
WDOGWINC		RESERVED		WS	WN	DH	CH	IP	IE	LD	RS	EN
0		0		0	0	0	0	0	0	0	*	*
rw		r		rw	rw	r	rw	wc	wc	rw	rw	rw

31: 16 Reload value for the watchdog window counter. The window counter is reloaded with this value each time the watchdog counter is reloaded. This register is only available for counter timer 7 in timer unit #1.

15: 9 Reserved. Always reads as '000...0'.

8 Disable Watchdog Output (WS/WDOGDIS): If this field is set to '1' then the GPTO.WDOG and GPTO.WDOGN outputs are disabled (fixed to '0' and '1' respectively). This functionality is only available for the last timer.

7 Enable Watchdog NMI (WN/WDOGNMI): If this field is set to '1' then the watchdog timer will also generate a non-maskable interrupt (15) when an interrupt is signalled. This functionality is only available for the last timer

6 Debug Halt (DH): Value of GPTI.DHALT signal which is used to freeze counters (e.g. when a system is in debug mode). Read-only.

5 Chain (CH): Chain with preceding timer. If set for timer *n*, timer *n* will be decremented each time when timer (*n*-1) underflows.

4 Interrupt Pending (IP): The core sets this bit to '1' when an interrupt is signalled. This bit remains '1' until cleared by writing '1' to this bit, writes of '0' have no effect.

3 Interrupt Enable (IE): If set the timer signals interrupt when it underflows.

2 Load (LD): Load value from the timer reload register to the timer counter value register.

1 Restart (RS): If set, the timer counter value register is reloaded with the value of the reload register when the timer underflows

0 Enable (EN): Enable the timer.

35.3.8 Timer N Latch Register

*Table 508.*0xnC, when n selects the times - TLATCHn - Timer n latch register

31	0
LTCV	
0	
r	

31: 0 Latched timer counter value (LTCV): Valued latched from corresponding timer. Read-only.

36 General Purpose Timer Unit (Secondary)

36.1 Overview

The secondary LEON3FT microcontroller have 2 General Purpose Timer Units. The General Purpose Timer Unit provides a common prescaler and 7 decrementing timers. The unit is capable of asserting interrupts on timer underflow.

.

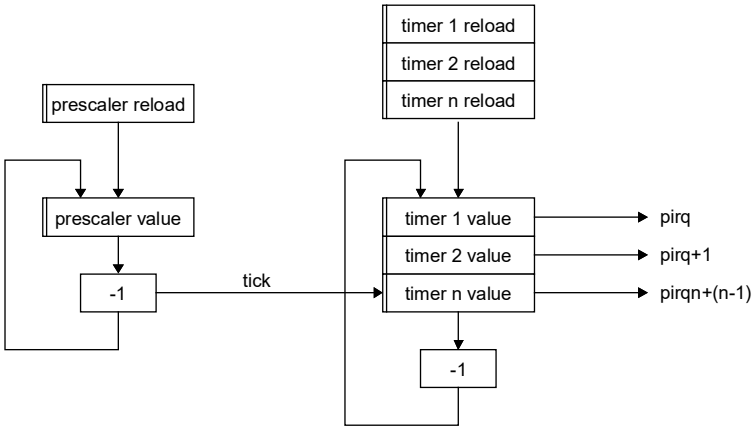


Figure 81. General Purpose Timer Unit block diagram

36.2 Operation

The prescaler is clocked by the system clock and decremented on each clock cycle. When the prescaler underflows, it is reloaded from the prescaler reload register and a timer tick is generated.

The operation of each timers is controlled through its control register. A timer is enabled by setting the enable bit in the control register. The timer value is then decremented on each prescaler tick. When a timer underflows, it will automatically be reloaded with the value of the corresponding timer reload register if the restart bit in the control register is set, otherwise it will stop at -1 and reset the enable bit.

The shared interrupt will be raised when any of the timers with interrupt enable bit underflows. The timer unit will signal an interrupt on appropriate line when a timer underflows (if the interrupt enable bit for the current timer is set). The interrupt pending bit in the control register of the underflown timer will be set and remain set until cleared by writing '1'.

To minimize complexity, timers share the same decrementer. This means that the minimum allowed prescaler division factor is *8* (reload register = *7*) where *7* is the number of timers. By setting the chain bit in the control register timer *n* can be chained with preceding timer *n*-1. Timer *n* will be decremented each time when timer *n*-1 underflows.

Each timer can be reloaded with the value in its reload register at any time by writing a 'one' to the load bit in the control register.

Each timer can be configured to latch its value to a dedicated register when an event is detected on the interrupt. All timers can be forced to reload when an event is detected on the interrupt bus A dedicated mask register is provided to filter the interrupts.

Simultaneous start of multiple timers are supported via timer configuration register CONFIG.TIMEREN. To simultaneously start two or more counters set the corresponding bits in the register CONFIG.TIMEREN.

36.3 Registers

The core is programmed through registers mapped into APB address space. The number of implemented registers depend on the number of implemented timers.

*Table 509.*General Purpose Timer Unit registers

APB address offset	Register
0x80004000	Scaler value
0x80004004	Scaler reload value
0x80004008	Configuration register
0x8000400C	Timer latch configuration register
0x80004010	Timer 1 counter value register
0x80004014	Timer 1 reload value register
0x80004018	Timer 1 control register
0x8000401C	Timer 1 latch register
0x80004020	Timer *2* counter value register
0x80004024	Timer *2* reload value register
0x80004028	Timer *2* control register
0x8000402C	Timer *2* latch register
0x80004030	Timer *3* counter value register
0x80004034	Timer *3* reload value register
0x80004038	Timer *3* control register
0x8000403C	Timer *3* latch register
0x80004040	Timer *4* counter value register
0x80004044	Timer *4* reload value register
0x80004048	Timer *4* control register
0x8000404C	Timer *4* latch register
0x80004050	Timer *5* counter value register
0x80004054	Timer *5* reload value register
0x80004058	Timer *5* control register
0x8000405C	Timer *5* latch register
0x80004060	Timer *6* counter value register
0x80004064	Timer *6* reload value register
0x80004068	Timer *6* control register
0x8000406C	Timer *6* latch register
0x80004070	Timer *7* counter value register
0x80004074	Timer *7* reload value register
0x80004078	Timer *7* control register
0x8000407C	Timer *7* latch register

36.3.1 Scaler Value Register

*Table 510.*0x00 - SCALER - Scaler value register

31 16	16-1 0
RESERVED	SCALER
0	all 1
r	rw

16-1: 0 Scaler value. This value will also be set by writes to the Scaler reload value register.
Any unused most significant bits are reserved. Always reads as ‘000...0’.

36.3.2 Scaler Reload Value Register

*Table 511.*0x04 - SRELOAD - Scaler reload value register

31 16	16-1 0
RESERVED	SCALER RELOAD VALUE
0	all 1
r	rw

16-1: 0 Scaler reload value. Writes to this register also set the scaler value.
Any unused most significant bits are reserved. Always read as ‘000...0’.

36.3.3 Configuration Register

*Table 512.*0x08 - CONFIG - Configuration register

31 23	22 16	15 14	13	12	11	10	9	8	7 3	2 0
"000..0"	TIMEREN	R	EV	ES	EL	EE	DF	SI	RESERVED	TIMERS
0	0	0	0	0	0	0	0	1	*	7
r	rw	r	rw	rw	rw	rw	rw	r	r	r

31: 23 Reserved. Always reads as '000...0'.

22: 16 Enable bits for each timer. Writing '1' to one of this bits sets the enable bit in the corresponding timers control register. Writing '0' has no effect to the timers. bit[16] corresponds to timer0, bit[17] to timer 1,...

15: 14 Reserved

13 External Events (EV). If EV is set to 0 then the latch and set events are taken from the least significant 32 bit of the interrupt bus, otherwise they are from some of the most significant ones and some external signals (see table 35.3.4).

12 Enable set (ES). If set, on the next matching interrupt, the timers will be loaded with the corresponding timer reload values. The bit is then automatically cleared, not to reload the timer values until set again.

11 Enable latching (EL). If set, on the next matching interrupt, the latches will be loaded with the corresponding timer values. The bit is then automatically cleared, not to load a timer value until set again.

10 Enable external clock source (EE). If set the prescaler is clocked from the external clock source.

9 Disable timer freeze (DF). If set the timer unit can not be freezed, otherwise signal GPTI.DHALT freezes the timer unit.

8 Separate interrupts (SI). Reads '1' if the timer unit generates separate interrupts for each timer, otherwise '0'. Read-only.

7: 3 Reserved

2: 0 Number of implemented timers. Read-only.

36.3.4 Timer Latch Configuration Register

*Table 513.*0x0C - CATCHCFG - Timer latch configuration register

31 0
LATCHSEL
0
rw

31: 0 This field specifies which bits of the interrupt bus or of the external signals (depending on EV field in table 35.3.3) cause the set and latch events. If EV is 0, the latching is done based on events on the 31:0 bits of the interrupt bus with a direct mapping. If the EV field is '1', the bits 29:0 correspond to the 61:32 bits of the interrupt bus, while the bit 30 corresponds to the TICKOUT signal from the SpaceWire Interface (see chapter 33) and the bit 31 corresponds to the rtsync signal from the MIL-STD-1553B / AS15531 Interface (see chapter 23).

36.3.5 Timer N Counter Value Register

*Table 514.*0xn0, when n selects the times - TCNTVALn - Timer n counter value register

32-1 0
TCVAL
0
rw

32-1: 0 Timer Counter value. Decremented by 1 for each prescaler tick.

Any unused most significant bits are reserved. Always reads as '000...0'.

36.3.6 Timer N Reload Value Register

*Table 515.*0xn4, when n selects the times - TRLDVALn - Timer n reload value register

32-1	0
TRCDUAL	
*	
rw	

32-1: 0 Timer Reload value. This value is loaded into the timer counter value register when '1' is written to load bit in the timers control register or when the RS bit is set in the control register and the timer underflows.

Any unused most significant bits are reserved. Always reads as '000...0'.

36.3.7 Timer N Control Register

*Table 516.*0xn8, when n selects the times - TCTRLn - Timer n control register

31	9	8	7	6	5	4	3	2	1	0
RESERVED				DH	CH	IP	IE	LD	RS	EN
0				0	0	0	0	0	*	*
r				r	rw	wc	wc	rw	rw	rw

31: 7 Reserved. Always reads as '000...0'.

6 Debug Halt (DH): Value of GPTI.DHALT signal which is used to freeze counters (e.g. when a system is in debug mode). Read-only.

5 Chain (CH): Chain with preceding timer. If set for timer *n*, timer *n* will be decremented each time when timer (*n*-1) underflows.

4 Interrupt Pending (IP): The core sets this bit to '1' when an interrupt is signalled. This bit remains '1' until cleared by writing '1' to this bit, writes of '0' have no effect.

3 Interrupt Enable (IE): If set the timer signals interrupt when it underflows.

2 Load (LD): Load value from the timer reload register to the timer counter value register.

1 Restart (RS): If set, the timer counter value register is reloaded with the value of the reload register when the timer underflows

0 Enable (EN): Enable the timer.

36.3.8 Timer N Latch Register

*Table 517.*0xnC, when n selects the times - TLATCHn - Timer n latch register

31	0
LTCV	
0	
r	

31: 0 Latched timer counter value (LTCV): Valued latched from corresponding timer. Read-only.

37 I²C to AHB bridge

The GR716B microcontroller comprises an I2C to AHB bridge (I2C2AHB). The I2C to AHB bridge controls its own external pins and has a unique AMBA address described in chapter 2.10. The I2C to AHB bridge is connected to external pins via the IOMUX.

The control and status registers are located on APB bus in the address range from 0x80105000 to 0x80105FFF. See I2C to AHB bridge connections in the next drawing. The figure shows memory locations and functions used for I2C2AHB configuration and control.

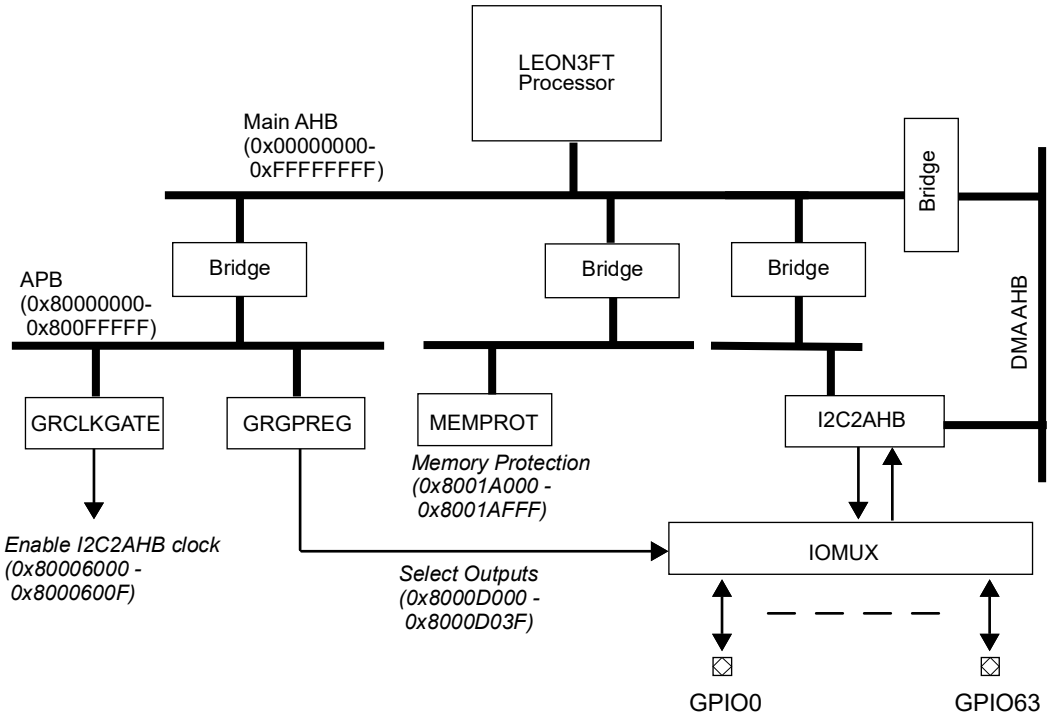


Figure 82. GR716B I2C2AHB bus and pin

The primary clock gating unit **GRCLKGATE** described in section 27 is used to enable/disable the I2C to AHB bridge. The unit **GRCLKGATE** can also be used to perform reset of the I2C to AHB bridge. Software must enable clock and release reset described in section 27 before configuration and transmission can start.

External IO selection and configuration is made in the system IO configuration registers (**GRGPREG**) in the address range from 0x8000D000 to 0x8000D03F. See section 7.1 for further information.

The system can be configured to protect and restrict access to the I2C to AHB bridge in the **MEMPROT** unit. See section 47 for more information.

37.1 Overview

The I²C slave to AHB bridge is an I²C slave that provides a link between the I²C bus and AMBA AHB. The core is compatible with the Philips I²C standard and external pull-up resistors must be supplied for both bus lines.

On the I²C bus the slave acts as an I²C memory device where accesses to the slave are translated to AMBA accesses. The core can translate I²C accesses to AMBA byte, halfword or word accesses. The core makes use of I²C clock stretching but can also be configured to use a special mode without clock

stretching in order to support systems where master or physical layer limitations prevent stretching of the I^2C clock period.

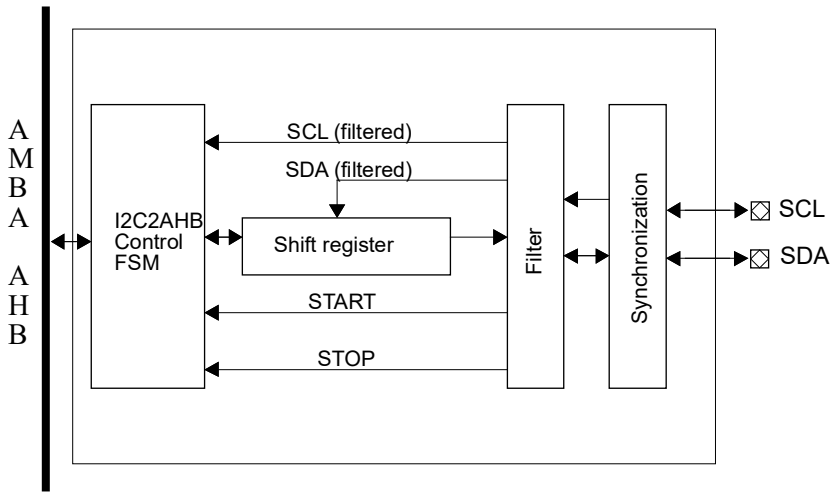


Figure 83. Block diagram

37.2 Operation

37.2.1 Transmission protocol

The I^2C-bus is a simple 2-wire serial multi-master bus with collision detection and arbitration. The bus consists of a serial data line (SDA) and a serial clock line (SCL). The I^2C standard defines three transmission speeds; Standard (100 kb/s), Fast (400 kb/s) and High speed (3.4 Mb/s).

A transfer on the I^2C-bus begins with a START condition. A START condition is defined as a high to low transition of the SDA line while SCL is high. Transfers end with a STOP condition, defined as a low to high transition of the SDA line while SCL is high. These conditions are always generated by a master. The bus is considered to be busy after the START condition and is free after a certain amount of time following a STOP condition. The bus free time required between a STOP and a START condition is defined in the I^2C-bus specification and is dependent on the bus bit rate.

Figure 84 shows a data transfer taking place over the I^2C-bus. The master first generates a START condition and then transmits the 7-bit slave address. The bit following the slave address is the R/$\overline{W}$ bit which determines the direction of the data transfer. In this case the R/$\overline{W}$ bit is zero indicating a write operation. After the master has transmitted the address and the R/$\overline{W}$ bit it releases the SDA line. The receiver pulls the SDA line low to acknowledge the transfer. If the receiver does not acknowledge the transfer, the master may generate a STOP condition to abort the transfer or start a new transfer by generating a repeated START condition.

After the address has been acknowledged the master transmits the data byte. If the R/$\overline{W}$ bit had been set to ‘1’ the master would have acted as a receiver during this phase of the transfer. After the data byte has been transferred the receiver acknowledges the byte and the master generates a STOP condition to complete the transfer.

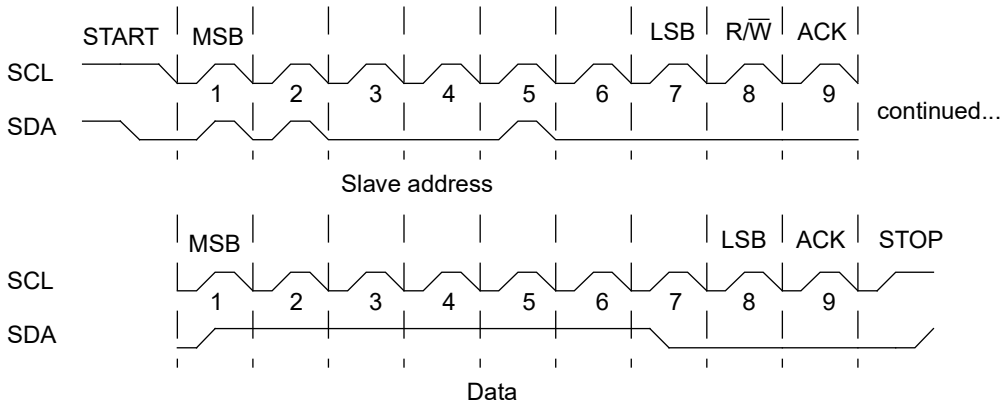


Figure 84. Complete I²C data transfer

If the data bit rate is too high for a slave device or if the slave needs time to process data, it may stretch the clock period by keeping SCL low after the master has driven SCL low. Clock stretching is a configurable parameter of the core (see sections 37.2.4 and 37.2.6).

37.2.2 Slave addressing

The core responds to two addresses on the I²C bus. Accesses to the I2C memory address are translated to AMBA AHB accesses and accesses to the I²C configuration address access the core's configuration register. I2C memory and slave addresses can be configured via control registers see register SLVADDR and SLVCFG in section 37.3.5 and 37.3.6.

37.2.3 System clock requirements and sampling

The core samples the incoming I²C SCL clock and does not introduce any additional clock domains into the system. Both the SCL and SDA lines first pass through two stage synchronizers and are then filtered with a low pass filter consisting of four registers.

START and STOP conditions are detected if the SDA line, while SCL is high, is at one value for two system clock cycles, toggles and keeps the new level for two system clock cycles.

The synchronizers and filters constrain the minimum system frequency. The core requires the SCL signal to be stable for at least four system clock cycles before the core accepts the SCL value as the new clock value. The core's reaction to transitions will be additionally delayed since both lines are taken through two-stage synchronizers before they are filtered. Therefore it takes the core over eight system clock cycles to discover a transition on SCL.

37.2.4 Configuration register access

The I²C configuration register is accessed via a separate I²C address (I²C configuration address). The configuration register has the layout shown in table 518.

*Table 518.*I2C2AHB configuration register

7	6	5	4	3	2	1	0
Reserved		PROT	MEXC	DMAACT	NACK	HSIZE	

7:6	Reserved, always zero (read only)
5	Memory protection triggered (PROT) - '1' if last AHB access was outside the allowed memory area. Updated after each AMBA access (read only)
4	Memory exception (MEXC) - '1' if core receives AMBA ERROR response. Updated after each AMBA access (read only)
3	DMA active (DMAACT) - '1' if core is currently performing a DMA operation.

*Table 518.*I2C2AHB configuration register

2	NACK (NACK) - Use NACK instead of clock stretching. See documentation in section 37.2.6.
1:0	AMBA access size (HSIZE) - Controls the access size that the core will use for AMBA accesses. 0: byte, 1: halfword, 2: word. HSIZE = “11” is illegal.

Reset value: 0x02

Reads from the I^2C configuration address will return the current value of the configuration register. Writes to the I^2C configuration address will affect the writable bits in the configuration register.

37.2.5 AHB accesses

All AMBA accesses are done in big endian format. The first byte sent to or from the slave is the most significant byte.

To write a word on the AHB bus the following I2C bus sequence should be performed:

1. Generate START condition
2. Send I2C memory address with the R/$\overline{W}$ bit set to ‘0’.
3. Send four byte AMBA address, the most significant byte is transferred first
4. Send four bytes to write to the specified address
5. If more than four consecutive bytes should be written, continue to send additional bytes, otherwise go to 6.
6. Generate STOP condition

To perform a read access on the AHB bus, the following I2C bus sequence should be performed:

1. Generate START condition
2. Send I2C memory address with the R/$\overline{W}$ bit set to ‘0’.
3. Send four byte AMBA address, the most significant byte is transferred first
4. Generate (repeated) START condition
5. Send I2C memory address with the R/$\overline{W}$ bit set to ‘1’.
6. Read the required number of bytes and NACK the last byte
7. Generate stop condition

During consecutive read or write operations, the core will automatically increment the address. The access size (byte, halfword or word) used on AHB is set via the HSIZE field in the I2C2AHB configuration register.

The core always respects the access size specified via the HSIZE field. If a write operation writes fewer bytes than what is required to do an access of the specified HSIZE then the write data will be dropped, no access will be made on AHB. If a read operation reads fewer bytes than what is specified by HSIZE then the remaining read data will be dropped at a START or STOP condition. This means, for instance, that if HSIZE is “10” (word) the core will perform two word accesses if a master reads one byte, generates a repeated start condition, and reads one more byte. Between these two accesses the address will have been automatically increased, so the fist access will be to address *n* and the second to address *n+4*.

The automatic address increment means that it is possible to write data and then immediately read the data located at the next memory position. As an example, the following sequence will write a word to address 0 and then read a word from address 4:

1. Generate START condition
2. Send I2C memory address with the R/$\overline{W}$ bit set to ‘0’.
3. Send four byte AMBA address, all zero.
4. Send four bytes to write to the specified address

5. Generate (repeated) START condition
6. Send I2C memory address with the R/$\overline{W}$ bit set to ‘1’.
7. Read the required number of bytes and lack the last byte
8. Generate stop condition

The core will not mask any address bits. Therefore it is important that the I^2C master respects AMBA rules when performing halfword and word accesses. A halfword access must be aligned on a two byte address boundary (least significant bit of address must be zero) and a word access must be aligned on a four byte boundary (two least significant address bits must be zero).

37.2.6 Clock stretching or NACK mode

The core has two main modes of operation for AMBA accesses. In one mode the core will use clock stretching while performing an AHB operation and in the other mode the core will not acknowledge bytes (abort the I^2C access) when the core is busy. Clock stretching is the preferred mode of operation. The NACK mode can be used in scenarios where the I^2C master or physical layer does not support clock stretching. The mode to use is selected via the NACK field in the I^2C configuration register.

When clock stretching is enabled (NACK field is ‘0’) the core will stretch the clock when the slave is accessed (via the I2C memory address) and the slave is busy processing a transfer. Clock stretching is also used when a data byte has been transmitted, or received, to keep SCL low until a DMA operation has completed. In the transmit (AMBA read) case SCL is kept low before the rising edge of the first byte. In the receive case (AMBA write) the ACK cycle for the previous byte is stretched.

When clock stretching is disabled (NACK field is ‘1’) the core will never stretch the SCL line. If the core is busy performing DMA when it is addressed, the address will not be acknowledged. If the core performs consecutive writes and the first write operation has not finished the core will now acknowledge the written byte. If the core performs a read operation and the read DMA operation has not finished when the core is supposed to deliver data then the core will go to its idle state and not respond to more accesses until a START condition is generated on the bus. This last part means that the NACK mode is practically unusable in systems where the AMBA access can take longer than one I^2C clock period. This can be compensated by using a very slow I^2C clock.

37.2.7 Memory protection

Default configuration allows full access to the complete AHB address range. The access range can be restricted via configuration registers.

The registers PADDR and PMASK are used to assign the memory protection area’s address and mask in the following way

Before the core performs an AMBA access it will perform the check:

(((incoming address) xor (PADDR)) and PMASK) /= 0x00000000

If the above expression is true (one or several bits in the incoming address differ from the protection address, and the corresponding mask bits are set to ‘1’) then the access is inhibited. As an example, assume that *PADDR* is 0xA0000000 and *PMASK* is 0xF0000000. Since *PMASK* only has ones in the most significant nibble, the check above can only be triggered for these bits. The address range of allowed accessed will thus be 0xA0000000 - 0xAFFFFFFF.

The core will set the configuration register bit PROT if an access is attempted outside the allowed address range. This bit is updated on each AHB access and will be cleared by an access inside the allowed range.

37.3 Registers

The core is programmed through registers mapped into APB address space.

*Table 519.*I^{2}C slave registers

APB address offset	Register
0x00	Control register
0x04	Status register
0x08	Protection address register
0x0C	Protection mask register
0x10	I2C slave memory address register
0x14	I2C slave configuration address register

37.3.1 Control Register

*Table 520.*0x0 - CTRL - Control register

31	2	1	0
RESERVED		IRQEN	EN
0		0	1
r		rw	rw

31: 2 RESERVED

1 Interrupt enable (IRQEN) - When this bit is set to '1' the core will generate an interrupt each time the DMA field in the status register transitions from '0' to '1'.

0 Core enable (EN) - When this bit is set to '1' the core is enabled and will respond to I2C accesses. Otherwise the core will not react to I2C traffic.

37.3.2 Status Register

*Table 521.*0x04 - STAT - Status register

31	3	2	1	0
RESERVED		PROT	WR	DMA
0x0		0	0	0
r		wc	r	wc

31: 3 RESERVED

2 Protection triggered (PROT) - Full access is granted the I2C2AHB interface

1 Write access (WR) - Last AHB access performed was a write access. This bit is read only.

0 Direct Memory Access (DMA) - This bit gets set to '1' each time the core attempts to perform an AHB access. By setting the IRQEN field in the control register this condition can generate an interrupt. This bit can be cleared by software by writing '1' to this position.

37.3.3 Protection Address Register

*Table 522.*0x08 - PADDR - Protection address register

31	0
PROTADDR	
0x0	
rw	

31 : 0 Protection address (PROTADDR) - Defines the base address for the memory area where the core is allowed to make accesses.

37.3.4 Protection Mask Register

*Table 523.*0x0C - PMASK - Protection mask register

31	0
PROTMASK	
0x0	
rw	

31 : 0 Protection mask (PROTMASK) - Selects which bits in the Protection address register that are used to define the protected memory area.

37.3.5 I2C Slave Memory Address Register

Table 524. 0x10 - SLVADDR - I2C slave memory address register

31 7	6 0
RESERVED	I2CSLVADDR
0	*
r	rw

31 : 7 RESERVED

6 : 0 I2C slave memory address (I2CSLVADDR) - Address that slave responds to for AHB memory accesses. The reset value is b101*xxx*0 where the three bits "*xxx*" are taken from the node ID bootstrap pins (GPIO[15] & GPIO[62] & GPIO[0]). See section 3.1 for more information.

37.3.6 I2C Slave Configuration Address Register

*Table 525.*0x14 - SLVCFG - I2C slave configuration address register

31 7	6 0
RESERVED	I2CCFGADDR
0	*
r	rw

31 : 7 RESERVED

6 : 0 I2C slave configuration address (I2CCFGADDR) - Address that slave responds to for configuration register accesses. The reset value is b101*xxx*1 where the three bits "*xxx*" are taken from the node ID bootstrap pins (GPIO[15] & GPIO[62] & GPIO[0]). See section 3.1 for more information.

38 I²C master

The LEON3FT microcontroller comprises two separate I²C master (I2CMST) units. Each I²C master unit controls its own external pins and has a unique AMBA address described in chapter 2.10.

The I²C master units are located on the APB bus in the address range from 0x8030E000 to 0x8030FFFF. See I²C master units connections in the next drawing. The figure shows memory locations and functions used for I²C master configuration and control.

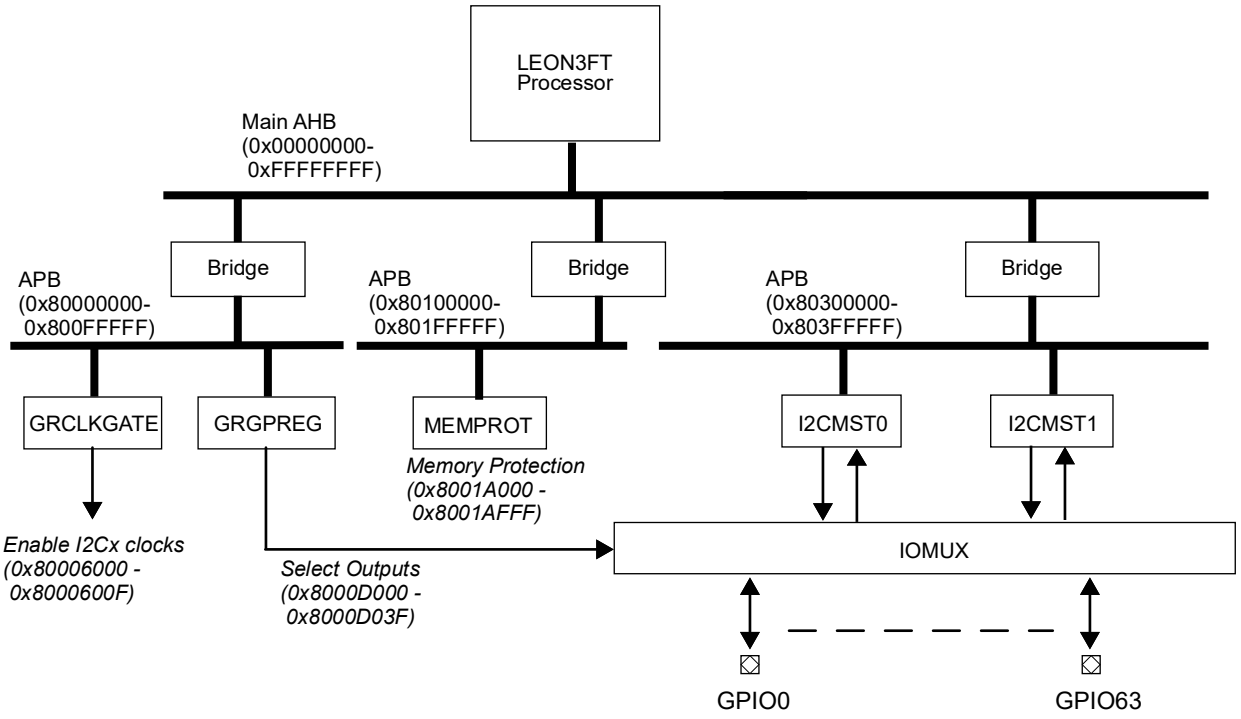


Figure 85. GR716B I²C-master bus and pin connection

The primary clock gating unit **GRCLKGATE** described in section 27 is used to enable/disable individual I²C master units. The unit **GRCLKGATE** can also be used to perform reset of individual I²C master units. Software must enable clock and release reset described in section 27 before I²C master configuration and transmission can start.

External IO selection per I²C master unit is made in the system IO configuration register (**GRGPREG**) in the address range from 0x8000D000 to 0x8000D03F. See section 7.1 for further information.

Each **I2CMSTx** unit controls its own external pins and has a unique AMBA address described in chapter 2.10. I2CMST unit 0 and 1 have identical configuration and status registers. Configuration and status registers are described in section 38.3.

The system can be configured to protect and restrict access to individual I²C-master unit in the **MEMPROT** unit. See section 47 for more information.

38.1 Overview

The I²C-master core is a modified version of the OpenCores I²C-Master with an AMBA APB interface. The core is compatible with Philips I²C standard and supports 7- and 10-bit addressing. Stan-

dard-mode (100 kb/s) and Fast-mode (400 kb/s) operation are supported directly. External pull-up resistors must be supplied for both bus lines.

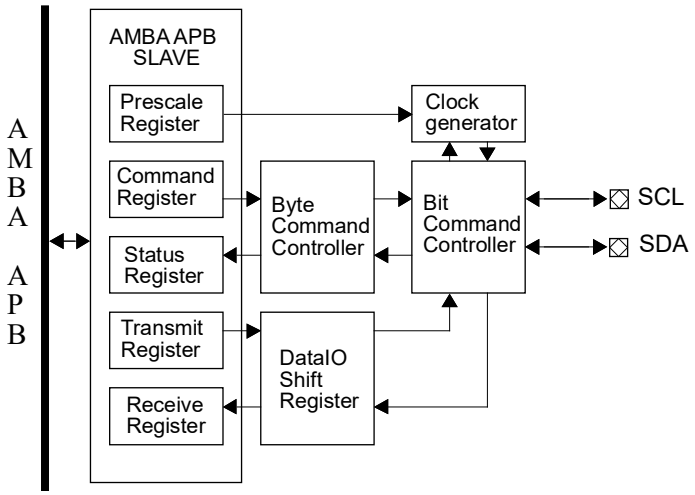


Figure 86. Block diagram

38.2 Operation

38.2.1 Transmission protocol

The I^2C-bus is a simple 2-wire serial multi-master bus with collision detection and arbitration. The bus consists of a serial data line (SDA) and a serial clock line (SCL). The I^2C standard defines three transmission speeds; Standard (100 kb/s), Fast (400 kb/s) and High speed (3.4 Mb/s).

A transfer on the I^2C-bus begins with a START condition. A START condition is defined as a high to low transition of the SDA line while SCL is high. Transfers end with a STOP condition, defined as a low to high transition of the SDA line while SCL is high. These conditions are always generated by a master. The bus is considered to be busy after the START condition and is free after a certain amount of time following a STOP condition. The bus free time required between a STOP and a START condition is defined in the I^2C-bus specification and is dependent on the bus bit rate.

Figure 87 shows a data transfer taking place over the I^2C-bus. The master first generates a START condition and then transmits the 7-bit slave address. The bit following the slave address is the R/$\overline{W}$ bit which determines the direction of the data transfer. In this case the R/$\overline{W}$ bit is zero indicating a write operation. After the master has transmitted the address and the R/$\overline{W}$ bit it releases the SDA line. The receiver pulls the SDA line low to acknowledge the transfer. If the receiver does not acknowledge the transfer, the master may generate a STOP condition to abort the transfer or start a new transfer by generating a repeated START condition.

After the first byte has been acknowledged the master transmits the data byte. If the R/$\overline{W}$ bit had been set to ‘1’ the master would have acted as a receiver during this phase of the transfer. After the data byte has been transferred the receiver acknowledges the byte and the master generates a STOP condition to complete the transfer. Section 38.2.3 contains three more example transfers from the perspective of a software driver.

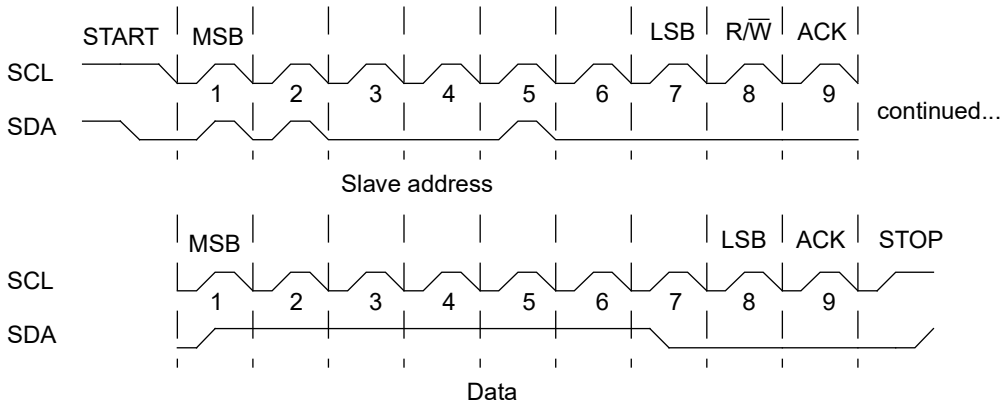


Figure 87. Complete I^{2}C data transfer

If the data bitrate is too high for a slave device, it may stretch the clock period by keeping SCL low after the master has driven SCL low.

38.2.2 Clock generation

The core uses the prescale register to determine the frequency of the SCL clock line and of the 5*SCL clock that the core uses internally. To calculate the prescale value use the formula:

$$Prescale = \frac{AMBAclockfrequency}{5 \cdot SCLfrequency} - 1$$

The *SCLfrequency* is 100 kHz for Standard-mode operation (100 kb/s) and 400 kHz for Fast mode operation. To use the core in Standard-mode in a system with a 60 MHz clock driving the AMBA bus the required prescale value is:

$$Prescale = \frac{60Mhz}{5 \cdot 100kHz} - 1 = 119 = 0x77$$

Note that the prescale register should only be changed when the core is disabled. The minimum recommended prescale value is 3 due to synchronization issues. This limits the minimum system frequency to 2 MHz for operation in Standard-mode (to be able to generate a 100 kHz SCL clock). However, a system frequency of 2 MHz will not allow the implementation fulfill the 100 ns minimum requirement for data setup time (required for Fast-mode operation). For compatibility with the I^{2}C Specification, in terms of minimum required data setup time, the minimum allowed system frequency is 20 MHz due to synchronization issues. If the core is run at lower system frequencies, care should be taken so that data from devices is stable on the bus one system clock period before the rising edge of SCL.

38.2.3 Software operational model

The core is initialized by writing an appropriate value to the clock prescale register and then setting the enable (EN) bit in the control register. Interrupts are enabled via the interrupt enable (IEN) bit in the control register.

To write a byte to a slave the I^{2}C-master must generate a START condition and send the slave address with the R/$\overline{W}$ bit set to ‘0’. After the slave has acknowledged the address, the master transmits the

data, waits for an acknowledge and generates a STOP condition. The sequence below instructs the core to perform a write:

1. Left-shift the I^{2}C-device address one position and write the result to the transmit register. The least significant bit of the transmit register (R/$\overline{W}$) is set to '0'.

2. Generate START condition and send contents of transmit register by setting the STA and WR bits in the command register.

3. Wait for interrupt, or for TIP bit in the status register to go low.

4. Read RxACK bit in status register. If RxACK is low the slave has acknowledged the transfer, proceed to step 5. If RxACK is set the device did not acknowledge the transfer, go to step 1.

5. Write the slave-data to the transmit register.

6. Send the data to the slave and generate a stop condition by setting STO and WR in the command register.

7. Wait for interrupt, or for TIP bit in the status register to go low.

8. Verify that the slave has acknowledged the data by reading the RxACK bit in the status register. RxACK should not be set.

To read a byte from an I^{2}C-connected memory much of the sequence above is repeated. The data written in this case is the memory location on the I^{2}C slave. After the address has been written the master generates a repeated START condition and reads the data from the slave. The sequence that software should perform to read from a memory device:

1. Left-shift the I^{2}C-device address one position and write the result to the transmit register. The least significant bit of the transmit register (R/W) is set to '0'.

2. Generate START condition and send contents of transmit register by setting the STA and WR bits in the command register.

3. Wait for interrupt or for TIP bit in the status register to go low.

4. Read RxACK bit in status register. If RxACK is low the slave has acknowledged the transfer, proceed to step 5. If RxACK is set the device did not acknowledge the transfer, go to step 1.

5. Write the memory location to be read from the slave to the transmit register.

6. Set the WR bit in the command register. Note that a STOP condition is not generated here.

7. Wait for interrupt, or for TIP bit in the status register to go low.

8. Read RxACK bit in the status register. RxACK should be low.

9. Address the I^{2}C-slave again by writing its left-shifted address into the transmit register. Set the least significant bit of the transmit register (R/W) to '1' to read from the slave.

10. Set the STA and WR bits in the command register to generate a repeated START condition.

11. Wait for interrupt, or for TIP bit in the status register to go low.

12. Read RxACK bit in the status register. The slave should acknowledge the transfer.

13. Prepare to receive the data read from the I^{2}C-connected memory. Set bits RD, ACK and STO on the command register. Setting the ACK bit NAKs the received data and signifies the end of the transfer.

14. Wait for interrupt, or for TIP in the status register to go low.

15. The received data can now be read from the receive register.

To perform sequential reads the master can iterate over steps 13 - 15 by not setting the ACK and STO bits in step 13. To end the sequential reads the ACK and STO bits are set. Consult the documentation of the I^{2}C-slave to see if sequential reads are supported.

The final sequence illustrates how to write one byte to an I2C-slave which requires addressing. First the slave is addressed and the memory location on the slave is transmitted. After the slave has acknowledged the memory location the data to be written is transmitted without a generating a new START condition:

1. Left-shift the I2C-device address one position and write the result to the transmit register. The least significant bit of the transmit register (R/W) is set to '0'.
2. Generate START condition and send contents of transmit register by setting the STA and WR bits in the command register.
3. Wait for interrupt or for TIP bit in the status register to go low.
4. Read RxACK bit in status register. If RxACK is low the slave has acknowledged the transfer, proceed to step 5. If RxACK is set the device did not acknowledge the transfer, go to step 1.
5. Write the memory location to be written from the slave to the transmit register.
6. Set the WR bit in the command register.
7. Wait for interrupt, or for TIP bit in the status register to go low.
8. Read RxACK bit in the status register. RxACK should be low.
9. Write the data byte to the transmit register.
10. Set WR and STO in the command register to send the data byte and then generate a STOP condition.
11. Wait for interrupt, or for TIP bit in the status register to go low.
12. Check RxACK bit in the status register. If the write succeeded the slave should acknowledge the data byte transfer.

The example sequences presented here can be generally applied to I2C-slaves. However, some devices may deviate from the protocol above, please consult the documentation of the I2C-slave in question. Note that a software driver should also monitor the arbitration lost (AL) bit in the status register.

38.3 Registers

The core is programmed through registers mapped into APB address space.

Table 526. I2C-master registers

APB address offset	Register
0x00	Clock prescale register
0x04	Control register
0x08	Transmit register*
0x08	Receive register**
0x0C	Command register*
0x0C	Status register**
0x10	Dynamic filter register

* Write only

** Read only

38.3.1 I²C-Master Clock Prescale Register

Table 527. 0x00 - PRESCALE - I²C-master Clock prescale register

31 16	15 0
RESERVED	Clock prescale
0	0xFFFF
r	rw

31 : 16 RESERVED

15:0 Clock prescale - Value is used to prescale the SCL clock line. Do not change the value of this register unless the EN field of the control register is set to '0'. The minimum recommended value of this register is 0x0003. Lower values may cause the master to violate I²C timing requirements due to synchronization issues.

38.3.2 I²C-Master Control Register

Table 528. 0x04 - CTRL - I²C-master control register

31 8	7	6	5 0
RESERVED	EN	IEN	RESERVED
0	0	0	0
r	rw	rw	r

31 : 8 RESERVED

7 Enable (EN) - Enable I²C core. The core is enabled when this bit is set to '1'.

6 Interrupt enable (IEN) - When this bit is set to '1' the core will generate interrupts upon transfer completion.

5:0 RESERVED

38.3.3 I²C-Master Transmit Register

Table 529. 0x08 - TX - I²C-master transmit register

31 8	7 1	0
RESERVED	TDATA	RW
0	0	0
-	w	w

31 : 8 RESERVED

7:1 Transmit data (TDATA) - Most significant bits of next byte to transmit via I²C

0 Read/Write (RW) - In a data transfer this is the data's least significant bit. In a slave address transfer this is the RW bit. '1' reads from the slave and '0' writes to the slave.

38.3.4 I²C-Master Receive Register

Table 530. 0x08 - RX - I²C-master receive register

31 8	7 0
RESERVED	RDATA
	0
	r

31 : 8 RESERVED

7:0 Receive data (RDATA) - Last byte received over I²C-bus.

38.3.5 I²C-Master Command Register

*Table 531.*0x0C -CMD - I²C-master command register

31 … 8	7	6	5	4	3	2 … 1	0
RESERVED	STA	STO	RD	WR	ACK	RESERVED	IACK
0	0	0	0	0	0	0	0
r	w*	w*	w*	w*	w*	r	-

31 : 8 RESERVED

7 Start (STA) - Generate START condition on I²C-bus. This bit is also used to generate repeated START conditions.

6 Stop (STO) - Generate STOP condition

5 Read (RD) - Read from slave

4 Write (WR) - Write to slave

3 Acknowledge (ACK) - Used when acting as a receiver. ‘0’ sends an ACK, ‘1’ sends a NACK.

2:1 RESERVED

0 Interrupt acknowledge (IACK) - Clears interrupt flag (IF) in status register.

38.3.6 I²C-Master Status Register

*Table 532.*0x0C - STAT - I²C-master status register

31 … 8	7	6	5	4 … 2	1	0
RESERVED	RxACK	BUSY	AL	RESERVED	TIP	IF
0	0	0	0	0	0	0
r	r	r	r	r	r	wc

31 : 8 RESERVED

7 Receive acknowledge (RxACK) - Received acknowledge from slave. ‘1’ when no acknowledge is received, ‘0’ when slave has acked the transfer.

6 I²C-bus busy (BUSY) - This bit is set to ‘1’ when a start signal is detected and reset to ‘0’ when a stop signal is detected.

5 Arbitration lost (AL) - Set to ‘1’ when the core has lost arbitration. This happens when a stop signal is detected but not requested or when the master drives SDA high but SDA is low.

4:2 RESERVED

1 Transfer in progress (TIP) - ‘1’ when transferring data and ‘0’ when the transfer is complete. This bit is also set when the core will generate a STOP condition.

0 Interrupt flag (IF) - This bit is set when a byte transfer has been completed and when arbitration is lost. If IEN in the control register is set an interrupt will be generated. New interrupts will ge generated even if this bit has not been cleared.

38.3.7 I²C-Master Dynamic Filter Register

*Table 533.*0x10 - FILT - I²C-master dynamic filter register

31 … 2	1 … 0
RESERVED	FILT
0	0x3
r	rw

31 : 2 RESERVED

1 : 0 Dynamic filter reload value (FILT) - This field sets the reload value for the dynamic filter counter. The core will ignore all pulses on the bus shorter than 2 * (system clock period) and may also ignore pulses shorter than 2 * 2 * (system clock period) - 1.

39 I²C slave

The LEON3FT microcontroller comprises two separate I²C slave (I2CSLV) units. Each I²C slave unit controls its own external pins and has a unique AMBA address described in chapter 2.10.

The I²C slave units are located on the APB bus in the address range from 0x8040C000 to 0x8040DFFF. See I²C slave units connections in the next drawing. The figure shows memory locations and functions used for I²C slave configuration and control.

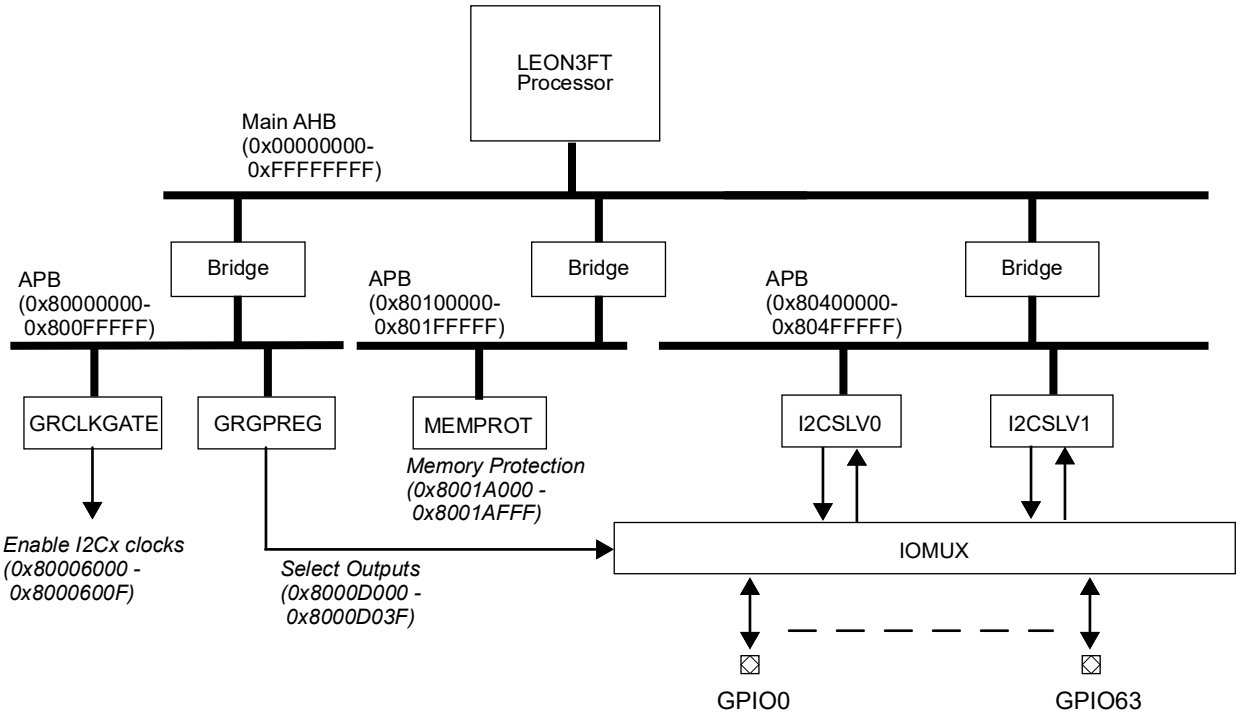


Figure 88. GR716B I²C-slave bus and pin connection

The primary clock gating unit **GRCLKGATE** described in section 27 is used to enable/disable individual I²C slave units. The unit **GRCLKGATE** can also be used to perform reset of individual I²C slave units. Software must enable clock and release reset described in section 27 before I²C slave configuration and transmission can start.

External IO selection per I²C slave unit is made in the system IO configuration register (**GRGPREG**) in the address range from 0x8000D000 to 0x8000D03F. See section 7.1 for further information.

Each **I2CSLVx** unit controls its own external pins and has a unique AMBA address described in chapter 2.10. I2CSLV unit 0 and 1 has identical configuration and status registers. Configuration and status registers are described in section 39.3.

System can be configured to protect and restrict access to individual I²C slave unit in the **MEMPROT** unit. See section 47 for more information.

39.1 Overview

The I²C slave core is a simple I²C slave that provides a link between the I²C bus and the AMBA APB. The core is compatible with Philips I²C standard and supports 7- and 10-bit addressing with an optionally software programmable address. Standard-mode (100 kb/s) and Fast-mode (400 kb/s) operation are supported directly. External pull-up resistors must be supplied for both bus lines.

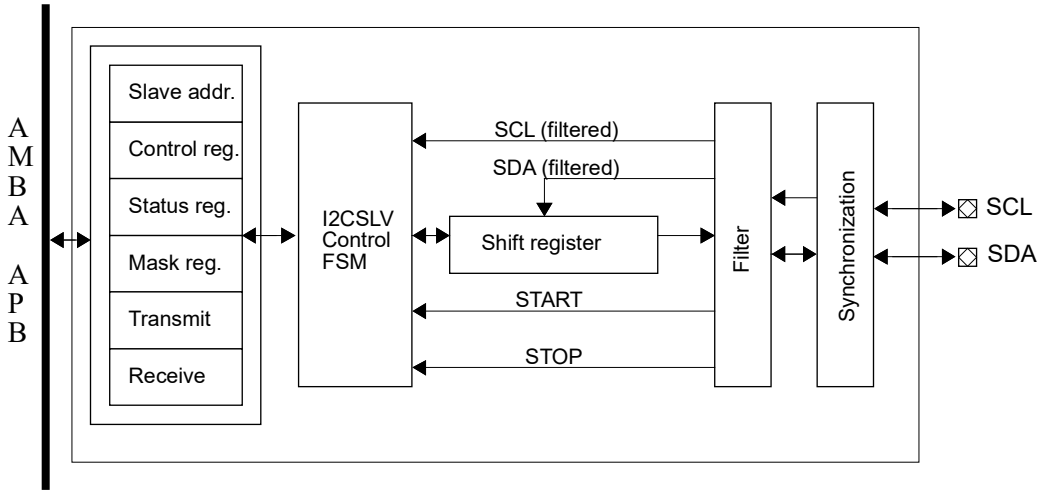


Figure 89. Block diagram

39.2 Operation

39.2.1 Transmission protocol

The I^2C-bus is a simple 2-wire serial multi-master bus with collision detection and arbitration. The bus consists of a serial data line (SDA) and a serial clock line (SCL). The I^2C standard defines three transmission speeds; Standard (100 kb/s), Fast (400 kb/s) and High speed (3.4 Mb/s).

A transfer on the I^2C-bus begins with a START condition. A START condition is defined as a high to low transition of the SDA line while SCL is high. Transfers end with a STOP condition, defined as a low to high transition of the SDA line while SCL is high. These conditions are always generated by a master. The bus is considered to be busy after the START condition and is free after a certain amount of time following a STOP condition. The bus free time required between a STOP and a START condition is defined in the I^2C-bus specification and is dependent on the bus bit rate.

Figure 90 shows a data transfer taking place over the I^2C-bus. The master first generates a START condition and then transmits the 7-bit slave address. I^2C also supports 10-bit addresses, which are discussed briefly below. The bit following the slave address is the R/$\overline{W}$ bit which determines the direction of the data transfer. In this case the R/$\overline{W}$ bit is zero indicating a write operation. After the master has transmitted the address and the R/$\overline{W}$ bit it releases the SDA line. The receiver pulls the SDA line low to acknowledge the transfer. If the receiver does not acknowledge the transfer, the master may generate a STOP condition to abort the transfer or start a new transfer by generating a repeated START condition.

After the address has been acknowledged the master transmits the data byte. If the R/$\overline{W}$ bit had been set to ‘1’ the master would have acted as a receiver during this phase of the transfer. After the data byte has been transferred the receiver acknowledges the byte and the master generates a STOP condition to complete the transfer.

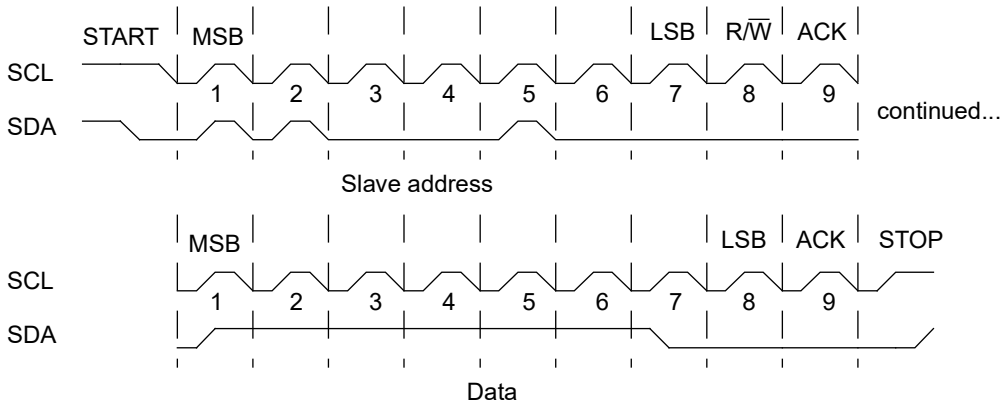


Figure 90. Complete I^{2}C data transfer

An I^{2}C slave may also support 10-bit addressing. In this case the master first transmits a pattern of five reserved bits followed by the two first bits of the 10-bit address and the R/$\overline{W}$ bit set to '0'. The next byte contains the remaining bits of the 10-bit address. If the transfer is a write operation the master then transmits data to the slave. To perform a read operation the master generates a repeated START condition and repeats the first part of the 10-bit address phase with the R/$\overline{W}$ bit set to '1'.

If the data bitrate is too high for a slave device or if the slave needs time to process data, it may stretch the clock period by keeping SCL low after the master has driven SCL low.

39.2.2 Slave addressing

The core have a programmable address and support for 7-bit and 10-bit addresses. The core is configured to use 10-bit address as default. The address mode controlled with the TBA bit in the Slave address register.

39.2.3 System clock requirements and sampling

The core samples the incoming I^{2}C SCL clock and does not introduce any additional clock domains into the system. Both the SCL and SDA lines first pass through two stage synchronizers and are then filtered with a low pass filter consisting of four registers.

START and STOP conditions are detected if the SDA line, while SCL is high, is at one value for two system clock cycles, toggles and keeps the new level for two system clock cycles.

The synchronizers and filters constrain the minimum system frequency. The core requires the SCL signal to be stable for at least four system clock cycles before the core accepts the SCL value as the new clock value. The core's reaction to transitions will be additionally delayed since both lines are taken through two-stage synchronizers before they are filtered. Therefore it takes the core over eight system clock cycles to discover a transition on SCL. To use the slave in Standard-mode operation at 100 kHz the recommended minimum system frequency is 2 MHz. For Fast-mode operation at 400 kHz the recommended minimum system frequency is 6 MHz.

39.2.4 Operational model

The core has four main modes of operation and is configured to use one of these modes via the Control register bits Receive Mode (RMOD) and Transmit Mode (TMOD). The mode setting controls the core's behavior after a byte has been received or transmitted.

The core will always NAK a received byte if the receive register is full when the whole byte is received. If the receive register is free the value of RMOD determines if the core should continue to listen to the bus for the master's next action or if the core should drive SCL low to force the master

into a wait state. If the value of the RMOD field is ‘0’ the core will listen for the master’s next action. If the value of the RMOD field is ‘1’ the core will drive SCL low until the Receive register has been read and the Status register bit Byte Received (REC) has been cleared. Note that the core has not accepted a byte if it does not acknowledge the byte.

When the core receives a read request it evaluates the Transmit Valid (TV) bit in the Control register. If the Transmit Valid bit is set the core will acknowledge the address and proceed to transmit the data held in the Transmit register. After a byte has been transmitted the core assigns the value of the Control register bit Transmit Always Valid (TAV) to the Transmit Valid (TV) bit. This mechanism allows the same byte to be sent on all read requests without software intervention. The value of the Transmit Mode (TMOD) bit determines how the core acts after a byte has been transmitted and the master has acknowledged the byte, if the master NAKs the transmitted byte the transfer has ended and the core goes into an idle state. If TMOD is set to ‘0’ when the master acknowledges a byte the core will continue to listen to the bus and wait for the master’s next action. If the master continues with a sequential read operation the core will respond to all subsequent requests with the byte located in the Transmit Register. If TMOD is ‘1’ the core will drive SCL low after a master has acknowledged the transmitted byte. SCL will be driven low until the Transmit Valid bit in the control register is set to ‘1’. Note that if the Transmit Always Valid (TAV) bit is set to ‘1’ the Transmit Valid bit will immediately be set and the core will have show the same behavior for both Transmit modes.

When operating in Receive or Transmit Mode ‘1’, the bus will be blocked by the core until software has acknowledged the transmitted or received byte. This may have a negative impact on bus performance and it also affects single byte transfers since the master is prevented to generate STOP or repeated START conditions when SCL is driven low by the core.

The core reports three types of events via the Status register. When the core NAKs a received byte, or its address in a read transfer, the NAK bit in the Status register will be set. When a byte is successfully received the core asserts the Byte Received (REC) bit. After transmission of a byte, the Byte Transmitted (TRA) bit is asserted. These three bits can be used as interrupt sources by setting the corresponding bits in the Mask register.

39.3 Registers

The core is programmed through registers mapped into APB address space.

Table 534. I^2C slave registers

APB address offset	Register
0x00	Slave address register
0x04	Control register
0x08	Status register
0x0C	Mask register
0x10	Receive register
0x14	Transmit register

39.3.1 Slave Address Register

*Table 535.*0x00 - SLVADDR - Slave address register

31	30 10	9 0
TBA	RESERVED	SLVADDR
1	0	0x50
rw	r	rw

31 Ten-bit Address (TBA) - When this bit is set the core will interpret the value in the SLVADDR field as a 10-bit address.

30 : 10 RESERVED

9:0 Slave address (SLVADDR) - Contains the slave I2C address.

39.3.2 Control Register

*Table 536.*0x04 - CTRL - Control register

31 5	4	3	2	1	0
RESERVED	RMOD	TMOD	TV	TAV	EN
0	NR	NR	NR	NR	NR
r	rw	rw	rw	rw	rw

31 : 5 RESERVED

4 Receive Mode (RMOD) - Selects how the core handles writes:

‘0’: The slave accepts one byte and NAKs all other transfers until software has acknowledged the received byte by reading the Receive register.

‘1’: The slave accepts one byte and keeps SCL low until software has acknowledged the received byte by reading the Receive register.

3 Transmit Mode (TMOD) - Selects how the core handles reads:

‘0’: The slave transmits the same byte to all if the master requests more than one byte in the transfer. The slave then NAKs all read requests as long as the Transmit Valid (TV) bit is unset.

‘1’: The slave transmits one byte and then keeps SCL low until software has acknowledged that the byte has been transmitted by setting the Transmit Valid (TV) bit.

2 Transmit Valid (TV) - Software sets this bit to indicate that the data in the transmit register is valid. The core automatically resets this bit when the byte has been transmitted. When this bit is ‘0’ the core will either NAK or insert wait states on incoming read requests, depending on the Transmit Mode (TMOD).

1 Transmit Always Valid (TAV) - When this bit is set, the core will not clear the Transmit Valid (TV) bit when a byte has been transmitted.

0 Enable core (EN) - Enables core. When this bit is set to ‘1’ the core will react to requests to the address set in the Slave address register. If this bit is ‘0’ the core will keep both SCL and SDA inputs in Hi-Z state.

39.3.3 Status Register

*Table 537.*0x08 - STAT - Status register

31	3	2	1	0
RESERVED		REC	TRA	NAK
0		0	0	0
r		*	wc	wc

31 : 3 RESERVED

2 Byte Received (REC) - This bit is set to '1' when the core accepts a byte and is automatically cleared when the Receive register has been read.

1 Byte Transmitted (TRA) - This bit is set to '1' when the core has transmitted a byte and is cleared by writing '1' to this position. Writes of '0' have no effect.

0 NAK Response (NAK) - This bit is set to '1' when the core has responded with NAK to a read or write request. This bit does not get set to '1' when the core responds with a NAK to an address that does not match the cores address. This bit is cleared by writing '1' to this position, writes of '0' have no effect.

39.3.4 Mask Register

*Table 538.*0x0C - MASK - Mask register

31	3	2	1	0
RESERVED		RECE	TRAE	NAKE
0		0	0	0
r		rw	rw	rw

31 : 3 RESERVED

2 Byte Received Enable (RECE) - When this bit is set the core will generate an interrupt when bit 2 in the Status register gets set.

1 Byte Transmitted Enable (TRAE) - When this bit is set the core will generate an interrupt when bit 1 in the Status register gets set.

0 NAK Response Enable (NAKE) - When this bit is set the core will generate an interrupt when bit 0 in the Status register gets set.

39.3.5 Receive Register

*Table 539.*0x10 - RX - Receive register

31	8	7	0
RESERVED		RECBYTE	
0		NR	
r		r	

31 : 8 RESERVED

7:0 Received Byte (RECBYTE) - Last byte received from master. This field only contains valid data if the Byte received (REC) bit in the status register has been set.

39.3.6 Transmit Register

*Table 540.*0x14 - TX - Transmit register

31 8	8 7 0
RESERVED	TRABYTE
0	NR
r	rw

31 : 8 RESERVED

7:0 Transmit Byte (TRABYTE) - Byte to transmit on the next master read request.

Reset value: Undefined

40 Interrupt Controller

The LEON3FT microcontroller have one Interrupt controller (IRQAMP). The Interrupt controller (IRQAMP) have a unique AMBA base address described in chapter 2.10.

The Interrupt controller (IRQAMP) is located on APB bus in the address range from 0x80002000 to 0x80002FFF. See the Interrupt controller (IRQAMP) control and status interface connection in next drawing. The drawing picture memory locations and functions used for Interrupt controller (IRQAMP) configuration and control.

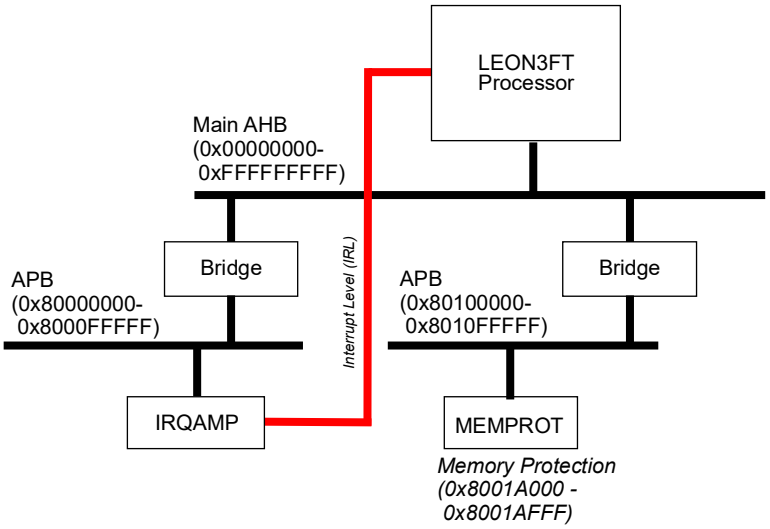


Figure 91. GR716B Interrupt controller bus connection

It is not possible to disable clock to the interrupt controller since the interrupt controller is used to wake-up the processor from deep-sleep i.e. when the clock to the processor is disabled.

The interrupt controllers configuration and status registers are describe in this section 40.3.

System can be configured to protect and restrict access to interrupt controller in the **MEMPROT** unit. For more information See section 47 for more information.

40.1 Overview

The LEON3FT microcontroller implements an interrupt scheme where interrupt lines are routed together with the remaining AHB/APB bus signals forming an interrupt bus. The interrupt controller core is attached to the AMBA bus as an APB slave and monitors the combined interrupt signals.

The interrupts generated on the interrupt bus are all forwarded to the interrupt controller. The interrupt controller prioritizes, masks and propagates the interrupt with the highest priority to the processor.

Interrupts from peripherals has been assigned a unique ID see chapter 2.12. The unique peripheral interrupt ID can be used for dynamically remapping of interrupts in the interrupt controller.

40.1.1 Definition

This chapter defines and explains the interrupt terminology used in this chapter. In the following chapters the following terms will be used:

- Bus interrupt line
- Interrupt ID number
- Extended Interrupt number

The ***bus interrupt line*** is the actual hardware interrupt used by the peripheral. The term ***bus*** is used since the internal hardware interrupt lines are distributed via the system bus architecture. The *bus interrupt line* is in the range from 0 to 63

The ***interrupt number*** refers to the interrupt line handled by the interrupt controller i.e. values between 1 to 15. Any arbitrary *bus interrupt line* can be mapped to any arbitrary *interrupt number* from 2 to 15. *Interrupt number 1* has the lowest priority and is reserved for *Extended interrupt numbers*.

Extended interrupt number is arbitrary bus interrupt lines mapped to arbitrary numbers between 16 to 32. *Extended interrupt numbers* is grouped into one *interrupt number* i.e. all *extended interrupt numbers* have the same priority and *interrupt number*.

40.1.2 Structure

This is a picture of the system interrupt generation and remapping functionality available in the GR716B device.

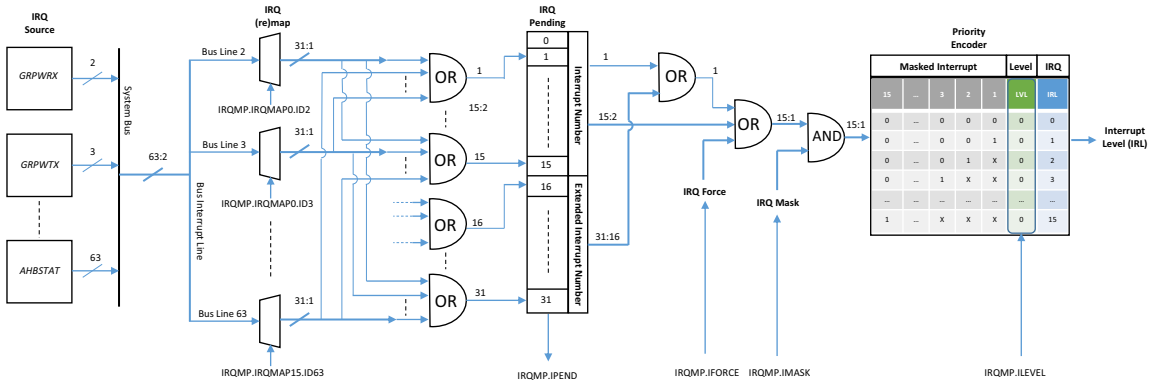


Figure 92. System and Interrupt controller block diagram

40.2 Operation

40.2.1 Interrupt prioritization

The interrupt controller monitors *interrupt number* 1 - 15 and *extended interrupt number* 16 - 32. When any of these interrupts are asserted high, the corresponding bit in the interrupt pending register is set. The pending bits will stay set until cleared by software or by an interrupt acknowledge from the processor.

Interrupt number 1 - 15 can be assigned to one of two levels (0 or 1) as programmed in the interrupt level register. Level 1 has higher priority than level 0. The interrupts are prioritised within each level, with interrupt 15 having the highest priority and interrupt 1 the lowest. The highest interrupt from level 1 will be forwarded to the processor. If no unmasked pending interrupt exists on level 1, then the highest unmasked interrupt from level 0 will be forwarded.

Extended interrupt number 16 - 32 are grouped and OR:ed into *Interrupt number 1* depict in figure 92. *Extended interrupt number* 16 - 32 has no level control an have no prioritization between individual interrupts.

When the LEON3FT processor acknowledges the interrupt, the corresponding pending bit will automatically be cleared. For *extended interrupt* the extended acknowledge register will identify which extended interrupt that was most recently acknowledged. This register can be used by software to invoke the appropriate interrupt handler for the extended interrupts.

Interrupt can also be forced by setting a bit in the interrupt force register. In this case, the processor acknowledgment will clear the force bit rather than the pending bit. After reset, the interrupt mask register is set to all zeros while the remaining control registers are undefined. Note that interrupt 15 cannot be maskable by the LEON3FT processor and should be used with care - most operating systems do not safely handle this interrupt.

40.2.2 Interrupt (re)map functionality

The LEON3FT microcontroller have 64 unique *bus interrupt line* sources listed in section 2.12, while the LEON3FT processor only supports 31 unique interrupt sources i.e. *interrupt ID number* 1 - 15 and *extended interrupt number* 16 - 32.

To accommodate all the 64 unique *bus interrupt line* sources the interrupt controller allow dynamic remapping between *bus interrupt lines* and any *interrupt ID number* 1 - 15 or any *extended interrupt number* 16 - 32. Individual remap logic on each incoming *bus interrupt line* will map the *bus interrupt line* sources to specified *interrupt ID number* 1 - 15 or *extended interrupt number* 16 - 32.

The Interrupt map registers is available starting at address 0x80002300 from the interrupt controller's base address. The interrupt map registers contain one field for each *bus interrupt line* in the system. The value within this field determines to which interrupt controller line the *bus interrupt line* is connected. In case several *bus interrupt lines* are mapped to the same *interrupt ID number* or *extended interrupt number* (several fields in the Interrupt map registers have the same value) then the *bus interrupt lines* will be OR:ed together.

Note that if *bus interrupt line X* is remapped to controller *interrupt ID number 2 - 15* then corresponding bit in the range 2 - 15 of the pending register will be set when a peripheral asserts interrupt *bus interrupt line X*. Where, the *bus interrupt line X* is remapped to controller *extended interrupt number* 16 - 32 then corresponding bit in the range 16 - 32 and bit 1 of the pending register will be set when a peripheral asserts interrupt *bus interrupt line X*

40.2.3 Processor status monitoring

The processor status can be monitored through the Processor Status Register. The STATUS field in this register indicates if a processor is halted (‘1’) or running (‘0’). A halted processor can be reset and restarted by writing a ‘1’ to its status field.

The interrupt controller also supports setting the reset start address dynamically. Please see section 40.2.7 for further information.

40.2.4 Interrupt timestamping description

Interrupt timestamping is controlled via the Interrupt Timestamp Control register(s). Each Interrupt Timestamp Control register contains a field (TSTAMP) that contains the number of timestamp registers sets that the core implements. A timestamp register sets consist of one Interrupt Timestamp Counter register, one Interrupt Timestamp Control register, one Interrupt Assertion Timestamp register and one Interrupt Acknowledge Timestamp register.

Software enables timestamping for a specific interrupt via a Interrupt Timestamp Control Register. When the selected interrupt line is asserted, software will save the current value of the interrupt timestamp counter into the Interrupt Assertion Timestamp register and set the S1 field in the Interrupt Timestamp Control Register. When the processor acknowledges the interrupt, the S2 field of the Interrupt Timestamp Control register will be set and the current value of the timestamp counter will be saved in the Interrupt Acknowledge Timestamp Register. The difference between the Interrupt Assertion timestamp and the Interrupt Acknowledge timestamp is the number of system clock cycles that was required for the processor to react to the interrupt and divert execution to the trap handler.

The core can be configured to stamp only the first occurrence of an interrupt or to continuously stamp interrupts. The behavior is controlled via the Keep Stamp (KS) field in the Interrupt Timestamp Con-

trol Register. If KS is set, only the first assertion and acknowledge of an interrupt is stamped. Software must then clear the S1 and S2 fields for a new timestamp to be taken. If Keep Stamp is disabled (KS field not set), the controller will update the Interrupt Assertion Timestamp Register every time the selected interrupt line is asserted. In this case the controller will also automatically clear the S2 field and also update the Interrupt Acknowledge Timestamp register with the current value when the interrupt is acknowledged.

40.2.5 Interrupt timestamping usage guidelines

Note that KS = '0' and a high interrupt rate may cause the Interrupt Assertion Timestamp register to be updated (and the S2 field reset) before the processor has acknowledged the first occurrence of the interrupt. When the processor then acknowledges the first occurrence, the Interrupt Acknowledge Timestamp register will be updated and the difference between the two Timestamp registers will not show how long it took the processor to react to the first interrupt request. If the interrupt frequency is expected to be high it is recommended to keep the first stamp (KS field set to '1') in order to get reliable measurements. KS = '0' should not be used in systems that include cores that use level interrupts, the timestamp logic will register each cycle that the interrupt line is asserted as an interrupt.

In order to measure the full interrupt handling latency in a system, software should also read the current value of the Interrupt Timestamp Counter when entering the interrupt handler. In the typical case, a software driver's interrupt handler reads a status register and then determines the action to take. Adding a read of the timestamp counter before this status register read can give an accurate view of the latency during interrupt handling.

The interrupt controller listens to the system interrupt vector when reacting to interrupt line assertions. This means that the Interrupt Assertion Timestamp Register(s) will not be updated if software writes directly to the pending or force registers. To measure the time required to serve a forced interrupt, read the value of the Interrupt Timestamp counter before forcing the interrupt and then read the Interrupt Acknowledge Timestamp and Interrupt Timestamp counter when the processor has reacted to the interrupt.

40.2.6 Watchdog

The interrupt controller supports for asserting a bit in the controller's Interrupt Pending Register when an external watchdog signal is asserted. This functionality can be used to implement a sort of soft watchdog for one or several processor cores. The controller's Watchdog Control Register contains a field that shows the number of external watchdog inputs supported and fields for configuring which watchdog inputs that should be able to assert a bit in the Interrupt Pending Register. The pending register will be assigned in each cycle that a selected watchdog input is high. Therefore it is recommended that the watchdog inputs are connected to sources which send a one clock cycle long pulse when a watchdog expires. Otherwise software should make sure that the watchdog signal is deasserted before re-enabling interrupts during interrupt handling.

The GR716B microcontroller supports soft watchdog events from GPTIMER0 timer 6 and GPTIMER0 timer 7.

40.2.7 Dynamic processor reset start address

The interrupt controller can be used to start processor execution from a specified start address. The interface provided to accomplish this is:

- Error mode status register
- Processor boot address registers

The register interface allows software to force a processor into debug or error mode. This means that the interface can be used to stop (and restart) a processor. Registers are available to allow starting a halted processor from an arbitrary 8 byte aligned entry point. The processor can be started with the

same register write as when the entry point is written, or the processor can be started later using the regular processor status register bit.

An error register is also added to allow monitoring processors for error mode, and to allow forcing a specific processor into error mode. This can be used to monitor and re-boot processors without reseting the system.

40.2.8 Restart processor from internal on-chip memory

To restart the processor from on-chip instruction memory set the register PROCBOOTADR=0x31000001.

40.2.9 Restart processor from external SRAM memory

To restart the processor from on-chip instruction memory set the register PROCBOOTADR=0x40000001.

40.3 Registers

The core is controlled through registers mapped into APB address space.

*Table 541.*Interrupt Controller registers

APB address offset	Register
0x80002000	Interrupt level register
0x80002004	Interrupt pending register
0x80002008	Interrupt force register
0x8000200C	Interrupt clear register
0x80002010	Status register
0x80002014	Reserved
0x80002018	Error Mode status register
0x8000201C	Watchdog control register
0x80002020	Reserved
0x80002024	Reserved
0x80002028	Reserved
0x8000202C	Reserved
0x80002030	Reserved
0x80002034	Extended Interrupt Clear Register
0x80002038	Reserved
0x8000203C	Reserved
0x80002040	Processor interrupt mask register
0x80002080	Processor interrupt force register
0x800020C0	Processor extended interrupt acknowledge register
0x80002100	Interrupt timestamp 0 counter register
0x80002104	Interrupt timestamp 0 control register
0x80002108	Interrupt assertion timestamp 0 register
0x8000210C	Interrupt acknowledge timestamp 0 register
0x80002110	Interrupt timestamp 1 counter register (mirrored in each set)
0x80002114	Interrupt timestamp *1* control register
0x80002118	Interrupt assertion timestamp *1* register
0x8000211C	Interrupt acknowledge timestamp *1* register
0x80002120	Interrupt timestamp 2 counter register (mirrored in each set)
0x80002124	Interrupt timestamp *2* control register
0x80002128	Interrupt assertion timestamp *2* register
0x8000212C	Interrupt acknowledge timestamp *2* register
0x80002130	Interrupt timestamp 3 counter register (mirrored in each set)
0x80002134	Interrupt timestamp *3* control register
0x80002138	Interrupt assertion timestamp *3* register
0x8000213C	Interrupt acknowledge timestamp *3* register
0x80002200	Processor boot address register
0x80002300 + 0x4 * *m*	Interrupt map register *m*

* Number of interrupts in LEON3FT microcontroller is 64 hence *m* is 16

40.3.1 Interrupt Level Register

*Table 542.*0x80002000 - ILEVEL - Interrupt Level Register

31 16	15 1	0
RESERVED	IL[15:1]	R
0	NR	0
r	rw	r

31:16 Reserved

15:1 Interrupt Level n (IL[n]) - Interrupt level for interrupt n

0 Reserved

40.3.2 Interrupt Pending Register

*Table 543.*0x80002004 - IPEND - Interrupt Pending Register

31 16	15 1	0
EIP[31:16]	IP[15:1]	R
0	0	0
rw	rw	r

31:16 Extended Interrupt Pending n (EIP[n])

15:1 Interrupt Pending n (IP[n]) - Interrupt pending for interrupt n

0 Reserved

40.3.3 Interrupt Force Register

*Table 544.*0x80002008 - IFORCE0 - Interrupt Force Register

31 16	15 1	0
RESERVED	IF[15:1]	R
0	0	0
r	rw	r

31:16 Reserved

15:1 Interrupt Force n (IF[n]) - Force interrupt nr n.

0 Reserved

40.3.4 Interrupt Clear Register

Table 545. 0x8000200C - ICLEAR - Interrupt Clear Register

31 16	15 1	0
EIC[31:16]	IC[15:1]	R
0	0	0
w	w	r

31:16 Extended Interrupt Clear n (EIC[n])

15:1 Interrupt Clear n (IC[n]) - Writing ‘1’ to IC[n] will clear interrupt n

0 Reserved

40.3.5 Status Register

Table 546. 0x80002010 - MPSTAT - Status Register

31 28	27	26	25 20	19 16	15 1	0
RESERVED	ba	er	RESERVED	EIRQ		STATUS
0	1	1	0	1	0	*
r	r	r	r	r	r	rw

31:28 Reserved

27: Broadcast Available (BA) - Default to '1' since NCPU > 0.

26: Default to '1' since extended boot registers available (ER).

25:20 Reserved

19:16 Extended IRQ (EIRQ) - Interrupt number 1 used for extended interrupts.

15:1 Reserved

0 Power-down status of CPU (STATUS) - 0x1 = power-down, 0x0 = running. Write STATUS with 0x1 to start processor.

40.3.6 Error Mode Status Register

Table 547. 0x80002018 - ERRSTAT - Error Mode Status Register

31 1	0
RESERVED	EM
0	0
r	rw

31:1 Reserved

0 Error Mode register (EM) - Read operation of register shows the error mode of the LEON3FT processor(1 = 'error mode', '0'=debug/run/power-down). Write to register will force LEON3FT processor into error mode.

40.3.7 Watchdog Control Register

Table 548. 0x8000201C - WDOGCTRL - Watchdog Control Register

31 27	26 20	19 16	15 0
NWDOG	Reserved	WDOGIRQ	WDOGMSK
2	0	NR	0
r	r	rw	rw

31:27 Number of watchdog inputs (NWDOG) - Number of watchdog inputs that the core supports.

26:20 Reserved

19:16 Watchdog interrupt (WDOGIRQ) - Selects the bit in the pending register to set when any line watchdog line selected by the WDOGMSK field is asserted.

15:0 Watchdog Mask n (WDOGMSK[n]) - If WDOGMSK[n] = '1' then the assertion of watchdog input n will lead to the bit selected by the WDOGIRQ field being set in the controller's Interrupt Pending Register.

Configurable soft watchdog inputs:

Bit #0 - Enable soft watchdog for GPTIMER0 timer 7

Bit #1 - Enable soft watchdog for GPTIMER0 timer 6

Bit #2 to Bit #15 are unused

40.3.8 Processor Interrupt Mask Register

Table 549. 0x80002040 - PIMASK - Processor Interrupt Mask Register

31 16	15 1	0
EIM[31:16]	IM15:1]	R
0	0	0
rw	rw	r

31:16 Extended Interrupt Mask n (EIC[n]) - Interrupt mask for extended interrupts

15:1 Interrupt Mask n (IM[n]) - If IM[n] = ‘0’ then interrupt n is masked, otherwise it is enabled.

0 Reserved

40.3.9 Processor Interrupt Force Register

Table 550. 0x80002080 - PCFORCE - Processor Interrupt Force Register

31 17	16	15 1	0
IFC[15:1]	R	IF15:1]	R
0	0	0	0
wc	r	rw*	r

31:17 Interrupt Force Clear n (IFC[n]) - Interrupt force clear for interrupt n

16 Reserved

15:1 Interrupt Force n (IF[n]) - Force interrupt nr n

0 Reserved

40.3.10 Extended Interrupt Acknowledge Register

Table 551. 0x800020C0 - PEXTACK - Extended Interrupt Acknowledge Register

31 6	5 0
RESERVED	EID[5:0]
0	0
r	r

31:6 Reserved

5:0 Extended interrupt ID (EID) - ID (16-63) of the most recent acknowledged extended interrupt

If this field is 0, and support for extended interrupts exist, the last assertion of interrupt *eirq* was not the result of an extended interrupt being asserted. If interrupt *eirq* is forced, or asserted, this field will be cleared unless one, or more, of the interrupts 63 - 16 are enabled and set in the pending register.

40.3.11 Interrupt Timestamp Counter Register

Table 552. 0x80002100 - TCNT0 - Interrupt Timestamp 0 Counter register

31 0
TCNT
0
r

31:0 Timestamp Counter (TCNT) - Current value of timestamp counter. The counter increments whenever a TSISEL field in a Timestamp Control Register is non-zero. The counter will wrap to zero upon overflow and is read only.

Table 553. 0x80002110 - TCNT1 - Interrupt Timestamp 1 Counter register

31	0
TCNT	
0	
r	

31:0 Timestamp Counter (TCNT) - Current value of timestamp counter. The counter increments whenever a TSISEL field in a Timestamp Control Register is non-zero. The counter will wrap to zero upon overflow and is read only.

Table 554. 0x80002120 - TCNT2 - Interrupt Timestamp 2 Counter register

31	0
TCNT	
0	
r	

31:0 Timestamp Counter (TCNT) - Current value of timestamp counter. The counter increments whenever a TSISEL field in a Timestamp Control Register is non-zero. The counter will wrap to zero upon overflow and is read only.

Table 555. 0x80002130 - TCNT3 - Interrupt Timestamp 3 Counter register

31	0
TCNT	
0	
r	

31:0 Timestamp Counter (TCNT) - Current value of timestamp counter. The counter increments whenever a TSISEL field in a Timestamp Control Register is non-zero. The counter will wrap to zero upon overflow and is read only.

40.3.12 Timestamp Control Register

Table 556. 0x80002104 - ITSTMPC0 - Timestamp 0 Control Register

31 27	26	25	24 6	5	4 0
TSTAMP	S1	S2	RESERVED	KS	TSISEL
0x4	0	0	0	0	0
r	wc	wc	r	rw	rw

31:27 Number of timestamp register sets (TSTAMP) - The number of available timestamp register sets.

26 Assertion Stamped (S1) - Set to '1' when the assertion of the selected line has received a timestamp. This bit is cleared by writing '1' to its position. Writes of '0' have no effect.

25 Acknowledge Stamped (S2) - Set to '1' when the processor acknowledge of the selected interrupt has received a timestamp. This bit can be cleared by writing '1' to this position, writes of '0' have no effect. This bit can also be cleared automatically by the core, see description of the KS field below.

24:6 RESERVED

5 Keep Stamp (KS) - If this bit is set to '1' the core will keep the first stamp value for the first interrupt until the S1 and S2 fields are cleared by software. If this bit is set to '0' the core will time stamp the most recent interrupt. This also has the effect that the core will automatically clear the S2 field whenever the selected interrupt line is asserted and thereby also stamp the next acknowledge of the interrupt.

4:0 Timestamp Interrupt Select (TSISEL) - This field selects the interrupt number (1 - 31) to timestamp.

Table 557. 0x80002114 - ITSTMPC1 - Timestamp 1 Control Register

31 27	26	25	24 6	5	4 0
TSTAMP	S1	S2	RESERVED	KS	TSISEL
0x4	0	0	0	0	0
r	wc	wc	r	rw	rw

31:27 Number of timestamp register sets (TSTAMP) - The number of available timestamp register sets.

26 Assertion Stamped (S1) - Set to '1' when the assertion of the selected line has received a timestamp. This bit is cleared by writing '1' to its position. Writes of '0' have no effect.

25 Acknowledge Stamped (S2) - Set to '1' when the processor acknowledge of the selected interrupt has received a timestamp. This bit can be cleared by writing '1' to this position, writes of '0' have no effect. This bit can also be cleared automatically by the core, see description of the KS field below.

24:6 RESERVED

5 Keep Stamp (KS) - If this bit is set to '1' the core will keep the first stamp value for the first interrupt until the S1 and S2 fields are cleared by software. If this bit is set to '0' the core will time stamp the most recent interrupt. This also has the effect that the core will automatically clear the S2 field whenever the selected interrupt line is asserted and thereby also stamp the next acknowledge of the interrupt.

4:0 Timestamp Interrupt Select (TSISEL) - This field selects the interrupt number (1 - 31) to timestamp.

Table 558. 0x80002124 - ITSTMPC2 - Timestamp 2 Control Register

31 27	26	25	24 6	5	4 0
TSTAMP	S1	S2	RESERVED	KS	TSISEL
0x4	0	0	0	0	0
r	wc	wc	r	rw	rw

31:27 Number of timestamp register sets (TSTAMP) - The number of available timestamp register sets.

26 Assertion Stamped (S1) - Set to ‘1’ when the assertion of the selected line has received a timestamp. This bit is cleared by writing ‘1’ to its position. Writes of ‘0’ have no effect.

25 Acknowledge Stamped (S2) - Set to ‘1’ when the processor acknowledge of the selected interrupt has received a timestamp. This bit can be cleared by writing ‘1’ to this position, writes of ‘0’ have no effect. This bit can also be cleared automatically by the core, see description of the KS field below.

24:6 RESERVED

5 Keep Stamp (KS) - If this bit is set to ‘1’ the core will keep the first stamp value for the first interrupt until the S1 and S2 fields are cleared by software. If this bit is set to ‘0’ the core will time stamp the most recent interrupt. This also has the effect that the core will automatically clear the S2 field whenever the selected interrupt line is asserted and thereby also stamp the next acknowledge of the interrupt.

4:0 Timestamp Interrupt Select (TSISEL) - This field selects the interrupt number (1 - 31) to timestamp.

Table 559. 0x80002104 - ITSTMPC3 - Timestamp 3 Control Register

31 27	26	25	24 6	5	4 0
TSTAMP	S1	S2	RESERVED	KS	TSISEL
0x4	0	0	0	0	0
r	wc	wc	r	rw	rw

31:27 Number of timestamp register sets (TSTAMP) - The number of available timestamp register sets.

26 Assertion Stamped (S1) - Set to ‘1’ when the assertion of the selected line has received a timestamp. This bit is cleared by writing ‘1’ to its position. Writes of ‘0’ have no effect.

25 Acknowledge Stamped (S2) - Set to ‘1’ when the processor acknowledge of the selected interrupt has received a timestamp. This bit can be cleared by writing ‘1’ to this position, writes of ‘0’ have no effect. This bit can also be cleared automatically by the core, see description of the KS field below.

24:6 RESERVED

5 Keep Stamp (KS) - If this bit is set to ‘1’ the core will keep the first stamp value for the first interrupt until the S1 and S2 fields are cleared by software. If this bit is set to ‘0’ the core will time stamp the most recent interrupt. This also has the effect that the core will automatically clear the S2 field whenever the selected interrupt line is asserted and thereby also stamp the next acknowledge of the interrupt.

4:0 Timestamp Interrupt Select (TSISEL) - This field selects the interrupt number (1 - 31) to timestamp.

40.3.13 Interrupt Assertion Timestamp Register

Table 560. 0x80002108 - ITSTMPAS0 - Interrupt Assertion Timestamp 0 register

31	0
TASSERTION	
0	
r	

31:0 Timestamp of Assertion (TASSERTION) - The current Timestamp Counter value is saved in this register when timestamping is enabled and the interrupt line selected by TSISEL is asserted.

Table 561. 0x80002118 - ITSTMPAS1 - Interrupt Assertion Timestamp 1 register

31	0
TASSERTION	
0	
r	

31:0 Timestamp of Assertion (TASSERTION) - The current Timestamp Counter value is saved in this register when timestamping is enabled and the interrupt line selected by TSISEL is asserted.

Table 562. 0x80002128 - ITSTMPAS2 - Interrupt Assertion Timestamp 2 register

31	0
TASSERTION	
0	
r	

31:0 Timestamp of Assertion (TASSERTION) - The current Timestamp Counter value is saved in this register when timestamping is enabled and the interrupt line selected by TSISEL is asserted.

Table 563. 0x80002138 - ITSTMPAS3 - Interrupt Assertion Timestamp 3 register

31	0
TASSERTION	
0	
r	

31:0 Timestamp of Assertion (TASSERTION) - The current Timestamp Counter value is saved in this register when timestamping is enabled and the interrupt line selected by TSISEL is asserted.

40.3.14 Interrupt Acknowledge Timestamp Register

Table 564. 0x8000210C - ITSTMPAS0 - Interrupt Acknowledge Timestamp 0 register

31	0
TACKNOWLEDGE	
0	
r	

31:0 Timestamp of Acknowledge (TACKNOWLEDGE) - The current Timestamp Counter value is saved in this register when timestamping is enabled, the Acknowledge Stamped (S2) field is ‘0’, and the interrupt selected by TSISEL is acknowledged by a processor connected to the interrupt controller.

Table 565. 0x8000211C - ITSTMPAS1 - Interrupt Acknowledge Timestamp 1 register

31	0
TACKNOWLEDGE	
0	
r	

31:0 Timestamp of Acknowledge (TACKNOWLEDGE) - The current Timestamp Counter value is saved in this register when timestamping is enabled, the Acknowledge Stamped (S2) field is ‘0’, and the interrupt selected by TSISEL is acknowledged by a processor connected to the interrupt controller.

Table 566. 0x8000212C - ITSTMPAS2 - Interrupt Acknowledge Timestamp 2 register

31	0
TACKNOWLEDGE	
0	
r	

31:0 Timestamp of Acknowledge (TACKNOWLEDGE) - The current Timestamp Counter value is saved in this register when timestamping is enabled, the Acknowledge Stamped (S2) field is ‘0’, and the interrupt selected by TSISEL is acknowledged by a processor connected to the interrupt controller.

Table 567. 0x8000213C - ITSTMPAS3 - Interrupt Acknowledge Timestamp 3 register

31	0
TACKNOWLEDGE	
0	
r	

31:0 Timestamp of Acknowledge (TACKNOWLEDGE) - The current Timestamp Counter value is saved in this register when timestamping is enabled, the Acknowledge Stamped (S2) field is ‘0’, and the interrupt selected by TSISEL is acknowledged by a processor connected to the interrupt controller.

40.3.15 Processor Boot Address Register

Table 568. 0x80002200 - PROCBOOTADR -Processor boot address register

31 … 3	2 … 1	0
BOOTADDR[31:3]	RES	AS
-	-	-
w	-	w

31:3	Entry point for booting up processor, 8-byte aligned
2:1	Reserved (write 0)
0	Start processor immediately after setting address

* For usage of this register see chapter 40.2.7

40.3.16 Interrupt Map Register

Table 569. 0x80002300 + 0x4 * *n* - IRQMAP*n* - Interrupt map register

31 … 24	23 … 16	15 … 8	7 … 0
IID[n*4+0]	IID[n*4+1]	IID[n*4+2]	IID[n*4+3]
**	**	**	**
rw	rw	rw	rw

31 : 24	Interrupt bus map *n* (ID[*n**4+0]) - Map register for bus interrupt line [*n**4+0]
23 : 16	Interrupt bus map *n* (ID[*n**4+1]) - Map register for bus interrupt line [*n**4+1]
15 : 8	Interrupt bus map *n* (ID[*n**4+2]) - Map register for bus interrupt line [*n**4+2]
7 : 0	Interrupt bus map *n* (ID[*n**4+3]) - Map register for bus interrupt line [*n**4+3]

* Number of interrupts in LEON3FT microcontroller is 64 hence *n* is 16

** Default values for interrupt bus ID is found in table 570

Table 570. Remap default settings (TBC)

Address	Register	Bit field	Default	Interrupt Line	Core
0x80002300	IRQMAP0	ID0	-	0	n/a
		ID1	1	1	Extended
		ID2	2	2	GRPWRX
		ID3	3	3	GRPWTX/GRSCRUB
0x80002304	IRQMAP1	ID4	4	4	GR1553
		ID5	5	5	GRSPWROUTER
		ID6	6	6	GRDMAC
		ID7	7	7	I2CSLV0
0x80002308	IRQMAP2	ID8	8	8	unused
		ID9	9	9	GPTIMER0
		ID10	10	10	GPTIMER0
		ID11	11	11	GPTIMER0/Timer32_A
0x8000230C	IRQMAP3	ID12	12	12	GPTIMER0/Timer32_B
		ID13	13	13	GPTIMER0/RegSYNC
		ID14	14	14	GPTIMER0/Timer27_0/ Timer27_1/PROTA
		ID15	15	15	GPTIMER0/PROTB
0x80002310	IRQMAP4	ID16	16	16	PROTC
		ID17	17	17	GRGPIO0/SYNC
		ID18	18	18	GRGPIO0/REGSYNCA
		ID19	19	19	GRGPIO0/APWMA0
0x80002314	IRQMAP5	ID20	20	20	GRGPIO0/APWMA1
		ID21	21	21	GRCANFD/APWMA2
		ID22	22	22	GRCANFD/APWMA3
		ID23	23	23	GRCANFD/TIMESTAMP
0x80002318	IRQMAP6	ID24	24	24	APBUART0/REGSYCAB0
		ID25	25	25	APBUART1/APWMAB0
		ID26	26	26	DAC0/REGSYNCAB1
		ID27	27	27	DAC1/APWMAB1
0x8000231C	IRQMAP7	ID28	28	28	ADC0/REGSYNCAB2
		ID29	29	29	ADC1/APWMAB2
		ID30	30	30	ADC2/REGSYNCAB3
		ID31	31	31	ADC3/APWMAB3

Address	Register	Bit field	Default	Interrupt Line	Core
0x80002320	IRQMAP8	ID32	28	32	REGSYNCCF0
		ID33	29	33	APWMCF0 0/FIRCCLK-GEN
		ID34	30	34	APWMCF0 1/FIR0
		ID35	31	35	APWMCF0 2/FIR1
0x80002324	IRQMAP9	ID36	26	36	DAC2/FIR2/APWMCF0 3
		ID37	27	37	DAC3/FIR3/REGSYNC-CF1
		ID38	16	38	GRGPIO1/FIR4/APW-MCF1 0
		ID39	17	39	GRGPIO1/FIR5/APW-MCF1 1
0x80002328	IRQMAP10	ID40	18	40	GRGPIO1/FIR6/APW-MCF1 2
		ID41	19	41	GRGPIO1/FIR7/APW-MCF1 3
		ID42	3	42	APBUART2/REGSYNC-CF2/CLOCKDET 0/ CLOCKDET 1
		ID43	4	43	GRSPWTDP/APWMDAC A/APWMCF2 0/APWMG* 0
0x8000232C	IRQMAP11	ID44	5	44	APBUART3/APWMDAC B/APWMCF2 1/APWMG* 1
		ID45	6	45	APBUART4/APWMDAC C/APWMCF2 2/APWMG* 2
		ID46	7	46	APBUART5/APWMDAC D/APWMCF2 3/APWMG* 3
		ID47	8	47	I2CSLV1/I2C2AHB / COMP A0/APWMG0
0x80002330	IRQMAP12	ID48	11	48	SPICTRL0/COMPA1/ APWMG1
		ID49	12	49	SPICTRL1/COMPA2/ APWMG2
		ID50	13	50	I2CM0/ COMPA3/APW-MG3
		ID51	14	51	I2CM1/ COMPB 0/APW-MG4

Address	Register	Bit field	Default	Interrupt Line	Core
0x80002334	IRQMAP13	ID52	2	52	SPIMCTRL0/ SPIMCTRL1/COMPB1/ APWMG5
		ID53	20	53	GPTIMER1/COMPB2/ APWMG6
		ID54	21	54	GPTIMER1/COMPB3/ APWMG7
		ID55	22	55	GPTIMER1/APWMG8
0x80002338	IRQMAP14	ID56	23	56	GPTIMER1/APWMG9
		ID57	24	57	GPTIMER1/APWMG10
		ID58	25	58	GPTIMER1/APWMG11
		ID59	26	59	GPTIMER1
0x8000233C	IRQMAP15	ID60	16	60	RTA0
		ID61	17	61	RTA1 / GRETH
		ID62	18	62	SPI2AHB / SPI4S
		ID63	19	63	PLL/BO/AHBSTAT/ DLRAM, ILRAM/GRG-PRBANK/MEMSCRUB/ MEMPROT/LVDSIO

41 LEON3 Statistics Unit

41.1 Overview

The LEON3 Statistics Unit (L3STAT) is used count events in the LEON3 processor and the AHB bus, in order to create performance statistics for various software applications.

The L3STAT core in the LEON3FT microcontroller consists of 4 32-bit counters, which increment on a certain event. The counters roll over to zero when reaching their maximum value, but can also be automatically cleared on reading to facilitate statistics building over longer periods. Each counter has a control register where the event type is selected. The table 571 below shows the event types that can be monitored.

Table 571. Event types and IDs for Main and DMA AMBA bus

ID	Event description
Processor events:	
0x10	Data write buffer hold
0x11	Total instruction count
0x12	Integer instructions
0x13	Floating-point unit instruction count
0x14	Branch prediction miss
0x15	Execution time, excluding debug mode
0x17	AHB utilization (per AHB master)
0x18	AHB utilization (total, master/CPU selection is ignored)
0x22	Integer branches
0x28	CALL instructions
0x30	Regular type 2 instructions
0x38	LOAD and STORE instructions
0x39	LOAD instructions
0x3A	STORE instructions
AHB events	
0x40	AHB IDLE cycles.
0x41	AHB BUSY cycles.
0x42	AHB NON-SEQUENTIAL transfers. Filtered on CPU/AHBM if SU(1) = ‘1’
0x43	AHB SEQUENTIAL transfers. Filtered on CPU/AHBM if SU(1) = ‘1’
0x44	AHB read accesses. Filtered on CPU/AHBM if SU(1) = ‘1’
0x45	AHB write accesses. Filtered on CPU/AHBM if SU(1) = ‘1’
0x46	AHB byte accesses. Filtered on CPU/AHBM if SU(1) = ‘1’
0x47	AHB half-word accesses. Filtered on CPU/AHBM if SU(1) = ‘1’
0x48	AHB word accesses. Filtered on CPU/AHBM if SU(1) = ‘1’
0x49	AHB double word accesses. Filtered on CPU/AHBM if SU(1) = ‘1’
0x4A	AHB quad word accesses. Filtered on CPU/AHBM if SU(1) = ‘1’
0x4B	AHB eight word accesses. Filtered on CPU/AHBM if SU(1) = ‘1’
0x4C	AHB waitstates. Filtered on CPU/AHBM if SU(1) = ‘1’
0x4D	AHB RETRY responses. Filtered on CPU/AHBM if SU(1) = ‘1’
0x4E	AHB SPLIT responses. Filtered on CPU/AHBM if SU(1) = ‘1’
0x4F	AHB SPLIT delay. Filtered on CPU/AHBM if SU(1) = ‘1’
0x50	AHB bus locked. Filtered on CPU/AHBM if SU(1) = ‘1’
0x51-0x5F	Reserved

Table 571. Event types and IDs for Main and DMA AMBA bus

ID	Event description
Implementation specific events:	
0x60	External event correctable error in Data on-chip RAM. The event signal from one slice out of the four memory slices is recorded. Thus, only 16 KiB (i.e., 1/4 of the 64 KiB memory) is covered by the recorded events.
0x61	External event correctable error in Instruction on-chip RAM. The event signal from one slice out of the four memory slices is recorded. Thus, only 16 KiB (i.e., 1/4 of the 64 KiB memory) is covered by the recorded events.
0x62	External event uncorrectable error in Data on-chip RAM. The event signal from one slice out of the four memory slices is recorded. Thus, only 16 KiB (i.e., 1/4 of the 64 KiB memory) is covered by the recorded events.
0x63	External event uncorrectable error in Instruction on-chip RAM. The event signal from one slice out of the four memory slices is recorded. Thus, only 16 KiB (i.e., 1/4 of the 64 KiB memory) is covered by the recorded events.
0x64	External event correctable error in Off-chip RAM/ROM.
0x65	External event correctable error in NVRAM RAM/ROM.
0x66	External event correctable error in SPI PROM0.
0x67	External event correctable error in SPI PROM1.
0x68	Not used
0x69	Not used
0x6A	Not used
0x6B	Not used
0x6C	Not used
0x6D	Not used
0x6E	Not used
0x6F	Not used
AHB events	
0x70	AHB IDLE cycles.
0x71	AHB BUSY cycles. Filtered on CPU/AHBM if SU(1) = ‘1’
0x72	AHB NON-SEQUENTIAL transfers. Filtered on CPU/AHBM if SU(1) = ‘1’
0x73	AHB SEQUENTIAL transfers. Filtered on CPU/AHBM if SU(1) = ‘1’
0x74	AHB read accesses. Filtered on CPU/AHBM if SU(1) = ‘1’
0x75	AHB write accesses. Filtered on CPU/AHBM if SU(1) = ‘1’
0x76	AHB byte accesses. Filtered on CPU/AHBM if SU(1) = ‘1’
0x77	AHB half-word accesses. Filtered on CPU/AHBM if SU(1) = ‘1’
0x78	AHB word accesses. Filtered on CPU/AHBM if SU(1) = ‘1’
0x79	AHB double word accesses. Filtered on CPU/AHBM if SU(1) = ‘1’
0x7A	AHB quad word accesses. Filtered on CPU/AHBM if SU(1) = ‘1’
0x7B	AHB eight word accesses. Filtered on CPU/AHBM if SU(1) = ‘1’
0x7C	AHB waitstates. Filtered on CPU/AHBM if SU(1) = ‘1’
0x7D	AHB RETRY responses. Filtered on CPU/AHBM if SU(1) = ‘1’
0x7E	AHB SPLIT responses. Filtered on CPU/AHBM if SU(1) = ‘1’
0x7F	AHB SPLIT delay. Filtered on CPU/AHBM if SU(1) = ‘1’
Events generated from REQ/GNT signals	
0x80 - 0x8F	Active when master selected by CPU/AHBM field has request asserted while grant is asserted for the master correspoding to the least significant nibble of the event ID. 0x80 is master 0 grant, 0x81 is master 1 grant, .., and so on. For the LEON3FT Microcontroller ID 0x80 to 0x83 is used for the main system bus and 0x80 to 0x87 is used for the DMA bus

Table 571. Event types and IDs for Main and DMA AMBA bus

ID	Event description
0x80*	Request by LEON3FT
0x81*	Request by DMA => Main bridge
0x82*	Request by Scrubber
0x80**	Request by Debug UART
0x81**	Request by 1553 core
0x82**	Request by SpaceWire core
0x83**	Request by CAN core 0
0x84**	Request by CAN core 1
0x85**	Request by AHBUART core
0x86**	Request by Main => DMA bridge
0x87**	Request by PWRX core
0x88**	Request by PWTX core
0x89**	Request by DMA core 0
0x8A**	Request by DMA core 1
0x90 - 0x9F	Active when master selected by CPU/AHBM field has request asserted while grant is deasserted for the master correspoding to the least significant nibble of the event ID. 0x90 is master 0 grant, 0x91 is master 1 grant, .., and so on.

* Only valid for L3STAT for Main system bus

** Only valid for L3STAT for DMA bus

Note that IDs 0x39 (LOAD instructions) and 0x3A (STORE instructions) will both count all LDST and SWAP instructions. The sum of events counted for 0x39 and 0x3A may therefore be larger than the number of events counted with ID 0x38 (LOAD and STORE instructions).

41.2 Using the LEON3 statistics unit

The debug monitor GRMON3 has build-in support for using LEON3 statistical unit. For more information see chapter for using the LEON3 statistical unit in the GRMON3 User's Manual [GRMON3].

41.3 Registers

The L3STAT core is programmed through registers mapped into APB address space.

Table 572. L3STAT counter control register*

APB address offset	Register
0x0	Counter 0 value register
0x4	Counter 1 value register
0x8	Counter 2 value register
0xC	Counter 3 value register
0x100	Counter 0 control register
0x104	Counter 1 control register
0x108	Counter 2 control register
0x10C	Counter 3 control register
0x200	Counter 0 max/latch register
0x204	Counter 1 max/latch register
0x208	Counter 2 max/latch register
0x20C	Counter 3 max/latch register
0x300	Timestamp register

41.3.1 Counter Value Register

Table 573. 0x00+n.4 - CVALn - Counter value register

31	0
CVAL	
NR	
rw	

31: 0 Counter value (CVAL) - This register holds the current value of the counter. If the Counter control register field CD is '1', then the value displayed by this register will be the maximum counter value reached with the settings in the counter's control register. Writing to this register will write both to the counter and the hold register for the maximum counter value.

41.3.2 Counter Control Register

Table 574. 0x100+n.4 - CCTRLn - Counter control register

31 28	27 23	22	21	20	19	18	17	16	15 14	13	12	11 4	3 0
NCPU	NCNT	MC	IA	DS	EE	AE	EL	CD	SU	CL	EN	EVENT ID	CPU/AHBM
0	3	1	1	1	1	1	NR	NR	NR	NR	0	NR	NR
r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw

31: 28 Number of CPU (NCPU) - Number of supported processors - 1

27: 23 Number of counters (NCNT) - Number of implemented counters - 1

22 Maximum count (MC) - This field is '1' indicating that the counter has support for keeping the maximum count value

21 Internal AHB count (IA) - This field is '1' indicating that the core supports events 0x17 and 0x18

20 DSU support (DS) - This field is '1' indicating that the core supports events 0x40-0x5F

19 External events (EE) - This field is '1' indicating that the core supports external events (events 0x60 - 0x6F)

18 AHBTRACE Events (AE) - This field is '1' indicating that the core supports events 0x70 - 0x7F.

17 Event Level (EL) - The value of this field determines the level where the counter keeps running when the CD field below has been set to '1'. If this field is '0' the counter will count the time between event assertions. If this field is '1' the counter will count the cycles where the event is asserted.

16 Count maximum duration (CD) - If this bit is set to '1' the core will save the maximum time the selected event has been at the level specified by the EL field. This also means that the counter will be reset when the event is activated or deactivated depending on the value of the EL field.

When this bit is set to '1', the value shown in the counter value register will be the maximum current value which may be different from the current value of the counter.

15: 14 Supervisor/User mode filter (SU) - "01" - Only count supervisor mode events, "10" - Only count user mode events, others values - Count events regardless of user or supervisor mode. This setting only applies to events 0x0 - 0x3A.

When SU = "1x" only events generated by the CPU/AHB master specified in the CPU/AHBM field will be counted. This applies to events 0x40 - 0x7F.

: 13 Clear counter on read (CL) - If this bit is set the counter will be cleared when the counter's value is read. The register holding the maximum value will also be cleared.

If an event occurs in the same cycle as the counter is cleared by a read then the event will not be counted. The counter latch register can be used to guarantee that no events are lost

12 Enable counter (EN) - Enable counter

11: 4 Event ID to be counted

3: 0 CPU or AHB master to monitor.(CPU/AHBM) - The value of this field does not matter when selecting one of the events coming from the Debug Support Unit or one of the external events.

41.3.3 Counter max/latch Register

Table 575. 0x200+4n - CSVALn - Counter max/latch register

31	0
CSVAL	
NR	
rw*	

31: 0 Counter max/latch value (CSVAL) - This register holds the current value of the counter max/latch register.

If the counter control register field CD is '1', then the value displayed by this register will be the maximum counter value reached with the settings in the counter's control register.

If the counter control register field CD is '0', then the value displayed by this register is the latched (saved) counter value. The counter value is saved whenever a write access is made to the core in address range 0x100 - 0x1FC (all counters are saved simultaneously). If the counter control register CL field is set, then the current counter value will be cleared when the counter value is saved into this register.

41.3.4 Timestamp Register

Table 576. 0x300 - TSTAMP - Timestamp register

31	0
TSTAMP	
NR	
rw*	

31: 0 Timestamp (TSTAMP) - Timestamp taken at latch of counters

42 Memory Scrubber and Status Register

The GR716B microcontroller have 1 AHB Memory Scrubber and Status Register unit (MEMSCRUB).

The MEMSCRUB unit monitors the system main bus or scrubber bus for accesses triggering an error response, and for correctable errors signaled from fault tolerant slaves on the bus. The MEMSCRUB unit can be programmed to scrub memories. The AHB Memory Scrubber and Status Register unit (MEMSCRUB) have a unique AMBA address described in chapter 2.10 for configuration and status.

The AHB Memory Scrubber and Status Register unit (MEMSCRUB) unit is located on AHB bus in the address range from 0xFFF00000 to 0xFFF00FFF.

See units connections in the next drawing. The drawing picture memory locations and functions used for configuration and control.

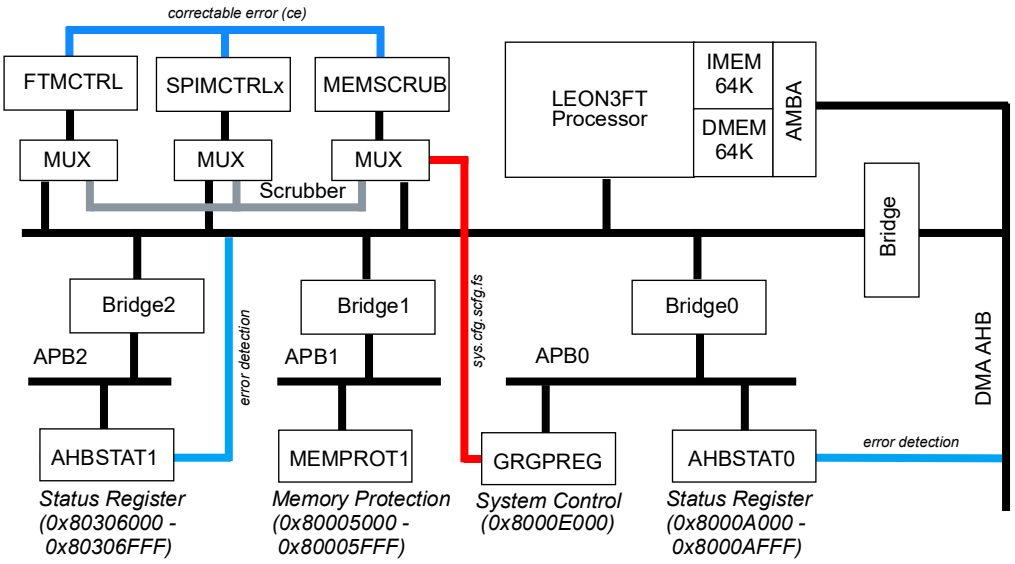


Figure 93. GR716B Scrubber and Status bus connection

42.1 Overview

The memory scrubber monitors an AMBA AHB bus for accesses triggering an error response, and for correctable errors signaled from fault tolerant slaves on the bus. The core can be programmed to scrub a memory area by reading through the memory and writing back the contents using a locked read-write cycle whenever a correctable error is detected. It can also be programmed to initialize a memory area to known values.

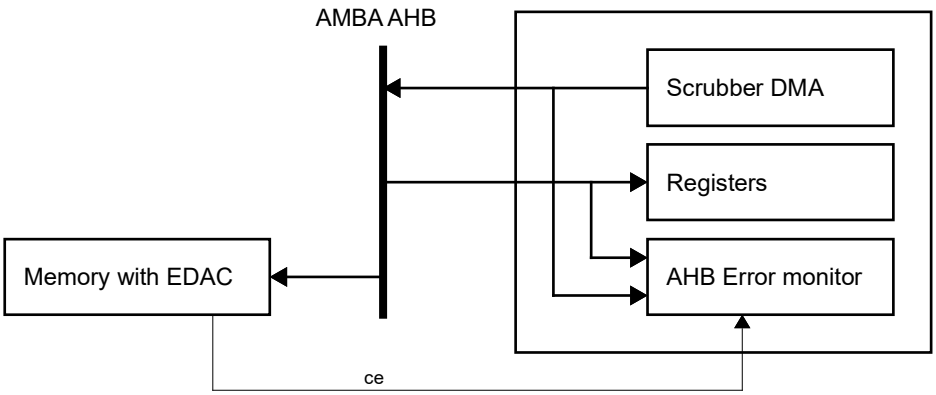


Figure 94. Memory scrubber block diagram

42.2 Operation

42.2.1 Errors

All AMBA AHB bus transactions are monitored and current HADDR, HWRITE, HMASTER and HSIZE values are stored internally. When an error response (HRESP = “01”) is detected, an internal counter is increased. When the counter exceeds a user-selected threshold, the status and address register contents are frozen and the New Error (NE) bit is set to one. At the same time an interrupt is generated, as described hereunder.

The default threshold is zero and enabled on reset so the first error on the bus will generate an interrupt.

Note that many of the fault tolerant units containing EDAC signal an un-correctable error as an AMBA error response, so that it can be detected by the processor as described above.

42.2.2 Correctable errors

Not only error responses on the AHB bus can be detected. Many of the fault tolerant units containing EDAC have a correctable error signal which is asserted each time a correctable error is detected. When such an error is detected, the effect will be the same as for an AHB error response. The only difference is that the Correctable Error (CE) bit in the status register is set to one when a correctable error is detected. Correctable and uncorrectable errors use separate counters and threshold values.

When the CE bit is set, the interrupt routine can acquire the address containing the correctable error from the failing address register and correct it. When it is finished it resets the CE bit and the monitoring becomes active again. Interrupt handling is described in detail hereunder.

42.2.3 Scrubbing

The memory scrubber can be commanded to scrub a certain memory area, by writing a start and end address to the scrubbers start/end registers, followed by writing “00” to the scrub mode field and ‘1’ to the scrub enable bit in the scrubber control register.

After starting, the core will proceed to read the memory region in bursts. The burst size is fixed and typically tuned to match the cache-line size or native block size of the slave. When a correctable error is detected, the scrubber performs a locked read-write cycle to correct the error, and then resumes the scrub operation.

If the correctable error detected is in the middle of a burst, the following read in the burst is completed before the read-write cycle begins. The core can handle the special case where that access also had a correctable error within the same locked scrub cycle.

If an uncorrectable error is detected, that location is left untouched.

Note that the status register functionality is running in parallel with the scrubber, so correctable and uncorrectable errors will be logged as usual. To prevent double logging, the core masks out the (expected) correctable error arising during the locked correction cycle.

To allow normal access to the bus, the core sleeps for a number of cycles between each burst. The number of cycles can be adjusted in the config register.

If the ID bit is set in the config register, the core will interrupt when the complete scrub is done.

42.2.4 Scrubber error counters

The core keeps track of the number of correctable errors detected during the current scrub run and the number of errors detected during processing of the current “count block”. The size of the count block is a fixed power of two equal or larger than the burst length .

The core can be set up to interrupt when the counters exceed given thresholds. When this happens, the NE bit, plus one of the SEC/SBC bits, is set in the status register.

42.2.5 External start and clear

If the ES bit is set in the config register, the scrub enable bit is set automatically when the start input signal goes high. This can be used to set up periodic scrubbing.

The external input signal clrerr can be used to clear the global error counters. If this is connected to a timer, it is possible to count errors that have occurred within a specific unit of time. This signal can be disabled through the EC bit in the config register.

42.2.6 Memory regeneration

The regeneration mode performs the same basic function as the scrub mode, but is optimised for the case where many (or all) locations have correctable errors.

In this mode, the whole memory area selected is scrubbed using locked read/write bursts.

If an uncorrectable error is encountered during the read burst, that burst block is processed once again using the regular scrub routine, and the regeneration mode resumes on the following block. This avoids overwriting uncorrectable error locations.

42.2.7 Initialization

The scrubber can be used to write a pre-defined pattern to a block of memory. This is often necessary on EDAC memory before it can be used.

Before running the initialization, the pattern to be written to memory should be written into the scrubber initialization data register. The pattern has the same size as the burst length, so the corresponding number of writes to the initialization data register must be made.

42.2.8 Interrupts

After an interrupt is generated, either the NE bit or the DONE bit in the status register is set, to indicate which type of event caused the interrupt.

The normal procedure is that an interrupt routine handles the error with the aid of the information in the status registers. When it is finished it resets the NE bit in the AHB status register or the DONE bit in the scrubber status register, and the monitoring becomes active again. Error interrupts can be generated for both AMBA error responses and correctable errors as described above.

42.2.9 Mode switching

Switching between scrubbing and regeneration modes can be done on the fly during a scrub by modifying the MODE field in the scrubber configuration register. The mode change will take effect on the following scrub burst.

If the address range needs to be changed, then the core should be stopped before updating the registers. This is done by clearing the SCEN bit, and waiting for the ACTIVE bit in the status register to go low. An exception is when making the range larger (i.e. increasing the end address or decreasing the start address), as this can be done on the fly.

42.2.10 Dual range support

The scrubber can work over two non-overlapping memory ranges. This feature is enabled by writing the start/end addresses of the second range into the scrubber's second range start/end registers and setting the SERA bit in the configuration register. The two address ranges should not overlap.

42.3 Registers

The core is programmed through registers mapped into an I/O region in the AHB address space. Only 32-bit accesses are supported.

*Table 577.*Memory scrubber registers

AHB address offset	Registers
AHB Memory Scrubber and Status Register unit (MEMSCRUB)	
0xFFF00000	AHB Status register
0xFFF00004	AHB Failing address register
0xFFF00008	AHB Error configuration register
0xFFF0000C	Reserved
0xFFF00010	Scrubber status register
0xFFF00014	Scrubber configuration register
0xFFF00018	Scrubber range low address register
0xFFF0001C	Scrubber range high address register
0xFFF00020	Scrubber position register
0xFFF00024	Scrubber error threshold register
0xFFF00028	Scrubber initialization data register
0xFFF0002C	Scrubber second range start address register
0xFFF00030	Scrubber second range end address register

42.3.1 AHB Status Register

Table 578. 0x00 - AHBS - AHB Status register

31 22	21 14	13	12	11	10	9	8	7	6 3	2 0
CECNT	UECNT	DONE	RES	SEC	SBC	CE	NE	HWRITE	HMASTER	HSIZE
0	0	0	0	0	0	0	0	NR	NR	NR
rw	rw	r	r	rw	rw	rw	rw	r	r	r

31: 22 CECNT: Global correctable error count

21: 14 UECNT: Global uncorrectable error count

13 DONE: Task completed. (read-only)

This is a read-only copy of the DONE bit in the status register.

12 RESERVED

11 SEC: Scrubber error counter threshold exceeded. Asserted together with NE.

10 SBC: Scrubber block error counter threshold exceeded. Asserted together with NE.

9 CE: Correctable Error. Set if the detected error was caused by a correctable error and zero otherwise.

8 NE: New Error. Deasserted at start-up and after reset. Asserted when an error is detected. Reset by writing a zero to it.

7 The HWRITE signal of the AHB transaction that caused the error.

6: 3 The HMASTER signal of the AHB transaction that caused the error.

2: 0 The HSIZE signal of the AHB transaction that caused the error

42.3.2 AHB Failing Address Register

Table 579. 0x04 - AHBFAR - AHB Failing address register

31 0
AHB FAILING ADDRESS
NR
r

31: 0 The HADDR signal of the AHB transaction that caused the error.

42.3.3 AHB Error Configuration Register

Table 580. 0x08 - AHBERC - AHB Error configuration register

31 22	21 14	13 2	1	0
CORRECTABLE ERROR COUNT THRESHOLD	UNCORR. ERROR COUNT THRESH.	RESERVED	CECTE	UECTE
0	0	0	0	0
rw	rw	r	rw	rw

31: 22 Interrupt threshold value for global correctable error count

21: 14 Interrupt threshold value for global uncorrectable error count

13: 2 RESERVED

1 CECTE: Correctable error count threshold enable

0 UECTE: Uncorrectable error count threshold enable

42.3.4 Scrubber Status Register

Table 581. 0x10 - STAT - Scrubber status register

31 22	21 14	13	12 5	4 1	0
SCRUB RUN ERROR COUNT	BLOCK ERROR COUNT	DONE	RESERVED	BURSTLEN	ACTIVE
0	0	0	0	*	0
r	r	wc	r	r	r

31: 22	Number of correctable errors in current scrub run (read-only)
21: 14	Number of correctable errors in current block (read-only)
13	DONE: Task completed. Needs to be cleared (by writing zero) before a new task completed interrupt can occur.
12: 5	RESERVED
4: 1	Burst length in 2-log of AHB bus cycles; “0000”=1, “0001”=2, “0010”=4, “0011”=8, ...
0	Current state: 0=Idle, 1=Running (read-only)

42.3.5 Scrubber Configuration Register

*Table 582.*0x14 - CONFIG - Scrubber configuration register

31 16	15 8	7	6	5	4	3 2	1	0
RESERVED	DELAY	IRQD	EC	SERA	LOOP	MODE	ES	SCEN
0	0	0	0	0	0	0	0	0
r	rw	rw	r	rw	rw	rw	rw	rw

31: 16	RESERVED
15: 8	Delay time between processed blocks, in cycles
7	Interrupt when scrubber has finished
6	External clear counter enable
5	Second memory range enable
4	Loop mode, restart scrubber when run finishes
3: 2	Mode (00=Scrub, 01=Regenerate, 10=Initialize, 11=Undefined)
1	External start enable
0	Enable

42.3.6 Scrubber Range Low Address Register

Table 583. 0x18 - RANGEL - Scrubber range low address register

31 0
SCRUBBER RANGE LOW ADDRESS
0
rw

31: 0	The lowest address in the range to be scrubbed The address bits below the burst size alignment are constant ‘0’

42.3.7 Scrubber Range High Address Register

Table 584. 0x1C - RANGEH - Scrubber range high address register

31	0
SCRUBBER RANGE HIGH ADDRESS	
0	
rw	

31: 0 The highest address in the range to be scrubbed

The address bits below the burst size alignment are constant ‘1’

42.3.8 Scrubber Position Register

Table 585. 0x20 - POS - Scrubber position register

31	0
SCRUBBER POSITION	
0	
rw	

31: 0 The current position of the scrubber while active, otherwise zero.

The address bits below the burst size alignment are constant ‘0’

42.3.9 Scrubber Error Threshold Register

Table 586. 0x24 - ERROR - Scrubber error threshold register

31 22	21 14	13 2	1	0
RECT	BECT	RESERVED	RECTE	BECTE
0	0	0	0	0
rw	rw	r	rw	rw

31: 22 Interrupt threshold value for current scrub run correctable error count

21: 14 Interrupt threshold value for current scrub block correctable error count

13: 2 RESERVED

1 RECTE: Scrub run correctable error count threshold enable

0 BECTE: Scrub block uncorrectable error count threshold enable

42.3.10 Scrubber Initialization Data Register

Table 587. 0x28 - INIT - Scrubber initialization data register (write-only)

31 0
SCRUBBER INITIALIZATION DATA
-
w

31: 0 Part of data pattern to be written in initialization mode. A write operation assigns the first part of the buffer and moves the rest of the words in the buffer one step.

42.3.11 Scrubber Second Range Low Address Register

Table 588. 0x2C - RANGEL2 - Scrubber second range low address register

31 0
SCRUBBER RANGE LOW ADDRESS
0
rw*

31: 0 The lowest address in the second range to be scrubbed

The address bits below the burst size alignment are constant ‘0’

42.3.12 Scrubber Second Range High Address Register

Table 589. 0x30 - RANGEH2 - Scrubber second range high address register

31 0
SCRUBBER RANGE HIGH ADDRESS
0
rw*

31: 0 The highest address in the second range to be scrubbed

The address bits below the burst size alignment are constant ‘1’

43 SPI to AHB bridge

The GR716B microcontroller comprises a SPI to AHB bridge (SPI2AHB). The SPI to AHB bridge controls its own external pins and has a unique AMBA address described in chapter 2.10. The SPI to AHB bridge is connected to external pins via the IOMUX.

The control and status registers are located on APB bus in the address range from 0x80104000 to 0x80104FFF. See SPI to AHB bridge connections in the next drawing. The figure shows memory locations and functions used for SPI2AHB configuration and control.

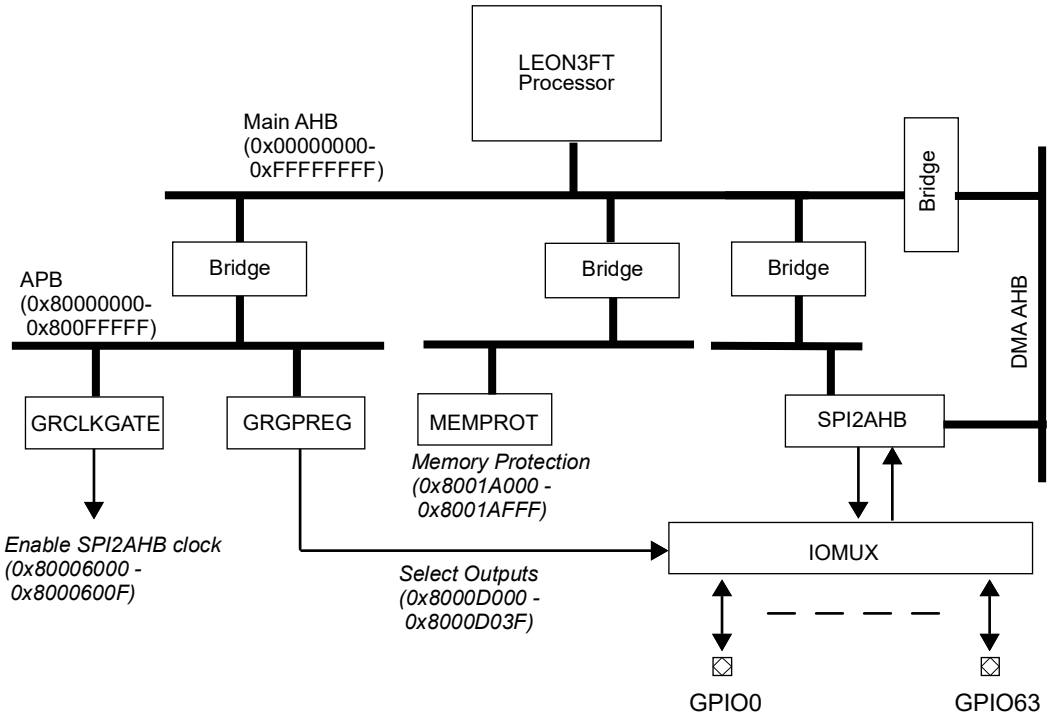


Figure 95. GR716B SPI2AHB bus and pin

The primary clock gating unit **GRCLKGATE** described in section 27 is used to enable/disable the SPI to AHB bridge The unit **GRCLKGATE** can also be used to perform reset of the SPI to AHB bridge. Software must enable clock and release reset described in section 27 before configuration and transmission can start.

External IO selection and configuration is made in the system IO configuration registers (**GRGPREG**) in the address range from 0x8000D000 to 0x8000D03F. See section 7.1 for further information.

The system can be configured to protect and restrict access to the SPI to AHB bridge in the **MEMPROT** unit. See section 47 for more information.

43.1 Overview

The SPI to AHB bridge is an SPI slave that provides a link between a SPI bus (that consists of two data signals, one clock signal and one select signal) and AMBA AHB. On the SPI bus the slave acts as an SPI memory device where accesses to the slave are translated to AMBA accesses. The core can translate SPI accesses to AMBA byte, half-word or word accesses. The access size to use is configurable via the SPI bus.

The core synchronizes the incoming clock and can operate in systems where other SPI devices are driven by asynchronous clocks.

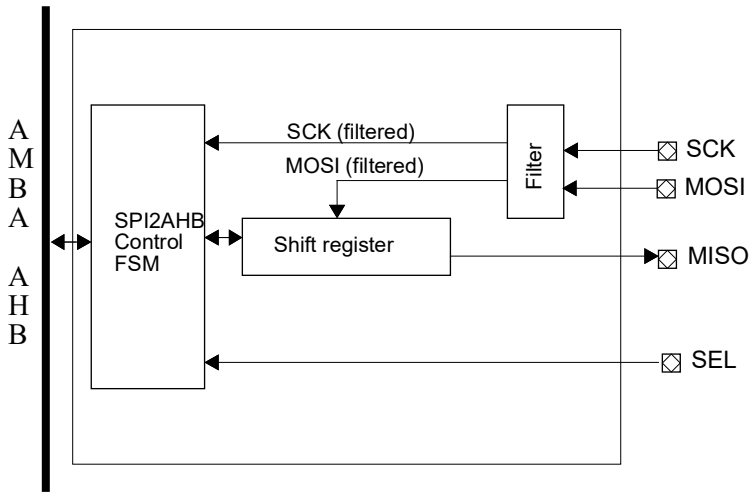


Figure 96. SPI to AHB bridge block diagram

43.2 Transmission protocol

The SPI bus is a full-duplex synchronous serial bus. Transmission starts when a master selects a slave through the slave's Slave Select (SEL) signal and the clock line SCK transitions from its idle state. Data is transferred from the master through the Master-Output-Slave-Input (MOSI) signal and from the slave through the Master-Input-Slave-Output (MISO) signal. In some systems with only one master and one slave, the Slave Select input of the slave may be always active and the master does not need to have a slave select output. This does not apply to this SPI to AHB bridge, the slave select signal must be used to mark the start and end of an operation.

During a transmission on the SPI bus data is either changed or read at a transition of SCK. If data has been read at edge n, data is changed at edge n+1. If data is read at the first transition of SCK the bus is said to have clock phase 0, and if data is changed at the first transition of SCK the bus has clock phase 1. The idle state of SCK may be either high or low. If the idle state of SCK is low, the bus has clock polarity 0 and if the idle state is high the clock polarity is 1. The combined values of clock polarity (CPOL) and clock phase (CPHA) determine the mode of the SPI bus. Figure 97 shows one byte (0x55) being transferred MSb first over the SPI bus under the four different modes. Note that the idle state of the MOSI line is '1' and that CPHA = 0 means that the devices must have data ready before the first transition of SCK. The figure does not include the MISO signal, the behavior of this line is the same as for the MOSI signal. However, due to synchronization the MISO signal will be delayed for a period of time that depends on the system clock frequency.

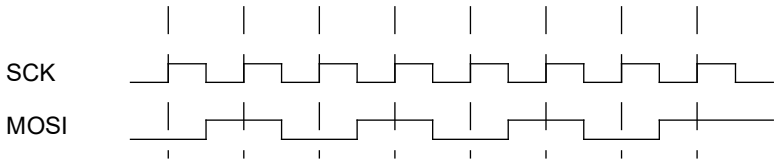


Figure 97. SPI transfer of byte 0x55 in all modes

The SPI to AHB bridge makes use of a protocol commonly used by SPI Flash memory devices. A master first selects the slave via the slave select signal and then issues a one-byte instruction. The instruction is then followed by additional bytes that contain address or data values. All instructions, addresses and data are transmitted with the most significant bit first. All AMBA accesses are done in big endian format. The first byte sent to or from the slave is the most significant byte.

43.3 System clock requirements and sampling

The core samples the incoming SPI SCK clock and does not introduce any additional clock domains into the system. Both the SCK and MOSI lines first pass through two stage synchronizers and are then filtered with a low pass filter.

The synchronizers and filters constrain the minimum system frequency. The core requires the SCK signal to be stable for at least two system clock cycles before the core accepts the SCK value as the new clock value. The core's reaction to transitions will be additionally delayed since both lines are taken through two-stage synchronizers before they are filtered. In order for the slave to be able to output data on the SCK 'change' transition and for this data to reach the master before the next edge the SCK frequency should not be higher than one tenth of the system frequency of core.

The slave select input should be asserted at least two system clock cycles before the SCK line starts transitioning.

43.4 SPI instructions

43.4.1 Overview

The core is controlled from the SPI bus by sending SPI instructions. Some commands require additional bytes in the form of address or data. The core makes use of the same instructions as commonly available SPI Flash devices. Table 590 summarizes the available instructions.

*Table 590.*SPI instructions

Instruction	Description	Instruction code	Additional bytes
RDSR	Read status/control register	0x05	Core responds with register value
WRSR	Write status/control register	0x01	New register value
READ	AHB read access	0x03	Four address bytes, after which core responds with data.
READD	AHB read access with dummy byte	0x0B	Four address butes and one dummy byte, after which core responds with data
WRITE	AHB write access	0x02	Four address bytes followed by data to be written

All instructions, addresses and data are transmitted with the most significant bit first. All AMBA accesses are done in big endian format. The first byte sent to or from the slave is the most significant byte.

43.4.2 SPI status/control register accesses (RDSR/WRSR)

The RDSR and WRSR instructions access the core's SPI status/control register. The register is accessed by issuing the wanted instruction followed by the data byte to be written (WRSR) or any value on the byte in order to shift out the current value of the status/control register (RDSR). The fields available in the SPI status/control register are shown in table 591.

*Table 591.*SPI2AHB SPI status/control register

7	6	5	4	3	2	1	0
Reserved	RAHEAD	PROT	MEXC	DMAACT	MALF	HSIZE	

7 Reserved, always zero (read only)

6 Read ahead (RAHEAD) - When this bit is set the core will make a new access to fetch data as soon as the last current data bit has been moved. Otherwise the core will not attempt the new access until the 'change' transition on SCK. Setting this bit to '1' allows higher SCK frequencies to be used but will also result in a data fetch as soon as the current data has been read out. This means that RAHEAD may not be suitable when accessing FIFO interfaces. (read/write)

Table 591. SPI2AHB SPI status/control register

5	Memory protection triggered (PROT) - '1' if last AHB access was outside the allowed memory area. Updated after each AMBA access (read only). Note that since this bit is updated after each access the RAHEAD = '1' setting may hide errors.
4	Memory exception (MEXC) - '1' if core receives AMBA ERROR response. Updated after each AMBA access (read only). Note that since this bit is updated after each access the RAHEAD = '1' setting may hide errors.
3	DMA active (DMAACT) - '1' if core is currently performing a DMA operation.
2	Malfunction (MALF): This bit is set to one by the core is DMA is not finished when a new byte starts getting shifted. If this bit is set to '1' then the last AHB access was not successful.
1:0	AMBA access size (HSIZE) - Controls the access size that the core will use for AMBA accesses. 0: byte, 1: half-word, 2: word. HSIZE = "11" is illegal.

Reset value: 0x42

43.4.3 Read and write instructions (WRITE and READ/READD)

The READD is the same as the READ instruction with an additional dummy byte inserted after the four address bytes. To perform a read operation on AHB via the SPI bus the following sequence should be performed:

1. Assert slave select
2. Send READ instruction
3. Send four byte AMBA address, the most significant byte is transferred first

3a. Send dummy byte (if READD is used)

4. Read the wanted number of data bytes
5. De-assert slave select

To perform a write access on AHB via the SPI bus, use the following sequence:

1. Assert slave select
2. Send WRITE instruction
3. Send four byte AMBA address, the most significant byte is transferred first
4. Send the wanted number of data bytes
5. De-assert slave select

During consecutive read or write operations, the core will automatically increment the address. The access size (byte, halfword or word) used on AHB is set via the HSIZE field in the SPI status/control register.

The core always respects the access size specified via the HSIZE field. If a write operation writes fewer bytes than what is required to do an access of the specified HSIZE then the write data will be dropped, no access will be made on AHB. If a read operation reads fewer bytes than what is specified by HSIZE then the remaining read data will be dropped when slave select is de-asserted.

The core will not mask any address bits. Therefore it is important that the SPI master respects AMBA rules when performing half-word and word accesses. A half-word access must be aligned on a two byte address boundary (least significant bit of address must be zero) and a word access must be aligned on a four byte boundary (two least significant address bits must be zero).

43.4.4 Memory protection

Default configuration allows full access to the complete AHB address range. The access range can be restricted via configuration registers.

The registers PADDR and PMASK are used to assign the memory protection area's address and mask in the following way:

Protection address, bits 31:16 (PADDR*[31:16]*): ahbaddrh
Protection address, bits 15:0 (PADDR*[15:0]*): ahbaddrl

Protection mask, bits 31:16 (PMASK*[31:16]*): ahbmaskh
Protection mask, bits 15:0 (PMASK*[15:0]*): ahbmaskl

Before the core performs an AMBA access it will perform the check:

(((incoming address) xor (protaddr)) and protmask) /= 0x00000000

If the above expression is true (one or several bits in the incoming address differ from the protection address, and the corresponding mask bits are set to '1') then the access is inhibited. As an example, assume that *protaddr* is 0xA0000000 and *protmask* is 0xF0000000. Since *protmask* only has ones in the most significant nibble, the check above can only be triggered for these bits. The address range of allowed accessed will thus be 0xA0000000 - 0xAFFFFFFF..

The core will set the configuration register bit PROT if an access is attempted outside the allowed address range. This bit is updated on each AHB access and will be cleared by an access inside the allowed range.

43.5 Registers

The core is programmed through registers mapped into APB address space.

*Table 592.*APB registers

APB address offset	Register
0x00	Control register
0x04	Status register
0x08	Protection address register
0x0C	Protection mask register

43.5.1 Control Register

*Table 593.*0x00 - CTRL - Control register

31	2	1	0
RESERVED		IRQEN	EN
0		0	1
r		rw	rw

31 : 2 RESERVED

1 Interrupt enable (IRQEN) - When this bit is set to '1' the core will generate an interrupt each time the DMA field in the status register transitions from '0' to '1'.

0 Core enable (EN) - When this bit is set to '1' the core is enabled and will respond to SPI accesses. Otherwise the core will not react to SPI traffic.

43.5.2 Status Register

*Table 594.*0x04 - STAT - Status register

31	3	2	1	0
RESERVED		PROT	WR	DMA
0		0	0	0
r		wc	r	wc

31 : 3 RESERVED

2 Protection triggered (PROT) - Set to '1' if an access has triggered the memory protection. This bit will remain set until cleared by writing '1' to this position. Note that the other fields in this register will be updated on each AHB access while the PROT bit will remain at '1' once set.

1 Write access (WR) - Last AHB access performed was a write access. This bit is read only.

0 Direct Memory Access (DMA) - This bit gets set to '1' each time the core attempts to perform an AHB access. By setting the IRQEN field in the control register this condition can generate an interrupt. This bit can be cleared by software by writing '1' to this position.

43.5.3 Protection Address Register

*Table 595.*0x08 - PADDR - Protection address register

31	0
PROTADDR	
0x0	
rw	

31 : 0 Protection address (PROTADDR) - Defines the base address for the memory area where the core is allowed to make accesses.

43.5.4 Protection Mask Register

*Table 596.*0x0C - PMASK - Protection mask register

31	0
PROTMASK	
0x0	
rw	

31 : 0 Protection mask (PROTMASK) - Selects which bits in the Protection address register that are used to define the protected memory area.

44 SPI Controller

The GR716B microcontroller comprises two separate SPI controller (SPICTRL) units. Each SPI controller unit controls its own external pins and has a unique AMBA address described in chapter 2.10.

Each SPI controller unit control and status registers are located on the APB bus in the address range from 0x80390000 to 0x803AFFFF. See SPICTRL units connections in the next drawing. The figure shows memory locations and functions used for SPICTRL configuration and control.

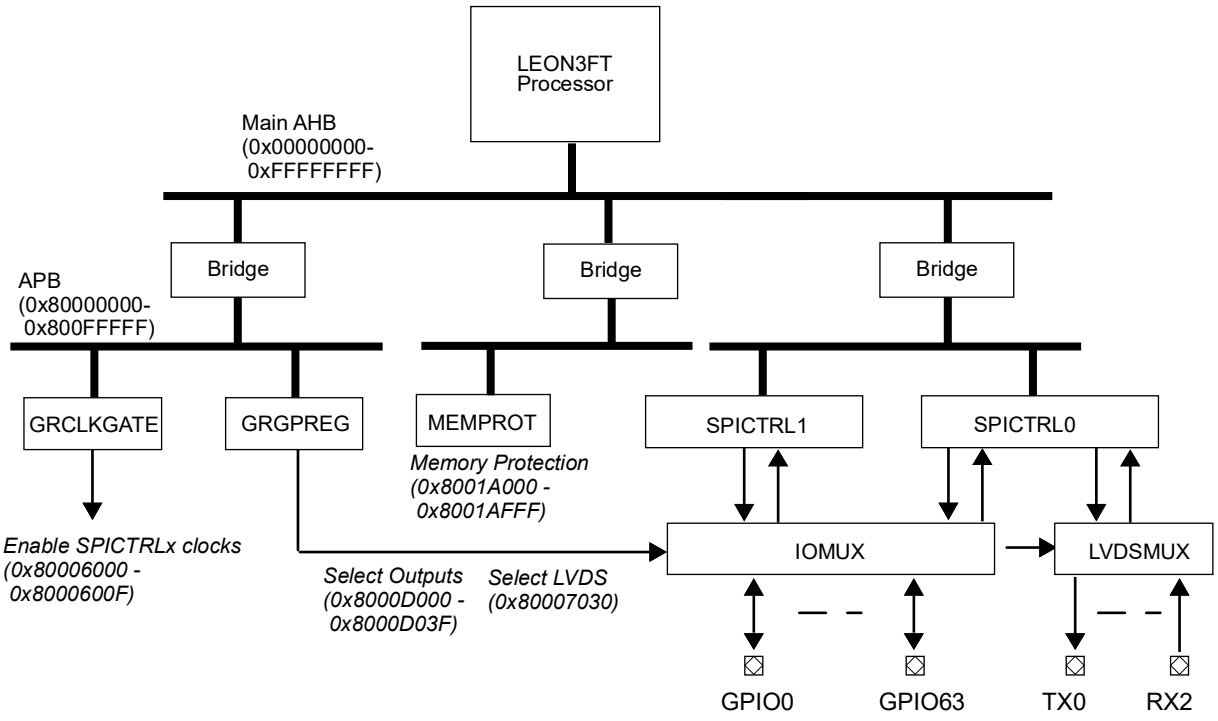


Figure 98. GR716B SPICTRLx bus and pin

The primary clock gating unit **GRCLKGATE** described in section 27 is used to enable/disable SPI controller units. The unit **GRCLKGATE** can also be used to perform reset of individual SPI controller units. Software must enable clock and release reset described in section 27 before SPI configuration and transmission can start.

External IO selection per SPI controller unit is made in the system IO and LVDS configuration register (**GRGPREG**) in the address range from 0x8000D000 to 0x8000D03F and 0x80007030. See section 7.1 for further information.

Each **SPICTRLx** unit controls its own external pins and has a unique AMBA address described in chapter 2.10. SPICTRL unit 0 and 1 has identical configuration and status registers. Configuration and status registers are described in this section 44.3

System can be configured to protect and restrict access to individual SPICTRL unit in the **MEMPROT** unit. See section 47 for more information.

44.1 Overview

The core provides a link between the AMBA APB bus and the Serial Peripheral Interface (SPI) bus and can be dynamically configured to function either as a SPI master or a slave. The SPI bus parameters are highly configurable via registers. Core features also include configurable word length, bit

ordering, clock gap insertion, automatic slave select and automatic periodic transfers of a specified length. All SPI modes are supported and also a 3-wire mode where one bidirectional data line is used. In slave mode the core synchronizes the incoming clock and can operate in systems where other SPI devices are driven by asynchronous clocks.

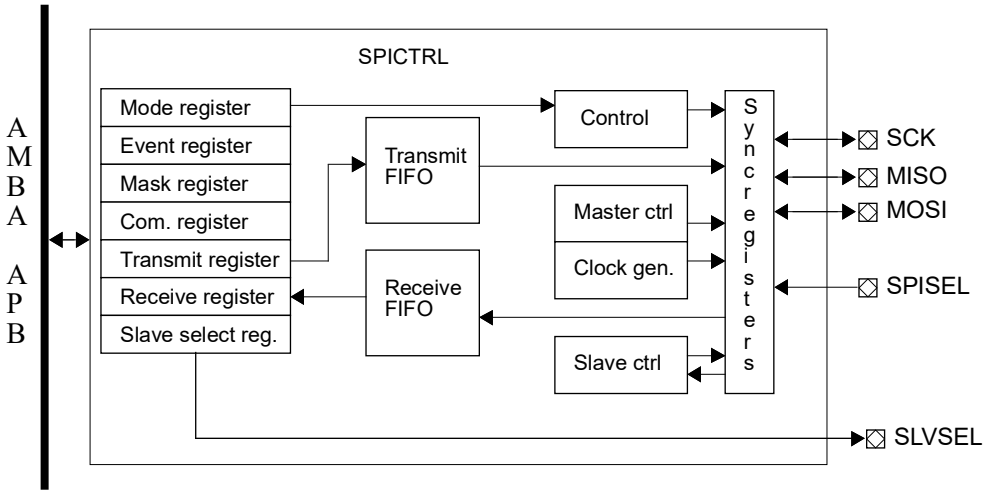


Figure 99. Block diagram

44.2 Operation

44.2.1 SPI transmission protocol

The SPI bus is a full-duplex synchronous serial bus. Transmission starts when a master selects a slave through the slave's Slave Select (SLVSEL) signal and the clock line SCK transitions from its idle state. Data is transferred from the master through the Master-Output-Slave-Input (MOSI) signal and from the slave through the Master-Input-Slave-Output (MISO) signal. In a system with only one master and one slave, the Slave Select input of the slave may be always active and the master does not need to have a slave select output. If the core is configured as a master it will monitor the SPISEL signal to detect collisions with other masters, if SPISEL is activated the master will be disabled.

During a transmission on the SPI bus data is either changed or read at a transition of SCK. If data has been read at edge n, data is changed at edge n+1. If data is read at the first transition of SCK the bus is said to have clock phase 0, and if data is changed at the first transition of SCK the bus has clock phase 1. The idle state of SCK may be either high or low. If the idle state of SCK is low, the bus has clock polarity 0 and if the idle state is high the clock polarity is 1. The combined values of clock polarity (CPOL) and clock phase (CPHA) determine the mode of the SPI bus. Figure 100 shows one byte (0x55) being transferred MSb first over the SPI bus under the four different modes. Note that the idle state of the MOSI line is '1' and that CPHA = 0 means that the devices must have data ready before the first transition of SCK. The figure does not include the MISO signal, the behavior of this line is the same as for the MOSI signal. However, due to synchronization issues the MISO signal will be delayed when the core is operating in slave mode, please see section 44.2.5 for details.

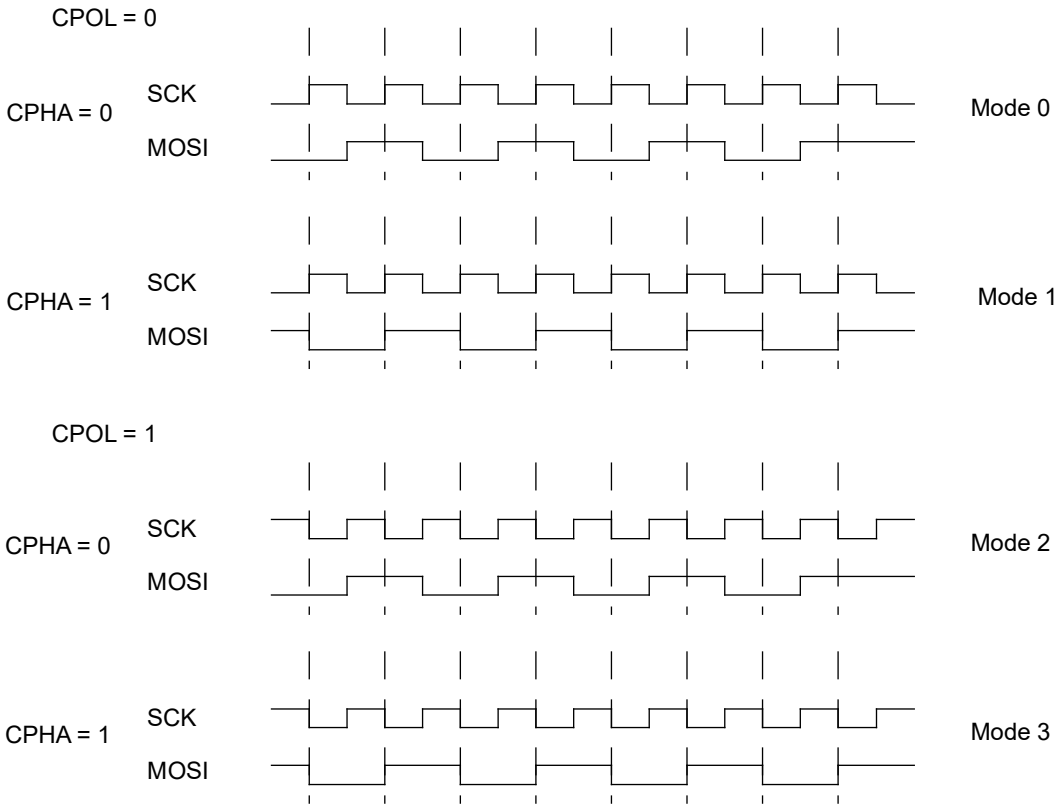


Figure 100. SPI transfer of byte 0x55 in all modes

44.2.2 3-wire transmission protocol

The core can be configured to operate in 3-wire mode, if the TWEN field in the core's Capability register is set to '1', where the controller uses a bidirectional dataline instead of separate data lines for input and output data. In 3-wire mode the bus is thus a half-duplex synchronous serial bus. Transmission starts when a master selects a slave through the slave's Slave Select (SLVSEL) signal and the clock line SCK transitions from its idle state. Only the Master-Output-Slave-Input (MOSI) signal is used for data transfer in 3-wire mode. The MISO signal is not used.

The direction of the first data transfer is determined by the value of the 3-wire Transfer Order (TTO) field in the core's Mode register. If TTO is '0', data is first transferred from the master (through the MOSI signal). After a word has been transferred, the slave uses the same data line to transfer a word back to the master. If TTO is '1' data is first transferred from the slave to the master. After a word has been transferred, the master uses the MOSI line to transfer a word back to the slave.

The data line transitions depending on the clock polarity and clock phase in the same manner as in SPI mode. The aforementioned slave delay of the MISO signal in SPI mode will affect the MOSI signal in 3-wire mode, when the core operates as a slave.

44.2.3 Receive and transmit queues

The core's transmit queue consists of the transmit register and the transmit FIFO. The receive queue consists of the receive register and the receive FIFO. The total number of words that can exist in each queue is thus the FIFO depth plus one. When the core has one or more free slots in the transmit queue it will assert the Not full (NF) bit in the event register. Software may only write to the transmit register when this bit is asserted. When the core has received a word, as defined by word length (LEN) in the Mode register, it will place the data in the receive queue. When the receive queue has one or more elements stored the Event register bit Not empty (NE) will be asserted. The receive register will only contain valid data if the Not empty bit is asserted and software should not access the receive register

unless this bit is set. If the receive queue is full and the core receives a new word, an overrun condition will occur. The received data will be discarded and the Overrun (OV) bit in the Event register will be set.

The core will also detect underrun conditions. An underrun condition occurs when the core is selected, via SPISEL, and the SCK clock transitions while the transmit queue is empty. In this scenario the core will respond with all bits set to '1' and set the Underrun (UN) bit in the Event register. An underrun condition will never occur in master mode. When the master has an empty transmit queue the bus will go into an idle state.

44.2.4 Clock generation

The core only generates the clock in master mode, the generated frequency depends on the system clock frequency and the Mode register fields DIV16, FACT, and PM. Without DIV16 the SCK frequency is:

$$SCKFrequency = \frac{AMBAclockfrequency}{(4 - (2 \cdot FACT)) \cdot (PM + 1)}$$

With DIV16 enabled the frequency of SCK is derived through:

$$SCKFrequency = \frac{AMBAclockfrequency}{16 \cdot (4 - (2 \cdot FACT)) \cdot (PM + 1)}$$

Note that the fields of the Mode register, which includes DIV16, FACT and PM, should not be changed when the core is enabled. If the FACT field is set to 0 the core's register interface is compatible with the register interface found in MPC83xx SoCs. If the FACT field is set to 1, the core can generate an SCK clock with higher frequency.

44.2.5 Slave operation

When the core is configured for slave operation it does not drive any SPI signal until the core is selected, via the SPISEL input, by a master. If the core operates in SPI mode when SPISEL goes low the core configures MISO as an output and drives the value of the first bit scheduled for transfer. If the core is configured into 3-wire mode the core will first listen to the MOSI line and when a word has been transferred drive the response on the MOSI line. If the core is selected when the transmit queue is empty it will transfer a word with all bits set to '1' and the core will report an underflow.

Since the core synchronizes the incoming clock it will not react to transitions on SCK until two system clock cycles have passed. This leads to a delay of three system clock cycles when the data output line should change as the result of a SCK transition. This constrains the maximum input SCK frequency of the slave to (system clock) / 8 or less. The controlling master must also allow the decreased setup time on the slave data out line.

The core can also filter the SCK input. The value of the PM field in the Mode register defines for how many system clock cycles the SCK input must be stable before the core accepts the new value. If the PM field is set to zero, then the maximum SCK frequency of the slave is, as stated above, (system clock) / 8 or less. For each increment of the PM field the clock period of SCK must be prolonged by two times the system clock period as the core will require longer time discover and respond to SCK transitions.

44.2.6 Master operation

When the core is configured for master operation it will transmit a word when there is data available in the transmit queue. When the transmit queue is empty the core will drive SCK to its idle state. If the

SPISEL input goes low during master operation the core will abort any active transmission and the Multiple-master error (MME) bit will be asserted in the Event register. If a Multiple-master error occurs the core will be disabled. Note that the core will react to changes on SPISEL even if the core is operating in loop mode and that the core can be configured to ignore SPISEL by setting the IGSEL field in the Mode register.

44.3 Registers

The core is programmed through registers mapped into APB address space.

*Table 597.*SPI controller registers

APB address offset	Register
0x00	Capability register
0x04-0x1C	Reserved
0x20	Mode register
0x24	Event register
0x28	Mask register
0x2C	Command register
0x30	Transmit register
0x34	Receive register
0x38	Slave Select register
0x3C	Automatic slave select register

44.3.1 SPI Controller Capability Register

*Table 598.*0x00 - CAP - SPI controller Capability register

31	24	23	20	19	18	17	16
SSSZ		MAXWLEN		TWEN	R	ASELA	SSEN
4		0x0		1	0	1	1
r		r		r	r	r	r

15	8	7	6	5	4	0
FDEPTH		SR	FT		REV	
0x10		0	0x0		5	
r		r	r		r	

31 : 24 Slave Select register size (SSSZ) - Number of slave select signals supported.

23 : 20 Maximum word Length (MAXWLEN) - The maximum word length supported is 32-bits

19 Three-wire mode Enable (TWEN)

18 Reserved

17 Automatic slave select available (ASELA)

16 Slave Select Enable (SSEN) - Multiple slave selects

15 : 8 FIFO depth (FDEPTH) - This field contains the depth of the core's internal FIFOs.

7 SYNCRAM (SR) - Core has buffers implemented with SYNCRAM components.

6 : 5 Fault-tolerance (FT) - Not used

4 : 0 Core revision (REV) - This manual applies to core revision 5.

44.3.2 SPI Controller Mode Register

*Table 599.*0x20 - MODE - SPI controller Mode register

31	30	29	28	27	26	25	24	23	20	19	16
R	LOOP	CPOL	CPHA	DIV16	REV	MS	EN	LEN		PM	
0	0	0	0	0	0	0	0	0		0	
r	rw	rw	rw	rw	rw	rw	rw	rw		rw	

15	14	13	12	11	7	6	5	4	3	2	1	0
TWEN	ASEL	FACT	OD	CG		ASELDEL		TAC	TTO	IGSEL	CITE	R
0	0	0	0	0		0		0	0	0	*	0
rw*	rw*	rw	rw*	rw		rw*		rw	rw	rw	rw	r

31 Reserved

30 Loop mode (LOOP) - When this bit is set, and the core is enabled, the core's transmitter and receiver are interconnected and the core will operate in loopback mode. The core will still detect, and will be disabled, on Multiple-master errors.

29 Clock polarity (CPOL) - Determines the polarity (idle state) of the SCK clock.

28 Clock phase (CPHA) - When CPHA is '0' data will be read on the first transition of SCK. When CPHA is '1' data will be read on the second transition of SCK.

27 Divide by 16 (DIV16) - Divide system clock by 16, see description of PM field below and see section 44.2.4 on clock generation. This bit has no significance in slave mode.

26 Reverse data (REV) - When this bit is '0' data is transmitted LSB first, when this bit is '1' data is transmitted MSB first. This bit affects the layout of the transmit and receive registers.

25 Master/Slave (MS) - When this bit is set to '1' the core will act as a master, when this bit is set to '0' the core will operate in slave mode.

24 Enable core (EN) - When this bit is set to '1' the core is enabled. No fields in the mode register should be changed while the core is enabled. This can bit can be set to '0' by software, or by the core if a multiple-master error occurs.

Table 599. 0x20 - MODE - SPI controller Mode register

23 : 20	Word length (LEN) - The value of this field determines the length in bits of a transfer on the SPI bus. Values are interpreted as: 0b0000 - 32-bit word length 0b0001-0b0010 - Illegal values 0b0011-0b1111 - Word length is LEN+1, allows words of length 4-16 bits. The value of this field must never specify a word length that is greater than the maximum allowed word length specified by the MAXWLEN field in the Capability register.
19 : 16	Prescale modulus (PM) - This value is used in master mode to divide the system clock and generate the SPI SCK clock. The value in this field depends on the value of the FACT bit. If bit 13 (FACT) is '0':The system clock is divided by 4*(PM+1) if the DIV16 field is '0' and 16*4*(PM+1) if the DIV16 field is set to '1'. The highest SCK frequency is attained when PM is set to 0b0000 and DIV16 to '0', this configuration will give a SCK frequency that is (system clock)/4. With this setting the core is compatible with the SPI register interface found in MPC83xx SoCs. If bit 13 (FACT) is '1': The system clock is divided by 2*(PM+1) if the DIV16 field is '0' and 16*2*(PM+1) if the DIV16 field is set to '1'. The highest SCK frequency is attained when PM is set to 0b0000 and DIV16 to '0', this configuration will give a SCK frequency that is (system clock)/2. In slave mode the value of this field defines the number of system clock cycles that the SCK input must be stable for the core to accept the state of the signal. See section 44.2.5.
15	Three-wire mode (TW) - If this bit is set to '1' the core will operate in 3-wire mode. This bit can only be set if the TWEN field of the Capability register is set to '1'.
14	Automatic slave select (ASEL) - If this bit is set to '1' the core will swap the contents in the Slave select register with the contents of the Automatic slave select register when a transfer is started and the core is in master mode. When the transmit queue is empty, the slave select register will be swapped back. Note that if the core is disabled (by writing to the core enable bit or due to a multiple-master-error (MME)) when a transfer is in progress, the registers may still be swapped when the core goes idle. This bit can only be set if the ASELA field of the Capability register is set to '1'. Also see the ASELDEL field which can be set to insert a delay between the slave select register swap and the start of a transfer.
13	PM factor (FACT) - If this bit is 1 the core's register interface is no longer compatible with the MPC83xx register interface. The value of this bit affects how the PM field is utilized to scale the SPI clock. See the description of the PM field.
12	Open drain mode (OD) - If this bit is set to '0', all pins are configured for operation in normal mode. If this bit is set to '1' all pins are set to open drain mode. The implementation of the core may or may not support open drain mode. If this bit can be set to '1' by writing to this location, the core supports open drain mode. The pins driven from the slave select register are not affected by the value of this bit.
11 : 7	Clock gap (CG) - The value of this field is only significant in master mode. The core will insert CG SCK clock cycles between each consecutive word. This only applies when the transmit queue is kept non-empty. After the last word of the transmit queue has been sent the core will go into an idle state and will continue to transmit data as soon as a new word is written to the transmit register, regardless of the value in CG. A value of 0b00000 in this field enables back-to-back transfers.
6 : 5	Automatic Slave Select Delay (ASELDEL) - If the core is configured to use automatic slave select (ASEL field set to '1') the core will insert a delay corresponding to ASELDEL*(SPI SCK cycle time)/2 between the swap of the slave select registers and the first toggle of the SCK clock. As an example, if this field is set to "10" the core will insert a delay corresponding to one SCK cycle between assigning the Automatic slave select register to the Slave select register and toggling SCK for the first time in the transfer. This field can only be set if the ASELA field of the Capability register is set to '1'.
4	Toggle Automatic slave select during Clock Gap (TAC) - If this bit is set, and the ASEL field is set, the core will perform the swap of the slave select registers at the start and end of each clock gap. The clock gap is defined by the CG field and must be set to a value >= 2 if this field is set. This field can only be set if the ASELA field of the Capability register is set to '1'.
3	3-wire Transfer Order (TTO) - This bit controls if the master or slave transmits a word first in 3-wire mode.If this bit is '0', data is first transferred from the master to the slave. If this bit is '1', data is first transferred from the slave to the master. This bit can only be set if the TWEN field of the Capability register is set to '1'.
2	Ignore SPISEL input (IGSEL) - If this bit is set to '1' then the core will ignore the value of the SPISEL input.

*Table 599.*0x20 - MODE - SPI controller Mode register

1	Require Clock Idle for Transfer End (CITE) - If this bit is ‘0’ the core will regard the transfer of a word as completed when the last bit has been sampled. If this bit is set to ‘1’ the core will wait until it has set the SCK clock to its idle level (see CI field) before regarding a transfer as completed. This setting only affects the behavior of the TIP status bit, and automatic slave select toggling at the end of a transfer, when the clock phase (CP field) is ‘0’.
0	RESERVED (R) - Read as zero and should be written as zero to ensure forward compatibility.

44.3.3 SPI Controller Event Register

*Table 600.*0x24 - EVENT - SPI controller Event register

31	30 16	15	14	13	12	11	10	9	8	7 0
TIP	R	AT	LT	R	OV	UN	MME	NE	NF	R
0	0	0	0	0	0	0	0	0	1	0
r	r	r	wc	r	wc	wc	wc	r	r	r

31 Transfer in progress (TIP) - This bit is ‘1’ when the core has a transfer in progress. Writes have no effect. This bit is set when the core starts a transfer and is reset to ‘0’ once the core considers the transfer to be finished. Behavior affected by setting of CITE field in Mode register.

30 : 16 RESERVED (R) - Read as zero and should be written to zero to ensure forward compatibility.

15 Automated transfers (AT) - Not used

14 Last character (LT) - This bit is set when a transfer completes if the transmit queue is empty and the LST bit in the Command register has been written. This bit is cleared by writing ‘1’, writes of ‘0’ have no effect.

13 RESERVED (R) - Read as zero and should be written to zero to ensure forward compatibility.

12 Overrun (OV) - This bit gets set when the receive queue is full and the core receives new data. The core continues communicating over the SPI bus but discards the new data. This bit is cleared by writing ‘1’, writes of ‘0’ have no effect.

11 Underrun (UN) - This bit is only set when the core is operating in slave mode. The bit is set if the core’s transmit queue is empty when a master initiates a transfer. When this happens the core will respond with a word where all bits are set to ‘1’. This bit is cleared by writing ‘1’, writes of ‘0’ have no effect.

10 Multiple-master error (MME) - This bit is set when the core is operating in master mode and the SPISEL input goes active. In addition to setting this bit the core will be disabled. This bit is cleared by writing ‘1’, writes of ‘0’ have no effect.

9 Not empty (NE) - This bit is set when the receive queue contains one or more elements. It is cleared automatically by the core, writes have no effect.

8 Not full (NF) - This bit is set when the transmit queue has room for one or more words. It is cleared automatically by the core when the queue is full, writes have no effect.

7 : 0 RESERVED (R) - Read as zero and should be written to zero to ensure forward compatibility.

44.3.4 SPI Controller Mask Register

*Table 601.*0x28 - MASK - SPI controller Mask register

31	30 16	15	14	13	12	11	10	9	8	7 0
TIPE	R	AT	LTE	R	OVE	UNE	MMEE	NEE	NFE	R
0	0	0	0	0	0	0	0	0	0	0
rw	r	rw	rw	rw	rw	rw	rw	rw	rw	r

31 Transfer in progress enable (TIPE) - When this bit is set the core will generate an interrupt when the TIP bit in the Event register transitions from ‘0’ to ‘1’.

30 : 16 RESERVED (R) - Read as zero and should be written to zero to ensure forward compatibility.

15 Automated transfers (AT) - Not used and should be always be set to ’0’

14 Last character enable (LTE) - When this bit is set the core will generate an interrupt when the LT bit in the Event register transitions from ‘0’ to ‘1’.

13 RESERVED (R) - Read as zero and should be written to zero to ensure forward compatibility.

12 Overrun enable (OVE) - When this bit is set the core will generate an interrupt when the OV bit in the Event register transitions from ‘0’ to ‘1’.

11 Underrun enable (UNE) - When this bit is set the core will generate an interrupt when the UN bit in the Event register transitions from ‘0’ to ‘1’.

10 Multiple-master error enable (MMEE) - When this bit is set the core will generate an interrupt when the MME bit in the Event register transitions from ‘0’ to ‘1’.

9 Not empty enable (NEE) - When this bit is set the core will generate an interrupt when the NE bit in the Event register transitions from ‘0’ to ‘1’.

8 Not full enable (NFE) - When this bit is set the core will generate an interrupt when the NF bit in the Event register transitions from ‘0’ to ‘1’.

7 : 0 RESERVED (R) - Read as zero and should be written to zero to ensure forward compatibility.

44.3.5 SPI Controller Command Register

*Table 602.*0x2C - CMD - SPI controller Command register

31 23	22	21 0
R	LST	R
0	0	0
r	rw	r

31 : 23 RESERVED (R) - Read as zero and should be written to zero to ensure forward compatibility.

22 Last (LST) - After this bit has been written to ‘1’ the core will set the Event register bit LT when a character has been transmitted and the transmit queue is empty. If the core is operating in 3-wire mode the Event register bit is set when the whole transfer has completed. This bit is automatically cleared when the Event register bit has been set and is always read as zero.

21 : 0 RESERVED (R) - Read as zero and should be written to zero to ensure forward compatibility.

44.3.6 SPI Controller Transmit Register

*Table 603.*0x30 - TX - SPI controller Transmit register

31 0
TDATA
0
w

31 : 0 Transmit data (TDATA) - Writing a word into this register places the word in the transmit queue. This register will only react to writes if the Not full (NF) bit in the Event register is set.

44.3.7 SPI Controller Receive Register

*Table 604.*0x34 - RXC - SPI controller Receive register

31	0
RDATA	
0	
r	

31 : 0 Receive data (RDATA) - This register contains valid receive data when the Not empty (NE) bit of the Event register is set. The placement of the received word depends on the Mode register fields LEN and REV:

For LEN = 0b0000 - The data is placed with its MSb in bit 31 and its LSb in bit 0.

For other lengths and REV = '0' - The data is placed with its MSB in bit 15.

For other lengths and REV = '1' - The data is placed with its LSB in bit 16.

To illustrate this, a transfer of a word with eight bits (LEN = 7) that are all set to one will have the following placement:

REV = '0' - 0x0000FF00

REV = '1' - 0x00FF0000

44.3.8 SPI Slave Select Register

*Table 605.*0x38 - SLVSEL - SPI Slave select register (optional)

31	4	3	0
R		SLVSEL	
0		0xF	
r		rw	

31 : 4 RESERVED (R) - Not used

3 : 0 Slave select (SLVSEL) - Slave select signals are mapped to this register on bits 3:0. Software is solely responsible for activating the correct slave select signals, the core does not assert or deassert any slave select signal automatically.

44.3.9 SPI Controller Automatic Slave Select Register

*Table 606.*0x3C - ASLVSEL - SPI controller Automatic slave select register

31	4	3	0
R		ASLVSEL	
0		0	
r		rw	

31 : 4 RESERVED (R)

3 : 0 Automatic Slave select (ASLVSEL) - If SSEN and ASELA in the Capability register are both '1' the core's slave select signals are assigned from this register when the core is about to perform a transfer and the ASEL field in the Mode register is set to '1'. After a transfer has been completed the core's slave select signals are assigned the original value in the slave select register.

45 SPI for Space Slave Controller

The GR716B microcontroller comprises an SPI for Space Slave controller (SPI4S). The SPI for Space Slave controller controls its own external pins and has a unique AMBA address described in chapter 2.10. The nominal SPI for Space Slave interface is connected via LVDS transceivers to external pins and the redundant interface is connected to external pins via the IOMUX, IO matrix available at table 7.

The control and status register are located on APB bus in the address range from 0x8040F000 to 0x8040F0FF. See SPI for Space Slave controller connections in the next drawing. The figure shows memory locations and functions used for SPI4S configuration and control.

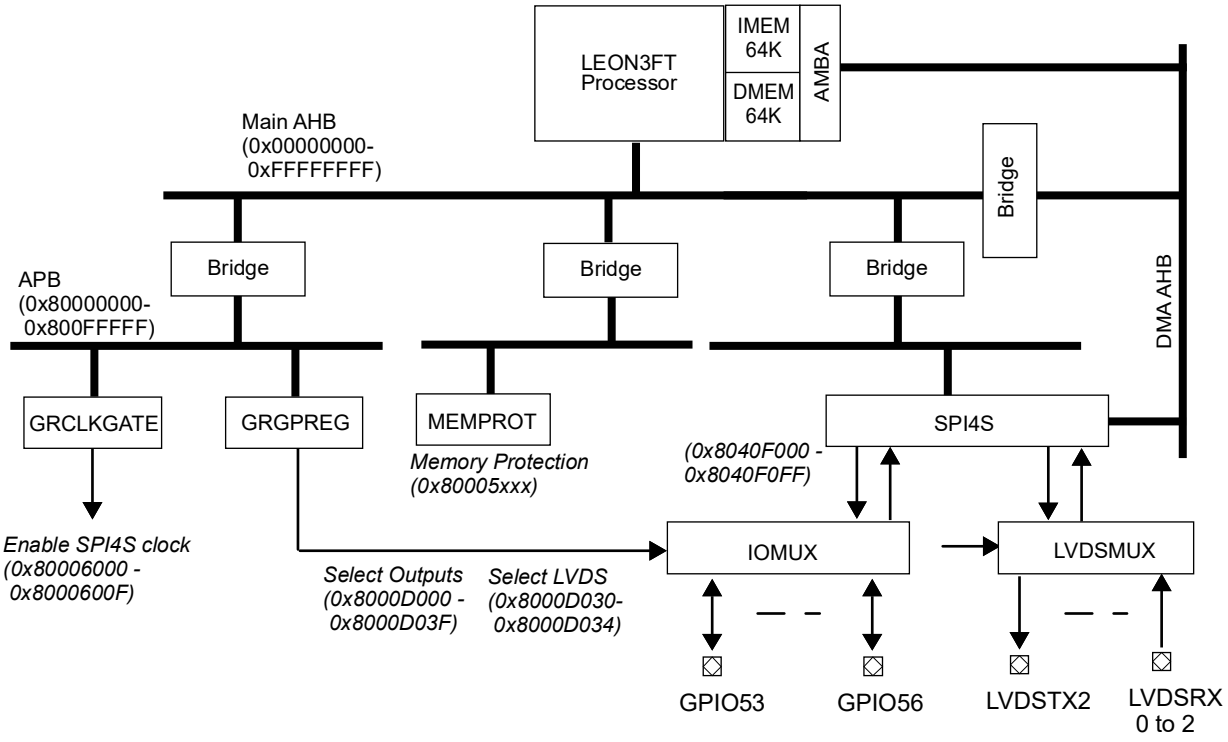


Figure 101. GR716B SPI4S bus and pin connection

The primary clock gating unit **GRCLKGATE** described in section 27 is used to enable/disable the SPI for Space Slave controller. The unit **GRCLKGATE** can also be used to perform reset of the SPI for Space Slave controller. Software must enable clock and release reset described in section 27 before configuration and transmission can start.

External IO selection and configuration is made in the system IO and LVDS configuration registers (**GRGPREG**) in the address range from 0x8000D000 to 0x8000D03F. See section 7.1 for further information.

The system can be configured to protect and restrict access to the SPI for Space Slave controller in the **MEMPROT** unit. See section 47 for more information.

45.1 Overview

This core is a Dual Port SPI Slave device that provides link between SPI and AMBA AHB and APB ports. Core features include configurable word length (4, 5, 6 ... 32 bits), bit ordering and all four SPI modes are supported. This core also has redundant SPI ports which can be interfaced using two differ-

ent masters. The slave takes two sets of SPI interfaces (nominal and redundant each consists of two data signals, one clock signal and one chip select signal).

45.2 Implementation of SPI protocols

In order to support the SPI 0 protocol the slave provides configurable word length of 4, 5, 6 ... 32 bits transmission and reception. The Word bit ordering can be MSB first or LSB first transferred.

For SPI 1 protocol the word length of the transfer can be 8, 16 or 24 bits. The word bit ordering MSB transferred first and LSB transferred last is supported. The parity bit can be appended at the end of every word, the parity bit is not included by the SPI slave device, since the implementation supports 9, 17 and 25 bits of word transfer the parity bit can be appended by the software.

All control and data transfer for SPI protocol 0 and 1 are supported only through the APB registers.

The SPI protocol 2 uses a fixed word length of 16 bits. The word bit ordering is MSB transferred first and LSB transferred last. Also this core implements the network layer of the SPI protocol 2, the slave hardware itself can process the SPI protocol 2 commands and provide responses. The APB interface is only for control and status, all the data transfer to the AMBA is performed using the AHB Master.

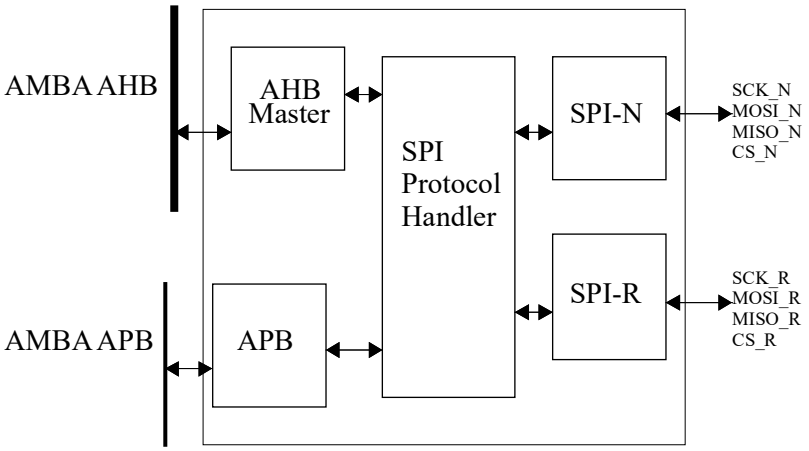


Figure 102. Block diagram

45.3 Transmission

The SPI bus is a full-duplex synchronous serial bus. Transmission starts when a master selects a slave through the slave's Slave Select (CS) signal and the clock line SCK transitions from its idle state. Data is transferred from the master through the Master-Output-Slave-Input (MOSI) signal and from the slave through the Master-Input-Slave-Output (MISO) signal. In some systems with only one master and one slave, the Slave Select input of the slave may be always active and the master does not need to have a slave select output. This does not apply to this device, the slave select signal must be used to mark the start and end of an operation.

During a transmission on the SPI bus data is either changed or read at a transition of SCK. If data has been read at edge n, data is changed at edge n+1. If data is read at the first transition of SCK the bus is said to have clock phase 0, and if data is changed at the first transition of SCK the bus has clock phase 1. The idle state of SCK may be either high or low. If the idle state of SCK is low, the bus has clock polarity 0 and if the idle state is high the clock polarity is 1. The combined values of clock polarity (CPOL) and clock phase (CPHA) determine the mode of the SPI bus. Figure below shows one byte (0x55) being transferred MSb first over the SPI bus under the four different modes. Note that the idle state of the MOSI line is '1' and that CPHA = 0 means that the devices must have data ready before

the first transition of SCK. The figure does not include the MISO signal, the behavior of this line is the same as for the MOSI signal.

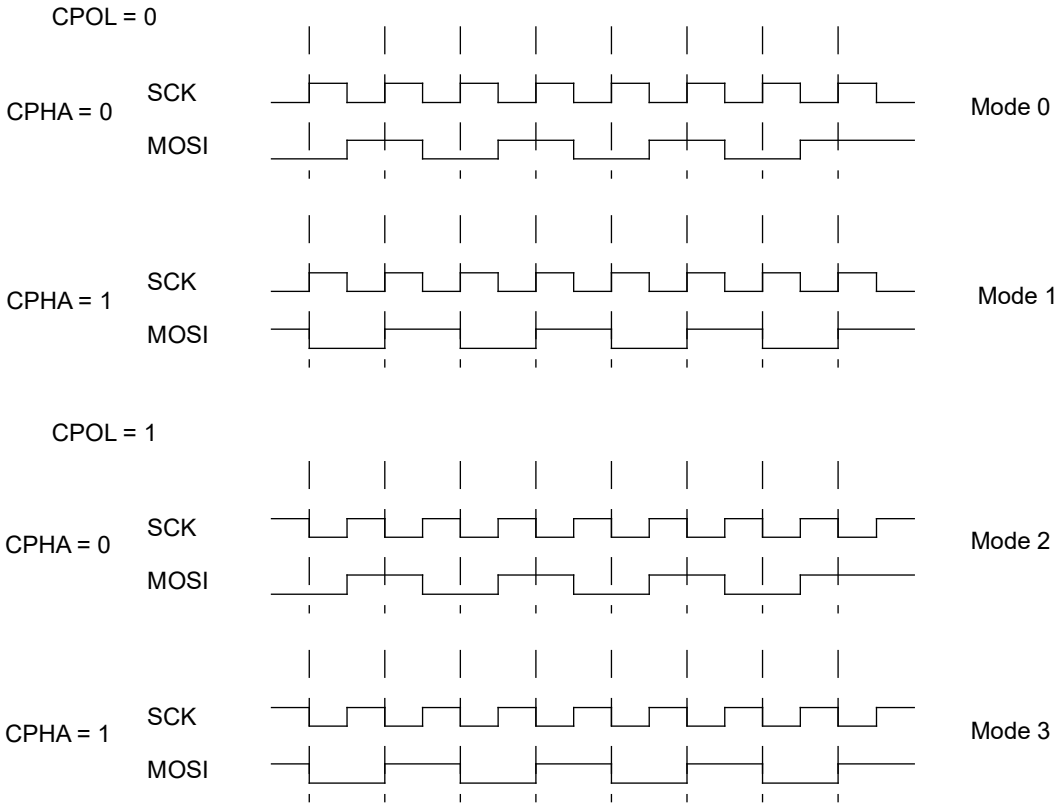


Figure 103. SPI transfer of byte 0x55 in all modes

45.4 Operation

The data transfer between the master and the slave is through APB registers or through command transfer from a master is determined by the EN bit in the SPI2 control register. When APB registers are used the data transferred by a master is available at receive registers (NRDATA or RRDATA depending on the port used) while during the same reception period the contents of the transmit registers (TDATA) are transferred to the master. When appropriate commands are transferred by a master SPI device and EN bit in the SPI2 control register is enabled then the commands are processed by the SPI 2 protocol handler available in this core. The SPI protocol 2 implementation is explained in detail in the following section.

45.5 SPI 2 Protocol Handler

The core is capable of handling the commands (based on SPI protocol 2) transferred by a SPI master and provide response. The message format transferred between a SPI master and SPI slave device is defined below.

Table 607. Example message format (write data)

Signal	Message Header		Payload	Payload CRC
MOSI	Command #1	Command #2	Data	CRC-16
MISO	Response #1	Response #2	0x0000	0x0000

Table 608. Example message format (read data)

Signal	Message Header		Payload	Payload CRC
MOSI	Command #1	Command #2	0x0000	0x0000
MISO	Response #1	Response #2	Data	CRC-16

The message header is composed of a Command token from the master and a Response token from the slave. The message also contains optional data words and CRC checksum appended at the end that are calculated for the data words transferred. The CRC is mandatory, if the message contains payload data then the message is always appended with one word of CRC. The received messages are processed by the SPI slave device and response and data are transferred as per the received command. Also note some of the status bits in the response token are status for the previously received command.

The SPI slave has the possibility to address incoming data with clock gaps (splitted) fashion. If the SPI master transfers the data with a clock gap the slave can accept the data and provide proper response. For example if the SPI master is software controlled and has a SPI controller with a word-width that is less than the full message then software needs to keep at least one word in the transmit queue at all times to avoid breaking the protocol. In order to provide a relaxed requirement on software, the SPI2 protocol allow gaps between clock periods (equivalent to stretching a clock period). The gaps must be at word level (16 bits) i.e. the clock gap can be between Command #1 and Command #2 not within the Command #1 16 bits.

45.6 Message Header - Command Token

The master transmits a message header that specifies the action need to be performed in slave. The command token sent by the master device consist of two 16 bit words. The message header content details are explained below.

Table 609. Command word 1

MSB	**Command Token Word #1**														**LSB**
Prefix		**Command Code**						**Spare**		**Message Length**					
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
’0’	’1’	C5	C4	C3	C2	C1	C0	’1’	’1’	L5	L4	L3	L2	L1	L0

Table 610. Command word 2

MSB	**Command Token Word #1**														**LSB**
Prefix		**Sub-Address**								**Spare**		**CRC-4**			
31	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
’0’	’1’	SA7	SA6	SA5	SA4	SA3	SA2	SA1	SA0	’1’	’1’	C3	C2	C1	C0

Prefix and spare

The prefix bits are transmitted initially. In the command word#1 and #2 the prefix and spare bits have fixed value in the current implementation, these are reserved bits. The spislave receives them and use it for validating the token. If an invalid prefix and spare bits are received then Status illegal command (SIC) status bit is enabled and also transmitted to master as part of the next response token.

Message Length

The number of payload words that will be transmitted in the current message. The number should not include the command token and the CRC checksum appended at the end of the message.

Sub-address

This field provide additional sub-address location for write and read commands.

CRC-4

The final four bits of the command token consist of a checksum for all the previous command token bits transmitted in this message. The CRC-4 should be computed for the following 28 bits, Command word #1 (16bits, MSB first sent to the CRC generator) and Command word #2 (excluding this CRC-4 field) (12bits, MSB first sent to the CRC generator). The Prefix and Spare fields are included in the CRC calculation. The generator polynomial used is X4 + X + 1. In this SPI slave receiving end the CRC-4 is calculated internally for the received command token, if the calculated CRC-4 does not match the expected value (this field) the corresponding command token is discarded and message error status is enabled and transmitted to master as part of the next response token.

Payload data

The payload consist of the data need to be transferred from the master to slave. Depending on the command executed the master must include valid data or dummy information in the form of string of zeros. For example the write command have the data to be written as the payload but the read command have dummy information in the form of string of zeros.

Payload CRC

When a valid payload is delivered in the payload data section of the message a Payload CRC (CRC-16) must be included at the end of the message. The generator polynomial used is x16+ x15+ x2+ 1. When dummy information in the form of string of zeros is included then no Payload CRC need to be attached then the payload CRC field must be all zero (0X0000). In the SPI slave receiving end the CRC-16 is calculated internally for the received payload data, if the calculated CRC-16 does not match the expected value (this field) the corresponding operation with respect to the command is not performed and message error status is enabled and transmitted to master as part of the next response token.

45.6.1 Command Code

The command code specify the operating instruction for the receiving slave. The detailed explanation of each command code and its implementation are explained in the table below.

Table 611. Reset Command

Code	Command	Length	Sub-Address	Payload	Description
0x00	RESET_SPI	0x00	0x00	None	This command will reset all the spi slave device registers to the default value except the time registers (TIME1, TIME2) and core enable registers (ENN and ENR)

The RESET_SPI command resets the SPI slave device to a power up initialized state. The SPI slave resets the system only if it received a valid command. If the prefix and spare bits does not match or if the calculated CRC-4 does not match the expected value then the RESET_SPI command is discarded.

Time synchronization command

Table 612. Time synchronization command

Code	Command	Length	Sub-Address	Payload	Description
0x07	SYNCH	0x04	0x00	MOSI: <SYNC1> <SYNC2> <SYNC3> <SYNC4> <CRC-16> MISO: <all zeros>	The master must transmit the SYNC command token followed by payload words containing synchronization information for it. These words are copied inside dedicated registers implemented in the SPI slave device after validation.

The time register is of 64 bit in width, the most significant time is transferred first in SYNC1 followed by SYNC2, SYNC3 and SYNC4. The time register roll over when its maximum count is reached. The time is synchronised only when all the words are received and also the command token CRC-4 and data CRC-16 must be valid. All bits are zero at reset. The RESET_SPI does not reset the time register.

Table 613. Time increment command

Code	Command	Length	Sub-Address	Payload	Description
0x08	TICK	0x00	Used as index for increment.	None	This command is used to advance the timing synchronization register available in the SPI slave device (same register used for the SYNC command)

A valid command increments the implemented time register. The sub address field specify from which bit the time register must increment.

Table 614. Read back sent command

Code	Command	Length	Sub-Address	Payload	Description
0x0A	READBACK	0x02	0x00	MOSI: <all zeros> MISO: <CMDTOKEN> < CRC-16>	The command can be used to verify the correct reception of the previous command. Upon reception of the command the slave respond with the previous command token

The SPI slave device after receiving the READBACK_CMD send the previous command token transmitted by the SPI master. This command is useful only when some other command (other than

RESET_SPI) was previously transmitted. If the previous command is RESET_SPI, the SPI master only receives zeros in the payload section of this command.

Table 615. Write Command

Code	Command	Length	Sub-Address	Payload	Description
0x0D	WRITE_SA	Number of words to be written, N	SA	MOSI: <DW1> <DW2> …<DWN> <CRC-16> MISO: <all zeros>	he command is used to write a certain number of data words into a slave specific Sub Address. Dedicated field of the command token select the payload length and the target SA.

When a valid WRITE_SA command is received the payload data is stored at the address specified. The address for writing the data is calculated by using the write address register (CONFIG_WRITE), and sub-address. The CRC-16 is calculated for received words and compared with the received payload CRC, if a data CRC error is detected then message error status is enabled and transmitted to master as part of the next response token.

Table 616. Read command

Code	Command	Length	Sub-Address	Payload	Description
0x0E	READ_SA	Number of words to be read, N	SA	MOSI: <all zeros> MISO: <DW1> <DW2> …<DWN> <CRC-16>	The command is used to read a certain number of data words into a slave specific Sub Address. Dedicated field of the command token select the payload length and the target SA

When a valid READ_SA command is received the payload data is transferred from the address specified. The address for reading the data is calculated by using the read address register (CONFIG_READ), and sub-address. The CRC-16 is calculated for transmitted words and sent as payload CRC by the slave device.

Table 617. Configure address commands

Code	Command	Length	Sub-Address	Payload	Description
0x20	CONFIG-WRITE_ADDR	0x02	0x00	MOSI: <CW1> <CW2> <CRC-16> MISO: <all zeros>	The command can be used to notify the slave about the address to which the data from the master is written, used for WRITE_SA command.
0x21	CONFIG READ_ADDR	0x02	0x00	MOSI: <CR1> <CR2> <CRC-16> MISO: <all zeros>	The command can be used to notify the slave about the address from which the data to the master is read, used for READ_SA command.

Dedicated registers for write address and read address is implemented, these registers takes value from this command. The purpose of this register is to access up to 32 bits of address space. The most significant word CW1 (or CR1) contains the most significant bytes of the target address.

Redundancy commands

Table 618. Redundancy commands

Code	Command	Length	Sub-Address	Payload	Description
0x24	ACTIVATE	0x00	0x00	None	The command is used to activate the other slave interface. This command cannot activate the interface in which it is receiving this command.
0x25	DEACTIVATE	0x00	0x00	None	The command is used to deactivate the other slave interface. This command cannot deactivate the interface in which it is receiving this command.

The SPI Slave device has two dedicated interfaces for two masters. The masters can send to its corresponding slave interface to activate or deactivate the other SPI interface. The master device cannot activate or deactivate the same ports on which it is connected, it can only activate or deactivate the other ports.

Initially both the SPI port interfaces are enabled to receive commands, when the communication between the nominal master and slave interface fails then the redundant master can deactivate the nominal interface using its dedicated redundant interface. The redundant master can also activate the nominal interface.

An example switchover scenario from nominal to redundant interface is described in the following text.

The nominal master communicates with its dedicated interface to a slave device, a fault occurred can be detected by the master using several options,

The status received by the master have invalid values (using the response token),

The Read back command sent does not provide appropriate values in the received payload,

Error bits are enabled in the status received by the master (using the response token),

Based on any of the above mentioned fault detection methods the master can send deactivate command in the redundant interface to deactivate the nominal interface of the slave. The master can send Read back sent commands (using redundant) to check if the previous deactivate command was received by the slave and can check the status of the response token as well. After conforming a proper communication has been established the master can use the redundant interface to perform its normal operations.

Any other commands which are not implemented is received then the command token is discarded and Status illegal command (SIC) bit is enabled and also transmitted to master as part of the next response token.

45.6.2 Message Header -Response Token

The slave transmits a message header which consist of status of module and details of error occurred. The message header sent by SPI slave device is called response token which consist of two 16 bit words. The message header content details are explained below..

Table 619. Response Token Word #1

MSB	Response Token Word #2														LSB
Prefix		Command Code					Spare					Module State			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
’0’	’1’	SFT	ME	AR	IC	’0’	’0’	’0’	’0’	’0’	’0’	MS3	MS2	MS1	MS0

Table 620. Response Token Word #3

MSB	Response Token Word #2														LSB
Prefix		DATA	Spare									CRC-4			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
’0’	’1’	SA7	’0’	’0’	’1’	’1’	’1’	’0’	’0’	’0’	’0’	C3	C2	C1	C0

Table 621. Response Status bit

Bit	Identifier	Type	Value	Description	Clear Condition	Comment
13	TERMINAL_FAULT	Error	’0’ = no fault ‘1’ = fault	The bit flag a SPI terminal fault condition.	According to the module current state.	In SPI slave device this bit is enabled or disabled by SPI2 control register (STF) using APB.
12	MESSAGE_ERROR	Error	’0’ = no fault ‘1’ = fault	This bit is utilized to indicate that the previous message received from the bus master has failed to pass the validity tests.	Always related to the previous command. Reception of a valid command will clear it (with a delay of one command).	This status bit is enabled when the received message fails to pass the command token and payload data CRC checks. The next valid command clears this status bit.
11	ADDRESS_ERROR	Error	’0’ = no fault ‘1’ = fault	This bit flag an AMBA error occurred while performing the previous command.	Always related to the previous command. Reception of a valid command will clear it (with a delay of one command)	The SPI slave device uses an AHB master to perform the memory read and write, this bit is enabled when an AHB error is reported.

Table 621. Response Status bit

Bit	Identifier	Type	Value	Description	Clear Condition	Comment
10	ILLEGAL_CMD	Error	’0’ = no fault ‘1’ = fault	This bit flag that the previous received command was not compatible with the SPI slave device.	Always related to the previous command. Reception of a valid command will clear it (with a delay of one command)	When the prefix and spare bits in the received command token do not match the intended value or an unimplemented command is received this status bit is enabled. The next valid command clears this status bit

Table 622. Response Module State bits

Bit	Description
3	In the SPI slave device these bits are enabled or disabled by SPI2 control register (MODSTAT) using APB. These bits can be used by Software controlling the slave device to provide additional status to the master.
2	
1	
0	

45.7 Redundancy

The SPI slave has a two SPI ports which can be interfaced using two different masters. Two SPI master capable of communicating individually to the respective port must be available inorder to achieve redundancy using this Dual-port SPI slave.The slave takes two sets of SPI interfaces (nominal and redundant). The configuration registers available in the device is used to enable which interface to communicate and it is possible to use dedicated commands (using SPI 2 protocol) to activate and deactivate ports.

While using configuration registers to activate or deactivate ports, the complete control of activation and deactivation must be performed by the external unit, only one port must active at any time. If both enabled then both the SPI ports are open for communication which is not supported while using external configuration for redundancy. The system which initiates the communication should take responsibility for which lane to take (there must be dedicated SPI Masters available in the system tocommunicate with the respective slave). If both are disabled then no communication is possible. The Master (driver) must have two dedicated SPI Master to perform communication on each lane of the SPI bus.

When commands are used to control the ports, the device can receive commands from both the interfaces. By receiving from both the interfaces the slave device can deactivate a non-working interface. The intention is to keep only one bus active for normal operation but using the redundant bus to achieve switchover. The SPI protocol 2 implementation supports dedicated commands to achieve the activation and deactivation of interfaces.

In normal working case the SPI masters Nominal and redundant (using HW or SW) should make sure not to write at the same time to both lanes of dual-port SPI Slave (for example to make a transfer). The SW or HW can command the Redundant master only when it detects problem with the Nominal communication. For worst case lets say, the SPI masters Nominal (in error state babelling some command repeatedly), using the redundant port the Nominal lane can be switched off (switch over command using redundant port or external configuration), the slave takes the redundant port input, the SPI slave is designed to take the redundant port inputs when it is available rather than nominal input.

45.8 Registers

The core is programmed through registers mapped into APB address space.

*Table 623.*APB registers

APB address offset	Register
0x00	Control register
0x04	Status register
0x08	Transmit register
0x0C	Nominal receive register
0x10	Redundant receive register
0x14	Interrupt enable register
0x18	Interrupt register
0x1C	Reserved
0x20	SPI2 control register
0x24	SPI2 time1 register
0x28	SPI2 time2 register
0x2C	SPI2 config address write register
0x30	SPI2 config address read register

45.8.1 Control Register

*Table 624.*0x00 - CTRL - Control register

31 24	23 13	12 8	7	6	5	4	3	2	1	0
Key	R	WLEN	IAMBA	CPHA	CPOL	REV	R	RESET	ENR	ENN
0	0	0x0F	0	0	0	1	0	0	1	1
w	r	rw	rw	rw	rw	rw	r	rw	rw	rw

31: 24	Safety code (KEY) - Must be 0x68 when writing, otherwise register write is ignored
23 : 17	RESERVED (R) - Read as zero and should be written to zero to ensure forward compatibility.
16	Overrun detect (OD) - To detect overrun condition (also to trigger overrun interrupt) this bit must be enabled. Valid only for SPI protocol 0 and 1.
15 : 13	RESERVED (R) - Read as zero and should be written to zero to ensure forward compatibility.
12 : 8	Word length (WLEN) - The value of this field determines the length in bits of a transfer on the SPI bus. Valid values are 0x03 to 0x1F Word length is WLEN+1, allows words of length 4-32 bits.
7	AMBA Interrupt enable (IAMBA) - If set, AMBA interrupt generation is enabled for the events that are individually maskable by the Interrupt enable (INTE) register
6	Clock phase (CPHA) - When CPHA is ‘0’ data will be read on the first transition of SCK. When CPHA is ‘1’ data will be read on the second transition of SCK.
5	Clock polarity (CPOL) - Determines the polarity (idle state) of the SCK clock.
4	Reverse data (REV) - When this bit is ‘0’ data is transmitted LSB first, when this bit is ‘1’ data is transmitted MSB first.
3	RESERVED (R) - Read as zero and should be written to zero to ensure forward compatibility.
2	Reset (RESET) - Resets all the registers in the core except time registers (TIME1, TIME2) and core enable registers (ENN and ENR).
1	Enable redundant port transfer (ENR) - Enable bit for redundant port transfer.
0	Enable nominal port transfer (ENN)- Enable bit for nominal port transfer.

45.8.2 Status Register

Table 625. 0x04 - STAT - Status register

31 8	7	6	5	4	3 2	1	0
RESERVED	ATR	ATN	SAR	SIC	R	RR	RN
0	0	1	0	0	0	0	0
r	r	r	r	r	r	r	r

31 : 3 RESERVED

7 Active transmission in redundant port (ATR) - This bit provides the status of the redundant transmission port. Set based on the incoming activate and deactivate commands (active '1' else '0'). Valid only for SPI protocol 2 implementation.

6 Active transmission in nominal port (ATN) - This bit provides the status of the nominal transmission port. Set based on the incoming activate and deactivate commands (active '1' else '0'). Valid only for SPI protocol 2 implementation.

5 Status address error (SAR) - This bit gets set to '1' when an AMBA write or read access resulted in a error. A valid new command clears this status bit. Valid only for SPI protocol 2 implementation.

4 Status illegal command (SIC) - This bit gets set to '1' when an illegal command is received. A valid new command clears this status bit. Valid only for SPI protocol 2 implementation.

3 : 2 RESERVED (R) - Read as zero and should be written to zero to ensure forward compatibility.

1 Received data redundant (RR) - This bit gets set to '1' each time a data is received in the redundant port. The bit gets set to '0' when the Redundant receive register is read.

0 Received data nominal (RN) - This bit gets set to '1' each time a data is received in the nominal port. The bit gets set to '0' when the Nominal receive register is read.

45.8.3 Transmit Register

Table 626. 0x08 - TDATA - Transmit register

31 0
TDATA
0
rw

31 : 0 Transmit data (TDATA) - The written data is transferred to the master device when appropriate conditions for CS and SCK are satisfied. The word to transmit should be written with its least significant bit at bit 0. Also note that the number of bits to be transferred from this register must match the word length register (WLEN). This setting is valid only for SPI protocols 0 and 1.

45.8.4 Nominal Receive Register

Table 627. 0x0C - NRDATA - Nominal receive register

31 0
NRDATA
0
r

31 : 0 Nominal Receive data (NRDATA) - This register contains received data from the nominal port. Valid only for SPI protocol 0 and 1.

45.8.5 Redundant Receive Register

Table 628. 0x10 - RRDATA - Redundant receive register

31 0
RRDATA
0

*Table 628.*0x10 - RRDATA - Redundant receive register

r

31 : 0 Redundant Receive data (RRDATA) - This register contains received data from the redundant port. Valid only for SPI protocol 0 and 1.

45.8.6 Interrupt Enable Register

*Table 629.*0x14 - INTE- Interrupt enable register

31 24	23 9	8	7	6	5	4	3	2	1	0
Key	RESERVED	OVRE	WDE	AE	CRE	CWE	TICKE	SYNCE	RXRE	RXNE
0	0	0	0	0	0	0	0	0	0	0
w	r	rw	rw	rw	rw	rw	rw	rw	rw	rw

31 : 24 Safety code (KEY) - Must be 0x68 when writing, otherwise register write is ignored.

23 : 9 RESERVED (R) - Read as zero and should be written to zero to ensure forward compatibility.

8 Overrun interrupt enable (OVRE) - If enabled an interrupt will be generated when overrun condition occurs for data reception. Valid only for SPI protocol 0 and 1.

7 Write data interrupt enable (WDE). Valid only for SPI protocol 2.

6 AMBA access error interrupt enable (AE). Valid only for SPI protocol 2.

5 Change in config read address interrupt enable (CRE). Valid only for SPI protocol 2.

4 Change in config write address interrupt enable (CWE). Valid only for SPI protocol 2.

3 Tick command received interrupt enable (TICKE). Valid only for SPI protocol 2.

2 Sync command received interrupt enable (SYNCE). Valid only for SPI protocol 2.

1 Data received in redundant port interrupt enable (RXRE).Valid only for SPI protocol 0 and 1.

0 Data received in nominal port interrupt enable (RXNE). Valid only for SPI protocol 0 and 1.

45.8.7 Interrupt Register

Table 630. 0x18- INT- Interrupt register

31 24	23 8	7	6	5	4	3	2	1	0
Key	RESERVED	WD	AI	CR	CW	TICK	SYNC	RXR	RXN
0	0	0	0	0	0	0	0	0	0
w	r	wc	wc	wc	wc	wc	wc	wc	wc

31 : 24	Safety code (KEY) - Must be 0x68 when writing, otherwise register write is ignored.
23 : 8	RESERVED (R) - Read as zero and should be written to zero to ensure forward compatibility.
7	Write data interrupt (WD). Valid only for SPI protocol 2.
6	AMBA access error interrupt (AI). Valid only for SPI protocol 2.
5	Change in config read address interrupt (CR). Valid only for SPI protocol 2.
4	Change in config write address interrupt (CW). Valid only for SPI protocol 2.
3	Tick command received interrupt (TICK). Valid only for SPI protocol 2.
2	Sync command received interrupt (SYNC). Valid only for SPI protocol 2.
1	Data received in redundant port interrupt (RXR).Valid only for SPI protocol 0 and 1.
0	Data received in nominal port interrupt (RXN). Valid only for SPI protocol 0 and 1.

45.8.8 SPI2 Control Register

Table 631. 0x20- SPI2C- SPI2 control register

31 24	23 8	7	6	5	4	3	2	1	0
Key	RESERVED	MODSTAT				RESERVED		STF	EN
0	0	0	0	0	0	0	0	0	1
w	r	rw	rw	rw	rw	r	r	rw	rw

31 : 24	Safety code (KEY) - Must be 0x68 when writing, otherwise register write is ignored.
23 : 8	RESERVED (R) - Read as zero and should be written to zero to ensure forward compatibility.
7: 4	Module state (MODSTAT). The values in these bits are sent to the master via the response token. These are user configurable registers which can be set to ‘1’ or ‘0’. Valid only for SPI protocol 2.
3: 2	RESERVED (R) - Read as zero and should be written to zero to ensure forward compatibility.
1	SPI terminal failure (STF). This value in this bit is sent to the master via the response token. In order to intimate a terminal failure this bit can be written to ‘1’ through software. Valid only for SPI protocol 2.
0	Enable (EN). SPI protocol 2 enable bit. If set to ‘1’ the commands received from master are handled by the SPI 2 protocol handler in the core. If set to ‘0’ the data received and transfered are using the APB registers.

45.8.9 SPI2 Time1 Register

Table 632. 0x24 - TIME1 - SPI2 time1 register

31 0
TIME1
0x00000000
r

31 : 0	Time 1 register (TIME1) - Provides the most significant 32 bits of the time register. This is a status (read only) register, the contents of this register is a reflection of the time modified/incremented using the sync and tick command respectively.

45.8.10 SPI2 Time2 Register

*Table 633.*0x28 - TIME2 - SPI2 time2 register

31	0
TIME2	
0x00000000	
r	

31 : 0 Time 2 register (TIME2) - Provides the lower 32 bits of the time register. This is a status (read only) register, the contents of this register is a reflection of the time modified/incremented using the sync and tick command respectively.

45.8.11 SPI2 Config Address Write Register

*Table 634.*0x2C - CONFW - SPI2 config address write register

31	0
CONFW	
0x40000000	
r	

31 : 0 Configuration write address (CONFW) - Defines the base address for the memory area where the core is allowed to make accesses. This is a status (read only) register, the contents of this register can be modified by the configuration write address command.

45.8.12 SPI2 Config Address Read Register

*Table 635.*0x30 - CONFR - SPI2 config address read register

31	0
CONFR	
0x40000000	
r	

31 : 0 Configuration read address (CONFR) - Defines the base address for the memory area where the core is allowed to make accesses. This is a status (read only) register, the contents of this register can be modified by the configuration read address command.

46 SPI Memory Controller

The GR716B microcontroller comprises 2 separate SPI memory controller units (SPIMCTRLx). Each SPI memory controller unit controls its own external pins and has a unique AMBA address described in chapter 2.10. SPI memory controller unit 0 (SPIMCTRL0) has dedicated external signals, while SPI memory controller unit 1 (SPIMCTRL1) has access to external signals via IO switch matrix described in section 2.5.

Each SPI memory controller unit control and status register are located on main AHB bus in the address range from 0xFFFF0000 to 0xFFFF02FFF. See SPIMCTRL units connections in the next drawing. The figure shows memory locations and functions used for SPIMCTRL configuration and control.

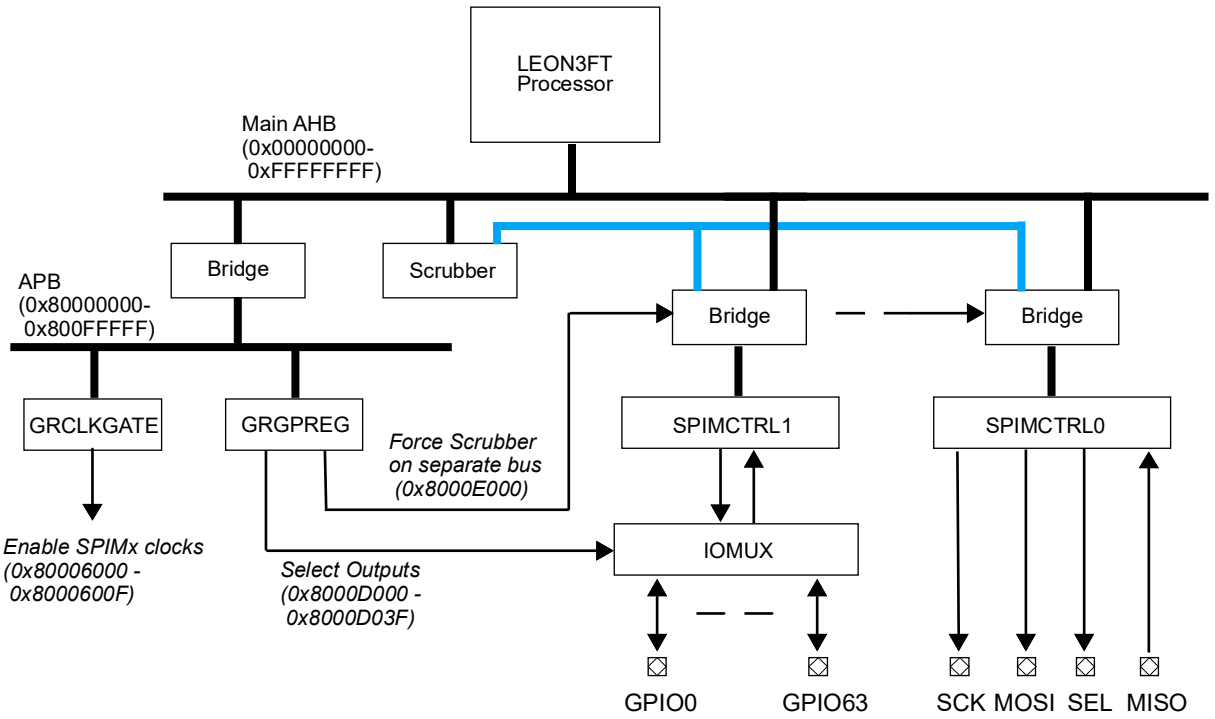


Figure 104. GR716B SPIMx bus and pin connection

The primary clock gating unit **GRCLKGATE** described in section 27 is used to enable/disable individual SPI memory controller units (SPIMCTRLx). The unit **GRCLKGATE** can also be used to perform reset of individual SPI memory controller units (SPIMCTRLx). Software must enable clock and release reset described in section 27 before SPI memory controller units (SPIMCTRLx) configuration and transfers can start.

External IO selection for SPI memory controller unit 1 (SPIMCTRL1) is made in the system IO configuration register (**GRGPREG**) in the address range from 0x8000D000 to 0x8000D03F. See section 7.1 for further information.

Each **SPIMCTRLx** unit controls its own external pins and has a unique AMBA address described in chapter 2.10. SPIMCTRL unit 0 and 1 has identical configuration and status registers. Configuration and status registers are described in this section 46.3

System can be configured to scrub memory contents of individual SPIMCTRL units in the **SCRUBBER** unit. See section 42 for more information.

46.1 Overview

The core maps a memory device connected via the Serial Peripheral Interface (SPI) into AMBA address space. Read accesses are performed by performing normal AMBA read operations in the mapped memory area. Other operations, such as writes, are performed by directly sending SPI commands to the memory device via the core's register interface. The core is highly configurable and supports most SPI Flash memory devices.

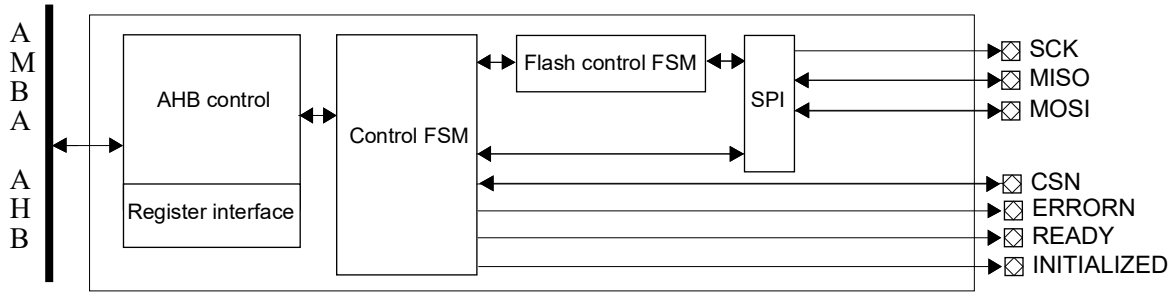


Figure 105. Block diagram

46.2 Operation

46.2.1 Operational model

The core has two memory areas that can be accessed via the AMBA bus; the I/O area and the ROM area. The ROM area maps the memory device into AMBA address space and the I/O area is utilized for status reporting and to issue user commands to the memory device.

When transmitting SPI commands directly to the device the ROM area should be left untouched. The core will issue an AMBA ERROR response if the ROM area is accessed when the core is busy performing an operation initiated via I/O registers.

Depending on the type of device attached the core may need to perform an initialization sequence. Accesses to the ROM area during the initialization sequence receive AMBA error responses. The core has successfully performed all necessary initialization when the Initialized bit in the core's status register is set.

46.2.2 I/O area

The I/O area contains registers that are used when issuing commands directly to the memory device. By default, the core operates in System mode where it will perform read operations on the memory device when the core's ROM area is accessed. Before attempting to issue commands directly to the memory device, the core must be put into User mode. This is done by setting the User Control (USRC) bit in the core's Control register. Care should be taken to not enter User mode while the core is busy, as indicated by the bits in the Status register. The core should also have performed a successful initialization sequence before User mode accesses (INIT bit in the Status register should be set).

Note that a memory device may need to be clocked when there has been a change in the state of the chip select signal. It is recommended that software transmits a byte with the memory device deselected after entering and before leaving User mode.

The following steps are performed to issue a command to the memory device after the core has been put into User mode:

1. Check Status register and verify that the BUSY and DONE bits are cleared. Also verify that the core is initialized and not in error mode.
2. Optionally enable DONE interrupt by setting the Control register bit IEN.
3. Write command to Transmit register.

4. Wait for interrupt (if enabled) or poll DONE bit in Status register.
5. When the DONE bit is set the core has transferred the command and will have new data available in the Receive register.
6. Clear the Status register's DONE bit by writing one to its position.

The core should not be brought out of User mode until the transfer completes. Accesses to ROM address space will receive an AMBA ERROR response when the core is in User mode and when an operation initiated under User mode is active.

46.2.3 ROM area

The ROM area only supports AMBA read operations. Write or locked access operations will receive AMBA ERROR responses. When a read access is made to the ROM area the core will perform a read operation on the memory device. The system has support for AMBA SPLIT responses and the core will issue command SPLIT the master until the read operation on the memory device has finished.

The AMBA read operation is transfered onto the external SPI interface using the parameters set in the configuration register. The read command bit field determines if normal or fast read is used. The additional bit fields determine the length of address and dummy state. The length of address and dummy bit fields are defined number of bytes.

Next is an example of using normal read. To enable normal read user should use the read command 0x3 and set number of dummy bytes to 0x0.

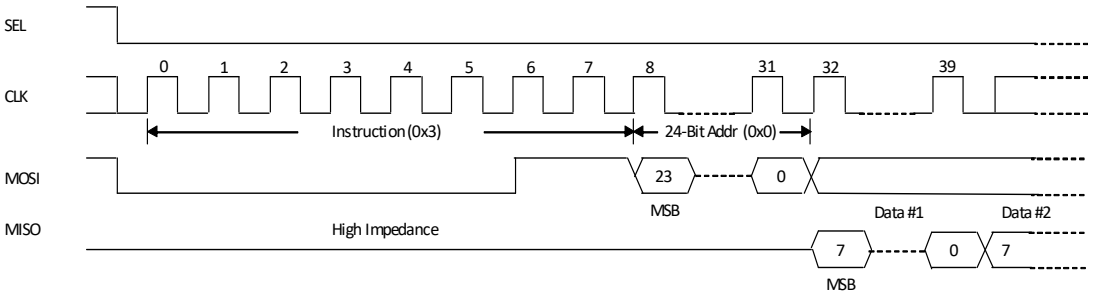


Figure 106. Read Data Bytes (READ) Instruction sequence and Data-Out sequence

Next is an example of using fast read. To enable normal read user should use the read command 0xB and set number of dummy bytes to 0x0 or greater. Note that the dummy byte bit field is set to 0x1 in the example but is set to 0x0 as default. The reason for this is that external SPI PROM might require additional dummy states at 50 MHz to return correct data.

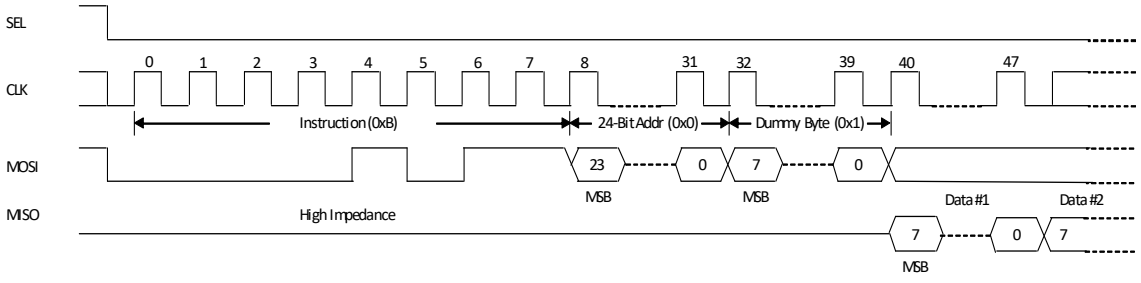


Figure 107. Read Data Bytes at higher speed (FAST_READ) Instruction sequence and Data-Out sequence

The expected read performance is determined by the following factors:

- Scaler mode (CTRL.EAS)

- Number of address bytes used (CONF.ADDRBYTES)
- Fast or normal read mode i.e. number of dummy bytes required (CONF.DUMMYBYTES)
- Number of Data bytes read (AMBA transaction length. Valid length 1,2 or 4 bytes)
- EDAC Enabled (ECONF.EE)
- Internal system delay (approximately 3 SPI clock cycles)

The number of system clocks required for a SPI READ operation can estimated using the formula:

$SPIOP_{ClkCycles}$ = (4 - CONF.ADDRBYTES + CONF.DUMMYBYTES + <NbrOfBytes>) x 8 x (8 - 6 x CTRL.EAS) x (1 + ECONF.EE)

For an 32 bit instruction fetch using normal scaler this would result in approximately 540 system clocks.

BCH EDAC protection requires two consecutive reads from two non-consecutive address i.e. a SPI READ operation with EDAC enabled will require twice as many system clocks to be completed.

46.2.4 BCH EDAC

The SPIMCTRL is provided with an BCH EDAC that can correct one error and detect two errors in a 32-bit word. For each word, a 7-bit checksum is generated according to the equations below. A correctable error will be handled transparently by the memory controller, but adding one waitstate to the access. If an un-correctable error (double-error) is detected, the current AHB cycle will end with an error response. The EDAC can be used during access to SPI Memories areas by setting the EDAC enable bits in the EDAC configuration register. The equations below show how the EDAC checkbits are generated:

```
CB0 = D0 ^ D4 ^ D6 ^ D7 ^ D8 ^ D9 ^ D11 ^ D14 ^ D17 ^ D18 ^ D19 ^ D21 ^ D26 ^ D28 ^ D29 ^ D31
CB1 = D0 ^ D1 ^ D2 ^ D4 ^ D6 ^ D8 ^ D10 ^ D12 ^ D16 ^ D17 ^ D18 ^ D20 ^ D22 ^ D24 ^ D26 ^ D28
CB2 = D0 ^ D3 ^ D4 ^ D7 ^ D9 ^ D10 ^ D13 ^ D15 ^ D16 ^ D19 ^ D20 ^ D23 ^ D25 ^ D26 ^ D29 ^ D31
CB3 = D0 ^ D1 ^ D5 ^ D6 ^ D7 ^ D11 ^ D12 ^ D13 ^ D16 ^ D17 ^ D21 ^ D22 ^ D23 ^ D27 ^ D28 ^ D29
CB4 = D2 ^ D3 ^ D4 ^ D5 ^ D6 ^ D7 ^ D14 ^ D15 ^ D18 ^ D19 ^ D20 ^ D21 ^ D22 ^ D23 ^ D30 ^ D31
CB5 = D8 ^ D9 ^ D10 ^ D11 ^ D12 ^ D13 ^ D14 ^ D15 ^ D24 ^ D25 ^ D26 ^ D27 ^ D28 ^ D29 ^ D30 ^ D31
CB6 = D0 ^ D1 ^ D2 ^ D3 ^ D4 ^ D5 ^ D6 ^ D7 ^ D24 ^ D25 ^ D26 ^ D27 ^ D28 ^ D29 ^ D30 ^ D31
```

Data is always accessed as words (4 bytes at a time) and the corresponding checkbits are located at the address acquired by inverting the word address using it as a byte address. The chip-select is kept active. A word written as four bytes to addresses 0, 1, 2, 3 will have its checkbits at address 0xFFFFFFF, addresses 4, 5, 6, 7 at 0xFFFFFFE and so on. All the bits up to the maximum bank size will be inverted while the same chip-select is always asserted. This way all the bank sizes can be supported and no memory will be unused (except for a maximum of 4 byte in the gap between the data and checkbit area). A read access will automatically read the four data bytes individually from the nominal addresses and the EDAC checkbit byte from the top part of the bank.

Write accesses are not being handled automatically. Instead, write accesses must only be performed as individual word accesses by the software, writing one word at a time, and the corresponding checkbit byte must be calculated and be written to the correct location by the software.

If a correctable EDAC error is detected during a memory read, the ERR bit in the EDAC Status Register is set and. If an uncorrectable EDAC error is detected during a read operation, the MERR bit in the EDAC Status Register will be set and an error response will be generated on the AHB access.

46.2.5 Extended Address mode operation

The SPIMCTRL can operate in 4 byte address mode when access at address area for 4 byte address access, see Table 20. Or when the SPIM CTRL interface is forced via the control bit CONF.F4B.

46.3 Registers

The core is programmed through registers mapped into AHB address space.

*Table 636.*SPIMCTRL registers

AHB address offset	Register
0x00	Configuration register
0x04	Control register
0x08	Status register
0x0C	Receive register
0x10	Transmit register
0x14	EDAC configuration register
0x18	EDAC status register

46.3.1 Configuration Register

Table 637. 0x00 - CONF - Configuration register

31 24	23 16	15 14 13	12	11 10	9 8	7 0
AMASK	READCMD4	RESERVED	F4B	DUMMYBYTES	ADDRBYTES	READCMD
0x0	0x13	0	0x0	0x0	0x1	0x3
rw	rw	r	rw	rw	rw	rw

31 :24 Address Mask Bits (AMASK) - Mask highest address bits

23 :16 Read instruction (READCMD4) - Read instruction that the core will use for reading from the memory device in 4byte address mode.

15 :12 RESERVED

12 Force 4 Byte Address Mode (F4B)

11:10 Use Dummy Byte (DUMMYBYTES) - Insert dummy bytes after last address bytes for higher speed rates i.e. when read instruction bit is set to use FAST read mode. The bit field DUMMYBYTES is the number of dummy bytes that is inserted after the last address byte.

9:8 Reduce number of address bytes (ADDRBYTES) - Default number of address bytes is set to 3.

"00" Use 4 address bytes

"01" Use 3 address bytes (Default)

"10" Use 2 address byte

"11" Use 1 address bytes

7:0 Read instruction (READCMD) - Read instruction that the core will use for reading from the memory device.

46.3.2 Control Register

Table 638. 0x04 - CTRL - Control register

31 5	4	3	2	1	0
RESERVED	RST	CSN	EAS	IEN	USRC
	0	1	0	0	0
	rw	rw	rw	rw	rw

31 :5 RESERVED

4 Reset core (RST) - By writing ‘1’ to this bit the user can reset the core. This bit is automatically cleared when the core has been reset. Reset core should be used with care. Writing this bit has the same effect as system reset. Any ongoing transactions, both on AMBA and to the SPI device will be aborted.

3 Chip select (CSN) - Controls core chip select signal. This field always shows the level of the core’s internal chip select signal. This bit is always automatically set to ‘1’ when leaving User mode by writing USRC to ‘0’.

2 Enable Alternate Scaler (EAS) - ’0’: SPI clock is system clock divided by 8.
’1’: SPI clock is system clock divided by 2.

1 Interrupt Enable (IEN) - When this bit is set the core will generate an interrupt when a User mode transfer completes.

0 User control (USRC) - When this bit is set to ‘1’ the core will accept SPI data via the transmit register. Accesses to the memory mapped device area will return AMBA ERROR responses.

46.3.3 Status Register

Table 639. 0x08 - STAT - Status register

31 3	2	1	0
RESERVED	INIT	BUSY	DONE
0	0	0	0
r	r	r	wc

*Table 639.*0x08 - STAT - Status register

31:3	RESERVED
2	Initialized (INIT) - This read only bit is set to '1' when the SPI memory device has been initialized. Accesses to the ROM area should only be performed when this bit is set to '1'.
1	Core busy (BUSY) - This bit is set to '1' when the core is performing an SPI operation.
0	Operation done (DONE) - This bit is set to '1' when the core has transferred an SPI command in user mode.

Reset value: 0x00000000

46.3.4 Receive Register

*Table 640.*0x0C - RX - Receive register

31 8	7 0
RESERVED	RDATA
0	nr
R	rw

31 :8	RESERVED
7:0	Receive data (RDATA) : Contains received data byte

Reset value: 0x000000UU, where U is undefined

46.3.5 Transmit Register

*Table 641.*0x10 - TX - Transmit register

31 8	7 0
RESERVED	TDATA
0	0
r	rw

31 :8	RESERVED
7:0	Transmit data (TDATA) - Data byte to transmit

46.3.6 EDAC Configuration Register

*Table 642.*0x14 - ECONF - EDAC Configuration register

31 2	1	0
RESERVED	EA	EE
0	0x0	0x0
r	rw	rw

31 :2	RESERVED
1	Enable AHB Error Response (EA) - When enabled error response (AMBA HRESP_ERROR) is provided when multiple bit errors has been detected
0	Enable EDAC BCH Protection (EE) - Enables BCH protection and correction

46.3.7 EDAC Status Register

*Table 643.*0x18 - ESTAT - EDAC Status register

31 2	1	0
EADDR	MERR	ERR
0	0x0	0x0
r	rw	rw

*Table 643.*0x18 - ESTAT - EDAC Status register

31 :2	Address when the last error occurred
1	Data word with multiple bit errors has been detected
0	Correctable Errors has been detected

47 AMBA Protection Unit

The GR716B microcontroller comprises two separate AMBA memory protection units (MEMPROT). The MEMPROT units described in this section have the capability to detect and protect memory areas from write accesses.

The first AMBA memory protection units (MEMPROT0) is connected to Main AHB bus and the second AMBA memory protection units (MEMPROT1) is connected to the DMA AMBA bus. Each AMBA memory protection unit has a unique AMBA address described in chapter 2.10 for configuration and status.

The control and status registers for the AMBA memory protection units are located on the APB bus in the address range from 0x80005000 to 0x80005FFF and in the range from 0x8010A000 to 0x8010AFFF. See AMBA memory protection units connections in the next drawing. The figure shows memory locations and functions used for AMBA memory protection units configuration and control.

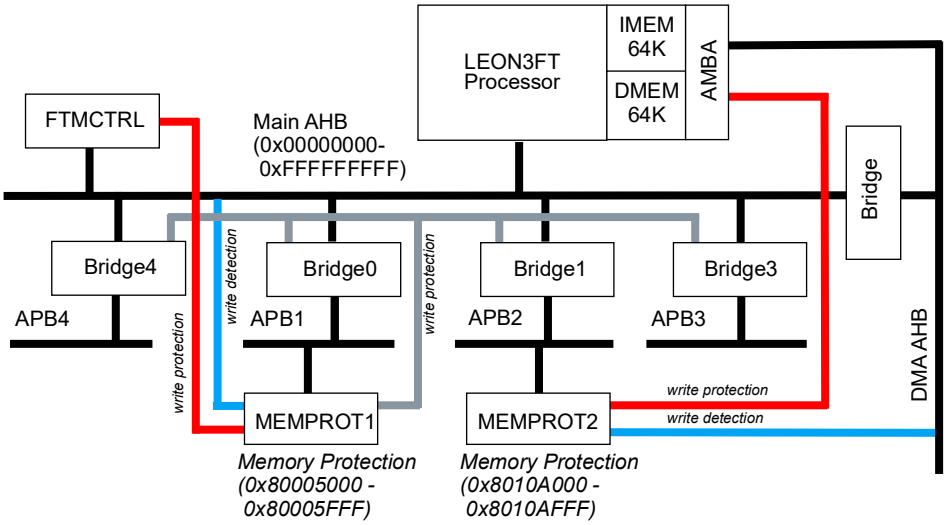


Figure 108. GR716B MEMPROTx bus connection

The primary clock gating unit **GRCLKGATE** described in section 27 is used to enable/disable the AMBA memory protection units. The unit **GRCLKGATE** can also be used to perform reset of individual AMBA memory protection units. Software must enable clock and release reset described in section 27 before configuration.

The system can be configured to protect and restrict access to the AMBA memory protection units.

47.1 Overview

The AMBA Protection unit allows user to define memory segments for protection, memory segments are defined by an start and stop address, to which write permissions can be set. The AMBA protection unit supports up to four individual segments for the system bus and four segments for the dma bus.

The memory protection unit can also restrict write access to individual APB slave interface for specific AHB masters. The restriction needs to be enabled by the user or software. It should be noted that write access to registers in the memory protection can not be restricted to prevent situation where the system can't control APB accesses.

The LEON3FT microcontroller includes 2 separate memory protection units hence register map is split into separate chapters for system and dma bus. The first protection unit monitors masters accesses on the system bus and the second protection unit monitors masters accesses on the DMA bus.

47.2 Operation

The External memory controller, On-chip memory controller and APB controller allows the software to define write protected memory segments, memory segments are defined by a start address and end address, to which write permissions for specific bus masters can be granted or denied. Four segments can be identified with a segment ID between 0 to 3. A segment with a low ID has precedence over one with a high ID, but only if the segment with lower ID is enabled. The precedence or segment ID are only of interests when specified memory area overlaps or bus masters are the same.

Each segment can be configured to grant or deny a write access individually for each AMBA master on the bus. This is done by setting the Enable bits in the relevant Configuration register for the segment.

The protection unit on the main bus provides also access control registers, to manage grants for each master to write in APB slaves on the main bus. They restricts write access to selected APB slaves for each master when the corresponding bit in the corresponding register is set high.

47.3 Registers

The core is programmed through registers mapped into APB address space.

Table 644. AHB system and DMA protection configuration and status registers

APB address offset	Registers
Memory Protection Unit for system bus (0x80005000)	
0x80005000	Protection Configuration register
0x80005004	Protection Segment 0 Start Address register
0x80005008	Protection Segment 0 End Address register
0x8000500C	Protection Segment 0 Configuration register
0x80005010	Not used
0x80005014	Protection Segment 1 Start Address register
0x80005018	Protection Segment 1 End Address register
0x8000501C	Protection Segment 1 Configuration register
0x80005020	Not used
0x80005024	Protection Segment 2 Start Address register
0x80005028	Protection Segment 2 End Address register
0x8000502C	Protection Segment 2 Configuration register
0x80005030	Not used
0x80005034	Protection Segment 3 Start Address register
0x80005038	Protection Segment 3 End Address register
0x8000503C	Protection Segment 3 Configuration register
0x80005040 - 0x800050FF	Not used
0x80005100	Access control for CPU and BRIDGE on APB bus 0
0x80005104	Access control for Scrubber on APB bus 0
0x80005108 - 0x8000510F	Not used
0x80005110	Access control for DMA controller #0 and # 1 on APB bus 0
0x80005114	Access control for DMA controller #2 and # 3 on APB bus 0
0x80005118 - 0x8000513F	Not used
0x80005140	Access control for CPU and BRIDGE on APB bus 1
0x80005144	Access control for Scrubber on APB bus 1
0x80005148 - 0x8000514F	Not used
0x80005150	Access control for DMA controller #0 and # 1 on APB bus 1

Table 644. AHB system and DMA protection configuration and status registers

APB address offset	Registers
0x80005154	Access control for DMA controller #2 and # 3 on APB bus 1
0x80005158 - 0x8000517F	Not used
0x80005180	Access control for CPU and BRIDGE on APB bus 3
0x80005184	Access control for Scrubber on APB bus 3
0x80005188 - 0x8000518F	Not used
0x80005190	Access control for DMA controller #0 and # 1 on APB bus 3
0x80005194	Access control for DMA controller #2 and # 3 on APB bus 3
0x80005198 - 0x800051BF	Not used
0x800051C0	Access control for CPU and BRIDGE on APB bus 4
0x800051C4	Access control for Scrubber on APB bus 4.
0x800051C8 - 0x800051DF	Not used
0x800051E0	Access control for DMA controller #0 and # 1 on APB bus 4
0x800051E4	Access control for DMA controller #2 and # 3 on APB bus 4
0x800051E8 - 0x80005FFF	Not used
Memory Protection Unit for DMA bus (0x8010A000)	
0x8010A000	Protection Configuration register
0x8010A004	Protection Segment 0 Start Address register
0x8010A008	Protection Segment 0 End Address register
0x8010A00C	Protection Segment 0 Configuration register
0x8010A010	Not used
0x8010A014	Protection Segment 1 Start Address register
0x8010A018	Protection Segment 1 End Address register
0x8010A01C	Protection Segment 1 Configuration register
0x8010A020	Not used
0x8010A024	Protection Segment 2 Start Address register
0x8010A028	Protection Segment 2 End Address register
0x8010A02C	Protection Segment 2 Configuration register
0x8010A030	Not used
0x8010A034	Protection Segment 3 Start Address register
0x8010A038	Protection Segment 3 End Address register
0x8010A03C	Protection Segment 3 Configuration register
0x8010A040 - 0x8010AFFF	Not used

47.3.1 System Protection register description

This chapter specifies access control registers for peripherals and registers accessible 0x40000000 to 0x4FFFFFFF and the range 0x80000000 to 0x8041FFFF.

Table 645. 0x80005000 - PCR - Protection Configuration register

31 24	23 4	3 1	0
NSEG	Reserved	PROT	EN
0x4	0x0	0x0	0
r	r	rw	rw

31: 24 NSEG - Number of segments supported.

23: 4 Reserved

3: 1 PROT - Protection of memory control access. This bit field needs to be set to 101b in order to be able to change any register configuration of the memory protection.

0 EN - Enable Memory Protection of specified memory segments. This is bit is used to enable and disable all protected segments at the same-time.

Table 646. 0x80005004 + segment*0x10 - PSA - Protection Segment Start Address register

31 0
SADDR
0x0
rw

31: 0 SADDR - Start address of segment. Start address should be in the range 0x40000000 to 0x4FFFFFFF and the range 0x80000000 to 0x8041FFFF.

Table 647. 0x80005008 + segment*0x10 - PEA - Protection Segment End Address register

31 0
EADDR
0x0
rw

31: 0 EADDR - End address of segment. End address should be in the range 0x40000000 to 0x4FFFFFFF and the range 0x80000000 to 0x8041FFFF.

Table 648. 0x8000500C + segmant*0x10 - PSC - Protection Segment Control register

31 30 29 28 27 26 25 24 23 22 21 20 19	18	17	16	15 1	0
Reserved	G2	G1	G0	Reserved	EN
0	0	0	0	0	0
r	rw	rw	rw	r	rw

31: 19 RESERVED

18 G2 - Grant SCRUBBER on the main bus exclusive write permission

17 G1 - Grant DMA bus masters exclusive write permission. DMA bus masters can be any master performing accesses on the DMA bus i.e. SpaceWire, CAN, MIL-1553, UART, I2C, and/or DMA

16 G0 - Grant LEON3FT processor exclusive write permission

15: 1 RESERVED

0 EN - Enable Memory Protection for specified memory segments. This bit will grant exclusive write permission to specified masters within protected memory segment

47.3.2 Protection register for AMBA APB 0

This chapter specifies access control registers for peripherals and registers accessible 0x80000000 to 0x8000FFFF.

Table 649. 0x80005100 - APB0PROT0 - APB Control 0 Protection register 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C15	C14	C13	C12	C11	C10	C9	C8	C7	C6	C5	C4	C3	C2	C1	C0	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

31		Memory controller with EDAC (C15)
30		Multi-processor Interrupt Ctrl. (C14)
29	C	Modular Timer Unit 0 (C13)
28	P	Modular Timer Unit 1 (C12)
27	U	Memory Protection Unit for system bus (C11)
26		Clock gating configuration register unit 0 (C10)
25	C	Clock gating configuration register unit 1 (C9)
23	O	Configuration and test registers (C8)
24	N	LEON3 Statistics Unit (C7)
22	T	AHB Status Register (C6)
21	R	On-chip Instruction memory control registers (C5)
20	O	CCSDS TDP / SpaceWire I/F (C4)
19	L	IO Mux configuration register (C3)
18		CCSDS TDP / SpaceWire I/F (C2)
17		Test register used for test purpose (C1)
16		Slave UART configuration (C0)
15		Memory controller with EDAC (B15)
14	B	Multi-processor Interrupt Ctrl. (B14)
13	R	Modular Timer Unit 0 (B13)
12	I	Modular Timer Unit 1 (B12)
11	D	Memory Protection Unit for system bus (B11)
10	G	Clock gating configuration register unit 0 (B10)
9	E	Clock gating configuration register unit 1 (B9)
8		Configuration and test registers (B8)
7	C	LEON3 Statistics Unit (B7)
6	O	AHB Status Register (B6)
5	N	On-chip Instruction memory control registers (B5)
4	T	CCSDS TDP / SpaceWire I/F (B4)
3	R	IO Mux configuration register (B3)
2	L	CCSDS TDP / SpaceWire I/F (B2)
1		Test register used for test purpose (B1)
0		Slave UART configuration (B0)

Table 650. 0x0x80005104 - APB0PROT1 - APB Control 0 Protection register 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
0x0																0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
r																rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Table 650. 0x0x80005104 - APB0PROT1 - APB Control 0 Protection register 1

31: 16		Not Used
15		Memory controller with EDAC (B15)
14		Multi-processor Interrupt Ctrl. (B14)
13	S	Modular Timer Unit 0 (B13)
12	C	Modular Timer Unit 1 (B12)
11	R	Memory Protection Unit for system bus (B11)
10	U	Clock gating configuration register unit 0 (B10)
9	B	Clock gating configuration register unit 1 (B9)
8		Configuration and test registers (B8)
7	C	LEON3 Statistics Unit (B7)
6	O	AHB Status Register (B6)
5	N	On-chip Instruction memory control registers (B5)
4	T	CCSDS TDP / SpaceWire I/F (B4)
3	R	IO Mux configuration register (B3)
2	L	CCSDS TDP / SpaceWire I/F (B2)
1		Test register used for test purpose (B1)
0		Slave UART configuration (B0)

Table 651. 0x0x80005110 - APB0PROT2 - APB Control 0 Protection register 2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

31		Memory controller with EDAC (A15)
30	D	Multi-processor Interrupt Ctrl. (A14)
29	M	Modular Timer Unit 0 (A13)
28	A	Modular Timer Unit 1 (A12)
27	0	Memory Protection Unit for system bus (A11)
26		Clock gating configuration register unit 0 (A10)
25	C	Clock gating configuration register unit 1 (A9)
23	O	Configuration and test registers (A8)
24	N	LEON3 Statistics Unit (A7)
22	T	AHB Status Register (A6)
21	R	On-chip Instruction memory control registers (A5)
20	O	CCSDS TDP / SpaceWire I/F (A4)
19	L	IO Mux configuration register (A3)
18		CCSDS TDP / SpaceWire I/F (A2)
17		Test register used for test purpose (A1)
16		Slave UART configuration (A0)
15		Memory controller with EDAC (B15)
14	D	Multi-processor Interrupt Ctrl. (B14)
13	M	Modular Timer Unit 0 (B13)
12	A	Modular Timer Unit 1 (B12)
11	1	Memory Protection Unit for system bus (B11)
10		Clock gating configuration register unit 0 (B10)
9	C	Clock gating configuration register unit 1 (B9)
8	O	Configuration and test registers (B8)

Table 651. 0x0x80005110 - APB0PROT2 - APB Control 0 Protection register 2

7	N	LEON3 Statistics Unit (B7)
6	T	AHB Status Register (B6)
5	R	On-chip Instruction memory control registers (B5)
4	O	CCSDS TDP / SpaceWire I/F (B4)
3	L	IO Mux configuration register (B3)
2		CCSDS TDP / SpaceWire I/F (B2)
1		Test register used for test purpose (B1)
0		Slave UART configuration (B0)

Table 652. 0x0x80005114 - APB0PROT3 - APB Control 0 Protection register 3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

31		Memory controller with EDAC (A15)
30	D	Multi-processor Interrupt Ctrl. (A14)
29	M	Modular Timer Unit 0 (A13)
28	A	Modular Timer Unit 1 (A12)
27	2	Memory Protection Unit for system bus (A11)
26		Clock gating configuration register unit 0 (A10)
25	C	Clock gating configuration register unit 1 (A9)
23	O	Configuration and test registers (A8)
24	N	LEON3 Statistics Unit (A7)
22	T	AHB Status Register (A6)
21	R	On-chip Instruction memory control registers (A5)
20	O	CCSDS TDP / SpaceWire I/F (A4)
19	L	IO Mux configuration register (A3)
18		CCSDS TDP / SpaceWire I/F (A2)
17		Test register used for test purpose (A1)
16		Slave UART configuration (A0)
15		Memory controller with EDAC (B15)
14	D	Multi-processor Interrupt Ctrl. (B14)
13	M	Modular Timer Unit 0 (B13)
12	A	Modular Timer Unit 1 (B12)
11	3	Memory Protection Unit for system bus (B11)
10		Clock gating configuration register unit 0 (B10)
9	C	Clock gating configuration register unit 1 (B9)
8	O	Configuration and test registers (B8)
7	N	LEON3 Statistics Unit (B7)
6	T	AHB Status Register (B6)
5	R	On-chip Instruction memory control registers (B5)
4	O	CCSDS TDP / SpaceWire I/F (B4)
3	L	IO Mux configuration register (B3)
2		CCSDS TDP / SpaceWire I/F (B2)
1		Test register used for test purpose (B1)
0		Slave UART configuration (B0)

47.3.3 Protection register for AMBA APB 1

This chapter specifies access control registers for peripherals and registers accessible 0x80100000 to 0x8010FFFF.

Table 653. 0x0x80005140 - APB1PROT0 - APB Control 1 Protection register 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C15	C14	C13	C12	C11	C10	C9	C8	C7	C6	C5	C4	C3	C2	C1	C0	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

31		GRSPW2 SpaceWire Serial Link (C15)
30		MIL-STD-1553B Interface. (C14)
29	C	CAN Controller with DMA (C13)
28	P	CAN Controller with DMA (C12)
27	U	SPI to AHB Bridge (C11)
26		I2C to AHB Bridge (C10)
25	C	Stand alone DMA unit 0 (C9)
23	O	Stand alone DMA unit 1 (C8)
24	N	Stand alone DMA unit 2 (C7)
22	T	Stand alone DMA unit 3 (C6)
21	R	On-chip Instruction memory control registers (C5)
20	O	Memory protection for DMA bus (C4)
19	L	IO Mux configuration register (C3)
18		PLL control registers (C2)
17		PacketWire Receiver with DMA (C1)
16		PacketWire Transmitter with DMA (C0)
15		GRSPW2 SpaceWire Serial Link (B15)
14	B	MIL-STD-1553B Interface (B14)
13	R	CAN Controller with DMA (B13)
12	I	CAN Controller with DMA (B12)
11	D	SPI to AHB Bridge (B11)
10	G	I2C to AHB Bridge (B10)
9	E	Stand alone DMA unit 0 (B9)
8		Stand alone DMA unit 1 (B8)
7	C	Stand alone DMA unit 2 (B7)
6	O	Stand alone DMA unit 3 (B6)
5	N	Memory protection for DMA bus (B5)
4	T	CCSDS TDP / SpaceWire I/F (B4)
3	R	Brown-Out detection control registers (B3)
2	L	PLL control registers (B2)
1		PacketWire Receiver with DMA (B1)
0		PacketWire Transmitter with DMA (B0)

Table 654. 0x0x80005144 - APB1PROT1 - APB Control 1 Protection register 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
0x0																0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
r																rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Table 654. 0x0x80005144 - APB1PROT1 - APB Control 1 Protection register 1

31: 16		Not Used
15		GRSPW2 SpaceWire Serial Link (B15)
14		MIL-STD-1553B Interface (B14)
13	S	CAN Controller with DMA (B13)
12	C	CAN Controller with DMA (B12)
11	R	SPI to AHB Bridge (B11)
10	U	I2C to AHB Bridge (B10)
9	B	Stand alone DMA unit 0 (B9)
8		Stand alone DMA unit 1 (B8)
7	C	Stand alone DMA unit 2 (B7)
6	O	Stand alone DMA unit 3 (B6)
5	N	Memory protection for DMA bus (B5)
4	T	CCSDS TDP / SpaceWire I/F (B4)
3	R	Brown-Out detection control registers (B3)
2	L	PLL control registers (B2)
1		PacketWire Receiver with DMA (B1)
0		PacketWire Transmitter with DMA (B0)

Table 655. 0x0x80005150 - APB1PROT2 - APB Control 1 Protection register 2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

31		GRSPW2 SpaceWire Serial Link (C15)
30	D	MIL-STD-1553B Interface. (C14)
29	M	CAN Controller with DMA (C13)
28	A	CAN Controller with DMA (C12)
27	0	SPI to AHB Bridge (C11)
26		I2C to AHB Bridge (C10)
25	C	Stand alone DMA unit 0 (C9)
23	O	Stand alone DMA unit 1 (C8)
24	N	Stand alone DMA unit 2 (C7)
22	T	Stand alone DMA unit 3 (C6)
21	R	On-chip Instruction memory control registers (C5)
20	O	Memory protection for DMA bus (C4)
19	L	IO Mux configuration register (C3)
18		PLL control registers (C2)
17		PacketWire Receiver with DMA (C1)
16		PacketWire Transmitter with DMA (C0)
15		GRSPW2 SpaceWire Serial Link (B15)
14	D	MIL-STD-1553B Interface (B14)
13	M	CAN Controller with DMA (B13)
12	A	CAN Controller with DMA (B12)
11	1	SPI to AHB Bridge (B11)
10		I2C to AHB Bridge (B10)
9	C	Stand alone DMA unit 0 (B9)
8	O	Stand alone DMA unit 1 (B8)

Table 655. 0x0x80005150 - APB1PROT2 - APB Control 1 Protection register 2

7	N	Stand alone DMA unit 2 (B7)
6	T	Stand alone DMA unit 3 (B6)
5	R	Memory protection for DMA bus (B5)
4	O	CCSDS TDP / SpaceWire I/F (B4)
3	L	Brown-Out detection control registers (B3)
2		PLL control registers (B2)
1		PacketWire Receiver with DMA (B1)
0		PacketWire Transmitter with DMA (B0)

Table 656. 0x0x80005154 - APB1PROT3 - APB Control 1 Protection register 3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

31		GRSPW2 SpaceWire Serial Link (C15)
30	D	MIL-STD-1553B Interface. (C14)
29	M	CAN Controller with DMA (C13)
28	A	CAN Controller with DMA (C12)
27	2	SPI to AHB Bridge (C11)
26		I2C to AHB Bridge (C10)
25	C	Stand alone DMA unit 0 (C9)
23	O	Stand alone DMA unit 1 (C8)
24	N	Stand alone DMA unit 2 (C7)
22	T	Stand alone DMA unit 3 (C6)
21	R	On-chip Instruction memory control registers (C5)
20	O	Memory protection for DMA bus (C4)
19	L	IO Mux configuration register (C3)
18		PLL control registers (C2)
17		PacketWire Receiver with DMA (C1)
16		PacketWire Transmitter with DMA (C0)
15		GRSPW2 SpaceWire Serial Link (B15)
14	D	MIL-STD-1553B Interface (B14)
13	M	CAN Controller with DMA (B13)
12	A	CAN Controller with DMA (B12)
11	3	SPI to AHB Bridge (B11)
10		I2C to AHB Bridge (B10)
9	C	Stand alone DMA unit 0 (B9)
8	O	Stand alone DMA unit 1 (B8)
7	N	Stand alone DMA unit 2 (B7)
6	T	Stand alone DMA unit 3 (B6)
5	R	Memory protection for DMA bus (B5)
4	O	CCSDS TDP / SpaceWire I/F (B4)
3	L	Brown-Out detection control registers (B3)
2		PLL control registers (B2)
1		PacketWire Receiver with DMA (B1)
0		PacketWire Transmitter with DMA (B0)

47.3.4 Protection register for AMBA APB 3

This subsection specifies access control registers for peripherals and registers accessible 0x80000 to 0x8030FFFF.

Table 657. 0x0x80005180 - APB3PROT0 - APB Control 3 Protection register 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C15	C14	C13	C12	C11	C10	R	R	C7	C6	C5	C4	C3	C2	C1	C0	B15	B14	B13	B12	B11	B10	R	R	B7	B6	B5	B4	B3	B2	B1	B0
0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
rw	rw	rw	rw	rw	rw	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	rw	rw	rw	rw	rw	rw	rw	rw

31		Generic UART 0 (C15)
30		Generic UART 1 (C14)
29	C	Generic UART 2 (C13)
28	P	Generic UART 3(C12)
27	U	Generic UART 4 (C11)
26		Generic UART 5 (C10)
25	C	unused
24	O	unused
23	N	External ADC / DAC Interface (C7)
22	T	SPI Controller 0 (C6)
21	R	SPI Controller 1 (C5)
20	O	PWM generator (C4)
19	L	General Purpose I/O port 0 to 31(C3)
18		General Purpose I/O port 32 to 64 (C2)
17		I2C-master 0 (C1)
16		I2C-master 1 (C0)
15		Generic UART 0 (B15)
14	B	Generic UART 1 (B14)
13	R	Generic UART 2 (B13)
12	I	Generic UART 3(B12)
11	D	Generic UART 4 (B11)
10	G	Generic UART 5 (B10)
9	E	unused
8		unused
7	C	External ADC / DAC Interface (B7)
6	O	SPI Controller 0 (B6)
5	N	SPI Controller 1 (B5)
4	T	PWM generator (B4)
3	R	General Purpose I/O port 0 to 31(B3)
2	L	General Purpose I/O port 32 to 64 (B2)
1		I2C-master 0 (B1)
0		I2C-master 1 (B0)

Table 658. 0x0x80005184 - APB3PROT1 - APB Control 3 Protection register 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																B15	B14	B13	B12	B11	B10	R	R	B7	B6	B5	B4	B3	B2	B1	B0
0x0																0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
r																rw	rw	rw	rw	rw	rw	r	r	rw	rw	rw	rw	rw	rw	rw	rw

Table 658. 0x0x80005184 - APB3PROT1 - APB Control 3 Protection register 1

31: 16		Not Used
15		Generic UART 0 (B15)
14		Generic UART 1 (B14)
13	S	Generic UART 2 (B13)
12	C	Generic UART 3(B12)
11	R	Generic UART 4 (B11)
10	U	Generic UART 5 (B10)
9	B	unused
8		unused
7	C	External ADC / DAC Interface (B7)
6	O	SPI Controller 0 (B6)
5	N	SPI Controller 1 (B5)
4	T	PWM generator (B4)
3	R	General Purpose I/O port 0 to 31(B3)
2	L	General Purpose I/O port 32 to 64 (B2)
1		I2C-master 0 (B1)
0		I2C-master 1 (B0)

Table 659. 0x0x80005190 - APB3PROT2 - APB Control 3 Protection register 2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A15	A14	A13	A12	A11	A10	R	R	A7	A6	A5	A4	A3	A2	A1	A0	B15	B14	B13	B12	B11	B10	R	R	B7	B6	B5	B4	B3	B2	B1	B0
0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
rw	rw	rw	rw	rw	rw	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	rw	rw	rw	rw	rw	rw	rw	rw

31		Generic UART 0 (A15)
30	D	Generic UART 1 (A14)
29	M	Generic UART 2 (A13)
28	A	Generic UART 3 (A12)
27	0	Generic UART 4 (A11)
26		Generic UART 5 (A10)
25	C	unused
23	O	unused
24	N	External ADC / DAC Interface (A7)
22	T	SPI Controller 0 (A6)
21	R	SPI Controller 1 (A5)
20	O	PWM generator (A4)
19	L	General Purpose I/O port 0 to 31 (A3)
18		General Purpose I/O port 32 to 64 (A2)
17		I2C-master 0 (A1)
16		I2C-master 1 (A0)
15		Generic UART 0 (B15)
14	D	Generic UART 1 (B14)
13	M	Generic UART 2 (B13)
12	A	Generic UART 3(B12)
11	1	Generic UART 4 (B11)
10		Generic UART 5 (B10)
9	C	unused
8	O	unused

Table 659. 0x0x80005190 - APB3PROT2 - APB Control 3 Protection register 2

7	N	External ADC / DAC Interface (B7)
6	T	SPI Controller 0 (B6)
5	R	SPI Controller 1 (B5)
4	O	PWM generator (B4)
3	L	General Purpose I/O port 0 to 31(B3)
2		General Purpose I/O port 32 to 64 (B2)
1		I2C-master 0 (B1)
0		I2C-master 1 (B0)

Table 660. 0x0x80005194 - APB3PROT3 - APB Control 3 Protection register 3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A15	A14	A13	A12	A11	A10	R	R	A7	A6	A5	A4	A3	A2	A1	A0	B15	B14	B13	B12	B11	B10	R	R	B7	B6	B5	B4	B3	B2	B1	B0
0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
rw	rw	rw	rw	rw	rw	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	rw	rw	rw	rw	rw	rw	rw	rw

31		Generic UART 0 (A15)
30	D	Generic UART 1 (A14)
29	M	Generic UART 2 (A13)
28	A	Generic UART 3 (A12)
27	2	Generic UART 4 (A11)
26		Generic UART 5 (A10)
25	C	unused
23	O	unused
24	N	External ADC / DAC Interface (A7)
22	T	SPI Controller 0 (A6)
21	R	SPI Controller 1 (A5)
20	O	PWM generator (A4)
19	L	General Purpose I/O port 0 to 31 (A3)
18		General Purpose I/O port 32 to 64 (A2)
17		I2C-master 0 (A1)
16		I2C-master 1 (A0)
15		Generic UART 0 (B15)
14	D	Generic UART 1 (B14)
13	M	Generic UART 2 (B13)
12	A	Generic UART 3(B12)
11	3	Generic UART 4 (B11)
10		Generic UART 5 (B10)
9	C	unused
8	O	unused
7	N	External ADC / DAC Interface (B7)
6	T	SPI Controller 0 (B6)
5	R	SPI Controller 1 (B5)
4	O	PWM generator (B4)
3	L	General Purpose I/O port 0 to 31(B3)
2		General Purpose I/O port 32 to 64 (B2)
1		I2C-master 0 (B1)
0		I2C-master 1 (B0)

47.3.5 Protection register for AMBA APB 4

This subsection specifies access control registers for peripherals and registers accessible 0x80400000 to 0x8040FFFF.

Table 661. 0x0x800051C0 - APB4PROT0 - APB Control 4 Protection register 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C15	C14	C13	C12	C11	C10	C9	C8	C7	C6	C5	C4	C3	C2	C1	C0	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

31		ADC0 (C15)
30		ADC1 (C14)
29	C	ADC2 (C13)
28	P	ADC3 (C12)
27	U	ADC4 (C11)
26		ADC5 (C10)
25	C	ADC6 (C9)
24	O	ADC7 (C8)
23	N	DAC0 (C7)
22	T	DAC1 (C6)
21	R	DAC2 (C5)
20	O	DAC3 (C4)
19	L	I2C-slave 0 (C3)
18		I2C-slave 1 (C2)
17		PWM generator 1 (C1)
16		SPI for Space slave (C0)
15		ADC0 (B15)
14	B	ADC1 (B14)
13	R	ADC2 (B13)
12	I	ADC3 (B12)
11	D	ADC4 (B11)
10	G	ADC5 (B10)
9	E	ADC6 (B9)
8		ADC7 (B8)
7	C	DAC0 (B7)
6	O	DAC1 (B6)
5	N	DAC2 (B5)
4	T	DAC3 (B4)
3	R	I2C-slave 0 (B3)
2	L	I2C-slave 1 (B2)
1		PWM generator 1 (B1)
0		SPI for Space slave (B0)

Table 662. 0x0x800051C4 - APB4PROT1 - APB Control 3 Protection register 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
0x0																0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
r																rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

31: 16 Not Used

Table 662. 0x0x800051C4 - APB4PROT1 - APB Control 3 Protection register 1

15		ADC0 (B15)
14		ADC1 (B14)
13	S	ADC2 (B13)
12	C	ADC3 (B12)
11	R	ADC4 (B11)
10	U	ADC5 (B10)
9	B	ADC6 (B9)
8		ADC7 (B8)
7	C	DAC0 (B7)
6	O	DAC1 (B6)
5	N	DAC2 (B5)
4	T	DAC3 (B4)
3	R	I2C-slave 0 (B3)
2	L	I2C-slave 1 (B2)
1		PWM generator 1 (B1)
0		SPI for Space slave (B0)

Table 663. 0x0x800051E0 - APB4PROT2 - APB Control 4 Protection register 2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

31		ADC0 (B15)
30	D	ADC1 (B14)
29	M	ADC2 (B13)
28	A	ADC3 (B12)
27	0	ADC4 (B11)
26		ADC5 (B10)
25	C	ADC6 (B9)
23	O	ADC7 (B8)
24	N	DAC0 (B7)
22	T	DAC1 (B6)
21	R	DAC2 (B5)
20	O	DAC3 (B4)
19	L	I2C-slave 0 (B3)
18		I2C-slave 1 (B2)
17		PWM generator 1 (B1)
16		SPI for Space slave (B0)
15		ADC0 (B15)
14	D	ADC1 (B14)
13	M	ADC2 (B13)
12	A	ADC3 (B12)
11	1	ADC4 (B11)
10		ADC5 (B10)
9	C	ADC6 (B9)
8	O	ADC7 (B8)
7	N	DAC0 (B7)
6	T	DAC1 (B6)

Table 663. 0x0x800051E0 - APB4PROT2 - APB Control 4 Protection register 2

5	R	DAC2 (B5)
4	O	DAC3 (B4)
3	L	I2C-slave 0 (B3)
2		I2C-slave 1 (B2)
1		PWM generator 1 (B1)
0		SPI for Space slave (B0)

Table 664. 0x0x800051E4 - APB4PROT3 - APB Control 4 Protection register 3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

31		ADC0 (B15)
30	D	ADC1 (B14)
29	M	ADC2 (B13)
28	A	ADC3 (B12)
27	2	ADC4 (B11)
26		ADC5 (B10)
25	C	ADC6 (B9)
23	O	ADC7 (B8)
24	N	DAC0 (B7)
22	T	DAC1 (B6)
21	R	DAC2 (B5)
20	O	DAC3 (B4)
19	L	I2C-slave 0 (B3)
18		I2C-slave 1 (B2)
17		PWM generator 1 (B1)
16		SPI for Space slave (B0)
15		ADC0 (B15)
14	D	ADC1 (B14)
13	M	ADC2 (B13)
12	A	ADC3 (B12)
11	3	ADC4 (B11)
10		ADC5 (B10)
9	C	ADC6 (B9)
8	O	ADC7 (B8)
7	N	DAC0 (B7)
6	T	DAC1 (B6)
5	R	DAC2 (B5)
4	O	DAC3 (B4)
3	L	I2C-slave 0 (B3)
2		I2C-slave 1 (B2)
1		PWM generator 1 (B1)
0		SPI for Space slave (B0)

47.3.6 DMA protection register description

This chapter specifies access control registers for peripherals and registers accessible 0x30000000 to 0x31FFFFFF.

Table 665. 0x8010A000 - PCR - Protection Configuration register

31 24	23 4	3 1	0
NSEG	Reserved	PROT	EN
0x4	0x0	0x0	1
r	r	rw	rw

31: 24 NSEG - Number of segments supported.

23: 4 Reserved

3: 1 PROT - Protection of memory control access. This bit field needs to be set to 101b in order to be able to change any register configuration of the memory protection.

0 EN - Enable Memory Protection of specified memory segments. This is bit can be used to enable and disable all protected segments at the same-time. This bit is enabled at startup and individual segment can be controlled via separate segment control registers

Table 666. 0x8010A004 + segment*0x10 - PSA - Protection Segment Start Address register

31 0
SADDR
0x0
rw

31: 0 SADDR - Start address of segment. Start address should be in the range 0x30000000 to 0x31FFFFFF.

Table 667. 0x8010A5008 + segment*0x10 - PEA - Protection Segment End Address register

31 0
EADDR
0x0
rw

31: 0 EADDR - End address of segment. End address should be in the range 0x30000000 to 0x31FFFFFF.

Table 668. 0x8010A00C + segmant*0x10 - PSC - Protection Segment Control register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15 ... 1	0
G15	G14	G13	G12	G11	G10	G9	G8	G7	G6	G5	G4	G3	G2	G1	G0	Reserved	EN
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	rw

31	G15 - Grant SPI4S on the DMA bus exclusive write permission
30	G14 - Grant DMA controller #3 on the DMA bus exclusive write permission
29	G13 - Grant DMA controller #2 on the DMA bus exclusive write permission
28	G12 - Grant DMA controller #1 on the DMA bus exclusive write permission
27	G11 - Grant DMA controller #0 on the DMA bus exclusive write permission
26	G10 - Grant PacketWire transmitter on the DMA bus exclusive write permission
25	G9 - Grant PacketWire receiver on the DMA bus exclusive write permission
24	G8 - Grant Bridge from System bus exclusive write permission.
23	G7 - Grant UART on the DMA bus exclusive write permission
22	G6 - Grant CAN on the DMA bus exclusive write permission
21	G5 - Grant CAN on the DMA bus exclusive write permission
20	G4 - Grant I2C on the DMA bus exclusive write permission
19	G3 - Grant SPI on the DMA bus exclusive write permission
18	G2 - Grant SpaceWire on the DMA bus exclusive write permission
17	G1 - Grant MIL-1553 on the DMA bus exclusive write permission.
16	G0 - Grant Bridge from Debug bus exclusive write permission.
15: 1	RESERVED
0	EN - Enable Memory Protection for specified memory segments. This bit will grant exclusive write permission to specified masters within protected memory segment

47.4 Example of configure and use the Memory protection

This chapter gives examples how of the memory protection unit can be used

47.4.1 Protect local instruction memory

To protect the local instruction memory from any erroneously writes during normal operation the protected area start and stop address should be specified and the master given access to the protected area: (For this example we use segment number #0 but any segment can be used for protection)

PSA0.SADDR = 0x31000000
PSA0.SADDR = 0x3100FFFF
PSAC0.GRANT = 0x0000
PSAC0.EN = 0x1
PCR.EN = 0x1

This example defines the protected area, grants no master access to the area and enable segment end global protection.

47.4.2 Protect external SRAM memory

To protect an area in the external SRAM memory from any erroneously writes during normal operation the protected area start and stop address should be specified and the master given access to the protected area: (For this example we use segment number #0 but any segment can be used for protection)

PSA0.SADDR = 0x40100000

PSA0.SADDR = 0x401FFFFF
PSAC0.GRANT = 0x0000
PSAC0.EN = 0x1
PCR.EN = 0x1

This example defines the protected area, grants no master access to the area and enable segment end global protection.

To give a AMBA master access to the protected area change the value in the register PSAC0.GRANT to e.g. 0x0002 to give access to masters accessing the system AMBA bus via the AHB2AHB bridge from the DMA AMBA bus.

47.4.3 Protect clock gating unit from erroneous accesses

To protect the clock gating from erroneous access the APB bridge can be programmed to deny all write accesses or to grant privilege access to specific AMBA bus masters.

The clock gating unit 1 and 2 are located on APB bus 1 and access to clock gating unit 1 and 2 are controlled via register APB0PROT0, APB1PROT1, APBPROT2 and APBPROT3.

To deny all DMA controllers access:

APB1PROT2.A9 = 0x1
APB1PROT2.A10 = 0x1
APB1PROT2.B9 = 0x1
APB1PROT2.B10 = 0x1
APB1PROT3.A9 = 0x1
APB1PROT3.A10 = 0x1
APB1PROT3.B9 = 0x1
APB1PROT3.B10 = 0x1

48 Serial Debug and Remote Access Interface

The GR716B microcontroller comprises two debug and remote access UART units. The UART units described in this section have the capability to respond on external UART signaling and act as a master on the internal bus without software support. The capability to respond to external access without software support differentiates the two remote access UART units from the regular UART units described in chapter 19.

The first unit (AHBUART0) is directly connected to AMBA debug bus and the second unit (AHBUART1) is connected to the DMA AMBA bus. Each unit have a unique AMBA address described in chapter 2.10 for configuration and status.

The first unit is the main debug interface during software development and have direct access to the internal state of the processor and trace buffers. This interface can be disabled during mission via external pin configuration i.e. tie DSU_EN to low.

The second unit is to be used for remote access of the GR716B microcontroller in mission mode i.e. when DSU_EN is low. The second unit is available via the IO switch matrix described in chapter 2.5. The second UART remote access unit is setup via boot straps.

The control and status register for the units are located on APB bus in the address range from 0x8000F000 to 0x8000FFFF and in the range from 0x94000000 to 0x9400FFF. See serial debug and remote access interface units connections in the next drawing. The figure shows memory locations and functions used for interfaces configuration and control.

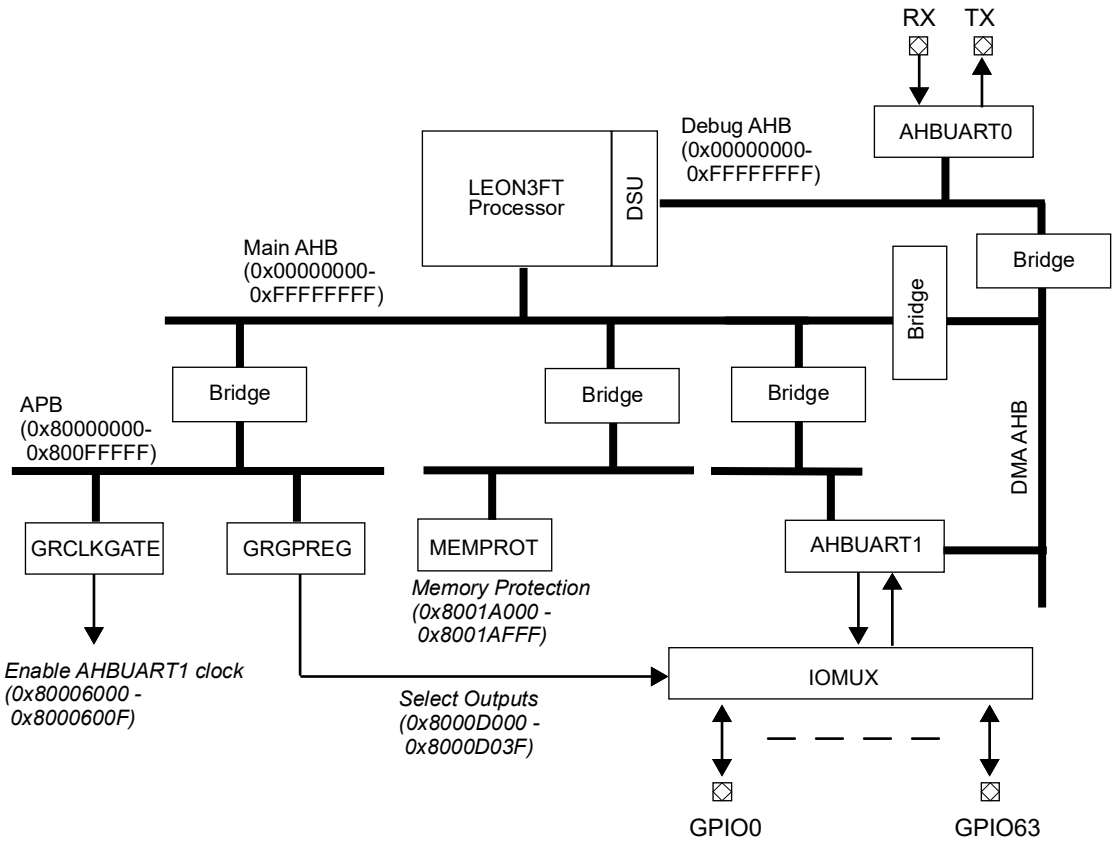


Figure 109. GR716B AHBUARTx bus and pin connection

The primary clock gating unit **GRCLKGATE** described in section 27 is used to enable/disable the AHBUART1. The unit **GRCLKGATE** can also be used to perform reset of the AHBUART1 unit.

Software must enable clock and release reset described in section 27 before configuration and transmission can start.

External IO selection and configuration is made in the system IO configuration registers (**GRGPREG**) in the address range from 0x8000D000 to 0x8000D03F. See section 7.1

The system can be configured to protect and restrict access to the AHBUART1 unit in the **MEMPROT** unit. For more information See section 47 for more information.

48.1 Overview

Each interface consists of a UART connected to the AMBA AHB bus as a master. A simple communication protocol is supported to transmit access parameters and data. Through the communication link, a read or write transfer can be generated to any address on the AMBA AHB bus.

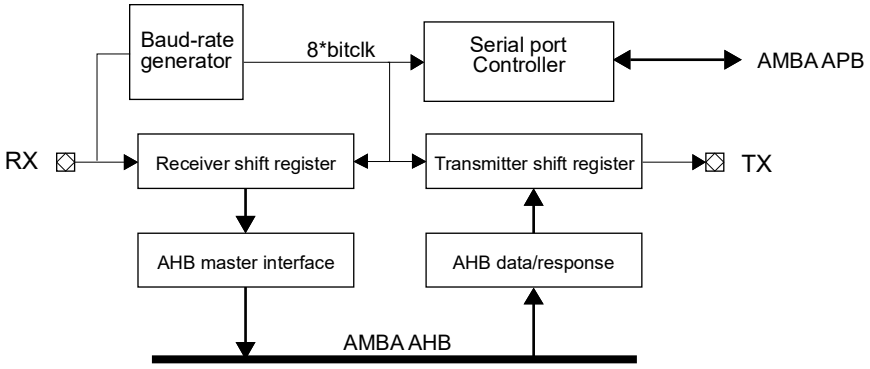


Figure 110. Block diagram

48.2 Operation

48.2.1 Transmission protocol

The interface supports a simple protocol where commands consist of a control byte, followed by a 32-bit address, followed by optional write data. Write access does not return any response, while a read access only returns the read data. Data is sent on 8-bit basis as shown below.

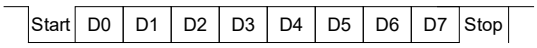


Figure 111. Data frame

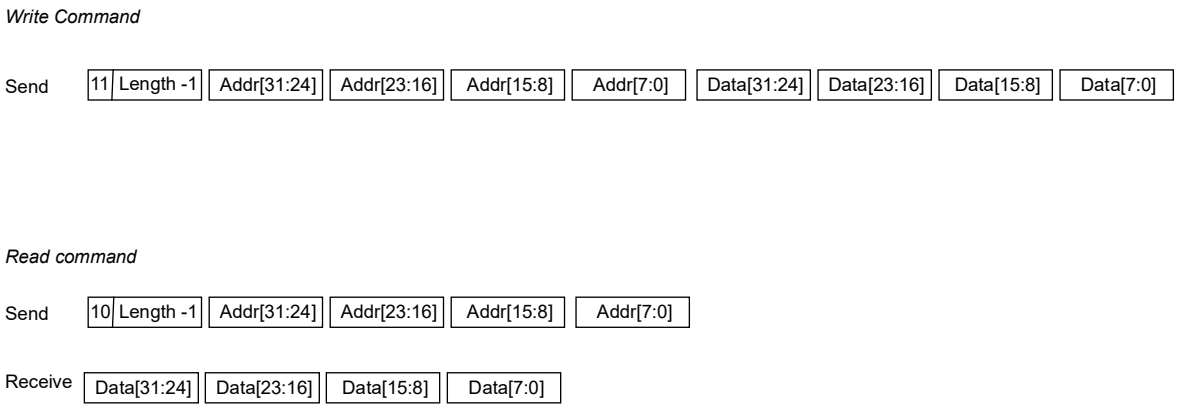


Figure 112. Commands

Block transfers can be performed be setting the length field to n-1, where n denotes the number of transferred words. For write accesses, the control byte and address is sent once, followed by the number of data words to be written. The address is automatically incremented after each data word. For read accesses, the control byte and address is sent once and the corresponding number of data words is returned.

48.2.2 Baud rate generation

The UART contains a 18-bit down-counting scaler to generate the desired baud-rate. The scaler is clocked by the system clock and generates a UART tick each time it underflows. The scaler is reloaded with the value of the UART scaler reload register after each underflow. The resulting UART tick frequency should be 8 times the desired baud-rate.

If not programmed by software, the baud rate will be automatically discovered. This is done by searching for the shortest period between two falling edges of the received data (corresponding to two bit periods). When three identical two-bit periods has been found, the corresponding scaler reload value is latched into the reload register, and the BL bit is set in the UART control register. If the BL bit is reset by software, the baud rate discovery process is restarted. The baud-rate discovery is also restarted when a ‘break’ or framing error is detected by the receiver, allowing to change to baudrate from the external transmitter. For proper baudrate detection, the value 0x55 should be transmitted to the receiver after reset or after sending break.

The best scaler value for manually programming the baudrate can be calculated as follows:

```
scaler = (((system_clk*10)/(baudrate*8))-5)/10
```

48.3 Maximum baudrate

The maximum AHBUART baudrate depends on the system clock frequency. The possible baudrates are of the form SYSFREQ/(8*n) where n is an integer larger than 1, with tolerance of 4%. The theoretical maximum is SYSFREQ/8 (n=1), but the effectiveness of the digital input filter is greatly improved at SYSFREQ/24 (n=3) and lower baudrates. For baudrates at and below SYSFREQ/104 (n=13), the 4% tolerance limit is satisfied automatically. But tolerance to errors in the automatic baudrate detection process is better for baudrates below SYSFREQ/200. During production tests the AHBUART is operated with n=2 and n=3 and 0% mismatch in baudrate.

For example, at 100MHz SYSFREQ, the maximum baudrate where the tolerance condition is automatically satisfied is 960000 b/s.

48.4 Registers

The core is programmed through registers mapped into APB address space.

Table 669. AHB UART registers

APB address offset	Register
0x4	AHB UART status register
0x8	AHB UART control register
0xC	AHB UART scaler register

48.4.1 AHB UART control register

*Table 670.*0x08 - CTRL - AHB UART control register

31 … 2	1	2
RESERVED	BL	EN
0	0	0
r	r	rw

0: Receiver enable (EN) - if set, enables both the transmitter and receiver. Reset value: ‘0’.

1: Baud rate locked (BL) - is automatically set when the baud rate is locked. Reset value: ‘0’.

48.4.2 AHB UART status register

*Table 671.*0x04 - STAT - AHB UART status register

31 … 10	9 … 8	7	6	5	4	3	2	1	0
RESERVED	TCNT	RX	FE	R	OV	BR	TH	TS	DR
0	0	*	0	0	0	0	1	1	0
r	r	r	rw	r	rw	rw	r	r	r

0: Data ready (DR) - indicates that new data has been received by the AMBA AHB master interface. Read only. Reset value: ‘0’.

1: Transmitter shift register empty (TS) - indicates that the transmitter shift register is empty. Read only. Reset value: ‘1’

2: Transmitter hold register empty (TH) - indicates that the transmitter hold register is empty. Read only. Reset value: ‘1

3: Break (BR) - indicates that a BREAKE has been received. Reset value: ‘0’

4: Overflow (OV) - indicates that one or more character have been lost due to receiver overflow. Reset value: ‘0’

6: Frame error (FE) - indicates that a framing error was detected. Reset value: ‘0’

7: Input state (RX) - Filtered input state. Reset value depends on the input state.

9: 8 Counter State (TCNT) - Internal Counter state

48.4.3 AHB UART scaler register

*Table 672.*0x0C - SCALER - AHB UART scaler register

31 … 18	17 … 0
RESERVED	SCALER RELOAD VALUE
0	0x3FFFB
r	rw

17: 0 Baudrate scaler reload value = (((system_clk*10)/(baudrate*8))-5)/10. Reset value: “3FFFF“.

49 AHB Status Registers

The GR716B microcontroller have 2 separate AHB Status Register units (AHBSTAT).

The 2 separate AHB Status Register units AHBSTAT0 and AHBSTAT1 monitors the DMA bus and the system bus respectively for accesses triggering an error response.

Each AHB Status Register unit (AHBSTATx) have a unique AMBA address described in chapter 2.10 for configuration and status.

The AHB Status Register units are located on APB bus in the address range from 0x8000A000 to 0x8000AFFF and 0x80306000 to 0x80306FFF.

See units connections in the next drawing. The drawing picture memory locations and functions used for configuration and control.

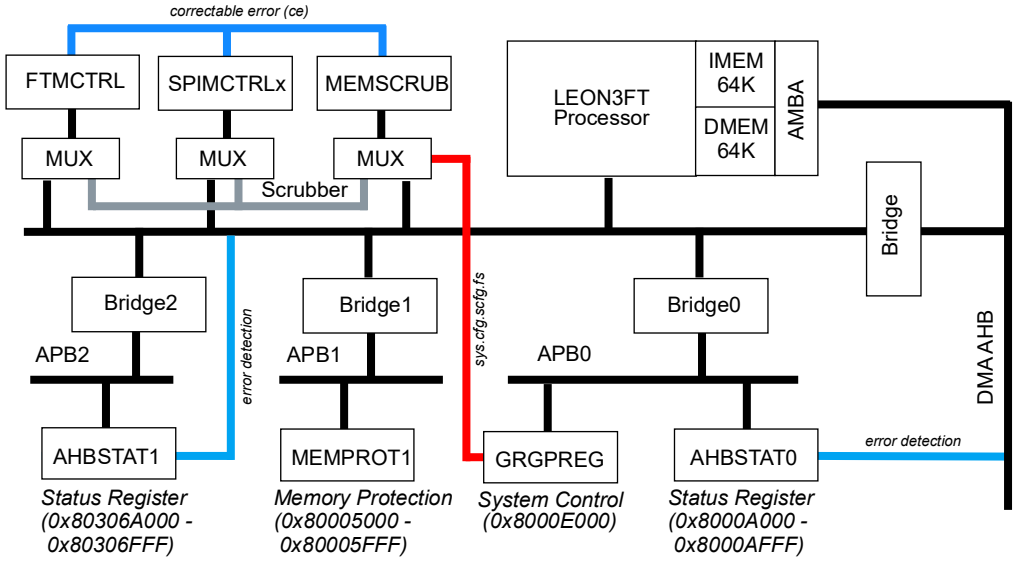


Figure 113. GR716B Status Register bus connection

AHBSTAT unit 0 and 1 has identical configuration and status register. Configuration and status registers are describe in this section 49.3.

System can be configured to protect and restrict access to individual AHBSTAT units in the **MEMPROT** unit. See section 47 for more information.

49.1 Overview

The status registers store information about AMBA AHB accesses triggering an error response. There is a status register and a failing address register capturing the control and address signal values of a failing AMBA bus transaction, or the occurrence of a correctable error being signaled from a fault tolerant core.

49.2 Operation

49.2.1 Errors

The registers monitor AMBA AHB bus transactions and store the current HADDR, HWRITE, HMASTER and HSIZE internally. The monitoring are always active after startup and reset until an error response (HRESP = “01”) is detected. When the error is detected, the status and address register

contents are frozen and the New Error (NE) bit is set to one. At the same time an interrupt is generated, as described hereunder.

Note that many of the fault tolerant units containing EDAC signal an un-correctable error as an AMBA error response, so that it can be detected by the processor as described above.

49.2.2 Correctable errors

Not only error responses on the AHB bus can be detected. Many of the fault tolerant units containing EDAC have a correctable error signal which is asserted each time a correctable error is detected. When such an error is detected, the effect will be the same as for an AHB error response. The only difference is that the Correctable Error (CE) bit in the status register is set to one when a correctable error is detected.

When the CE bit is set the interrupt routine can acquire the address containing the correctable error from the failing address register and correct it. When it is finished it resets the NE bit and the monitoring becomes active again. Interrupt handling is described in detail hereunder.

The AHBSTAT0 records correctable errors from the On-Chip Memory (OCM) described in Section 21; such errors do not originate from AHB accesses. The OCM can be accessed by the CPU and by its internal scrubber, which may detect errors in the memory and flag them to AHBSTAT0. When such an event occurs, the status and failing registers are frozen and do not reflect the OCM addresses where the event occurred. The OCM scrubber should be monitored separately to handle such events. Event signals from two (one from Data memory and one from Instruction memory) out of the eight memory slices are connected to AHBSTAT0. Thus, only 32 KiB (i.e., 1/4 of the 128 KiB memory) contributes to the error reporting in AHBSTAT0.

49.2.3 Interrupts

The interrupt is connected to the interrupt controller to inform the processor of the error condition. The normal procedure is that an interrupt routine handles the error with the aid of the information in the status registers. When it is finished it resets the NE bit and the monitoring becomes active again. Interrupts are generated for both AMBA error responses and correctable errors as described above.

49.2.4 Filtering and multiple error detection

The status register can optionally be implemented with two sets of status and failing address register. In this case the core also supports filtering on errors and has a status bit that gets set in case additional errors are detected when the New Error (NE) bit is set. The core will only react to the first error in a burst operation. After the first error has been detected, monitoring of the burst is suspended.

An error event will only be recorded by the first status register that should react based on filter settings. If register set 1 has reacted then register 2 will not be set for the same error event. In the case where an error occurs and all register sets that could have recorded an error, based on filter settings, already have the NE bit set then the ME bit will be set for all the matching register sets.

49.3 Registers

The core is programmed through registers mapped into APB address space.

*Table 673.*AHB Status registers

APB address offset	Registers
AHB Status Register unit #1 (AHBSTAT1)	
0x8000A000	AHB Status register
0x8000A004	AHB Failing address register
0x8000A008	AHB Status register 2
0x8000A00C	AHB Failing Address register 2
AHB Status Register unit #1 (AHBSTAT2)	
0x80306000	AHB Status register
0x80306004	AHB Failing address register
0x80306008	AHB Status register 2
0x8030600C	AHB Failing Address register 2

49.3.1 AHB Status register

Table 674. 0x00, 0x08- AHBS - AHB Status register

31 14	13	12	11	10	9	8	7	6 3	2 0
RESERVED	ME	FW	CF	AF	CE	NE	HWRITE	HMASTER	HSIZE
0	0	0	0	0	0	0	NR	NR	NR
r	rw*	w*	rw*	rw*	rw	rw	r	r	r

31: 14 RESERVED

13 Multiple Error detection (ME) - This field is set to 1 when the New Error bit is set and one more error is detected. Filtering is considered when setting the ME bit.

This field is only available in version 1 of the core (version is selected at implementation).

12 Filter Write (FW) - This bit needs to be set to '1' during a write operation for CF and AF fields to be updated in the same write operation. Always reads as zero.

This field is only available in version 1 of the core (version is selected at implementation).

11 Correctable Error Filter (CF) - If this bit is set to 1 then this status register will ignore correctable errors. This field will only be written if the FW bit is set.

This field is only available in version 1 of the core (version is selected at implementation).

10 AMBA ERROR Filter (AF) - If this bit is set to 1 then this status register will ignore AMBA ERROR. This field will only be written if the FW bit is set.

This field is only available in version 1 of the core (version is selected at implementation).

9 Correctable Error (CE) - Set if the detected error was caused by a correctable error and zero otherwise.

8 New Error (NE) - Deasserted at start-up and after reset. Asserted when an error is detected. Reset by writing a zero to it.

7 The HWRITE signal of the AHB transaction that caused the error.

6: 3 The HMASTER signal of the AHB transaction that caused the error.

2: 0 The HSIZE signal of the AHB transaction that caused the error

49.3.2 AHB Failing address register

*Table 675.*0x04, 0x0C - AHBFAR - AHB Failing address register

31 0
AHB FAILING ADDRESS
NR
t

31: 0 The HADDR of the AHB transaction that caused the error.

50 Trace buffer

The GR716B microcontroller have 2 separate AMBA trace buffer (AHBTRACE) units. The AHB-TRACE units described in this section have the capability to trace all transactions on the main AHB bus and Debug AHB bus.

The first AMBA trace buffer (AHBTRACE0) is tracing the Main AHB bus and the second AMBA trace buffer (AHBTRACE1) is tracing the DMA AMBA bus. Each AMBA trace buffer (AHB-TRACE) unit have a unique AMBA address described in chapter 2.10 for configuration and status.

The control and status register for the AMBA trace buffer units are located on AHB bus in the address range from 0x90000000 to 0x907FFFFF and in the range from 0x9FF20000 to 0x9FF3FFFF. See AMBA memory protection units connections in the next drawing. The drawing picture memory locations and functions used for AMBA memory protection units configuration and control.

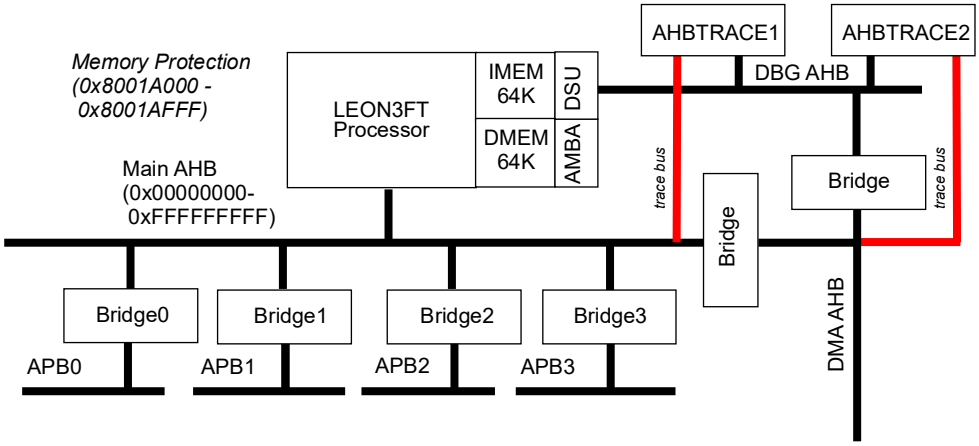


Figure 114. GR716B AHBTRACEx bus and pin connection

AHBTRACE unit 0 and 1 has identical configuration and status register. Configuration and status registers for AHBTRACE1 are describe in this section 50.4 and register for AHBTRACE0 is embedded into DSU3 described in section 20.7.

50.1 Overview

The trace buffer consists of a circular buffer that stores AMBA AHB data transfers. The user can select to trace the main bus, DMA bus, Debug bus or Scrubber bus. The address, data and various control signals of the selected AHB bus are stored and can be read out for later analysis.

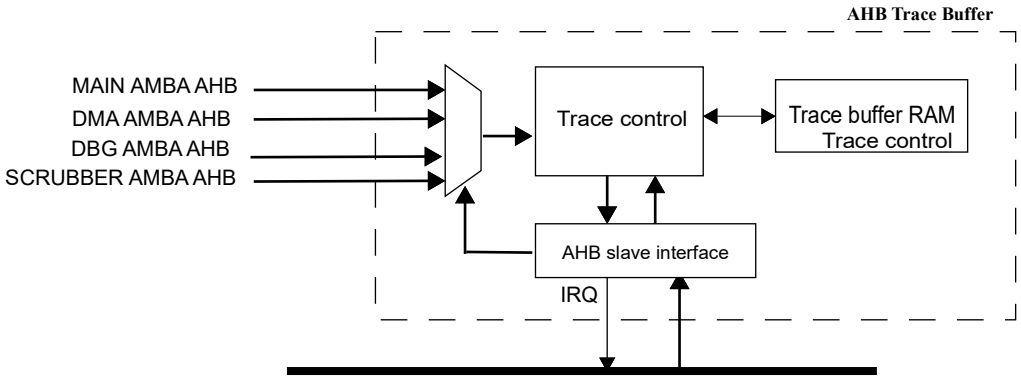


Figure 115. Block diagram

When the trace buffer is configured to store the information as indicated in the table below:

*Table 676.*AHB Trace buffer data allocation

Bits	Name	Definition
127:96	Time tag	The value of the time tag counter
95	AHB breakpoint hit	Set to '1' if a DSU AHB breakpoint hit occurred.
94:80	-	Not used
79	Hwrite	AHB HWRITE
78:77	Htrans	AHB HTRANS
76:74	Hsize	AHB HSIZE
73:71	Hburst	AHB HBURST
70:67	Hmaster	AHB HMASTER
66	Hmastlock	AHB HMASTLOCK
65:64	Hresp	AHB HRESP
63:32	Load/Store data	AHB HRDATA[31:0] or HWDATA[31:0]
31:0	Load/Store address	AHB HADDR

In addition to the AHB signals, a 32-bit counter is also stored in the trace as time tag.

50.2 Operation

50.2.1 Overview

The trace buffer is enabled by setting the enable bit (EN) in the trace control register. Each AMBA AHB transfer is then stored in the buffer in a circular manner. The address to which the next transfer is written is held in the trace buffer index register, and is automatically incremented after each transfer. Tracing is stopped when the EN bit is reset, or when a AHB breakpoint is hit. An interrupt is generated when a breakpoint is hit.

50.2.2 AHB statistics

The trace module generates statistics from the traced AHB bus. Statistics is collected and output to LEON statistics unit (L3STAT). The statistical outputs can be filtered by the AHB trace buffer filters, this is controlled by the Performance counter Filter bit (PF) in the AHB trace buffer control register. The core can collect data for the events listed in table 677 below.

*Table 677.*AHB events

Event	Description	Note
idle	HTRANS=IDLE	Active when HTRANS IDLE is driven on the AHB slave inputs and slave has asserted HREADY.
busy	HTRANS=BUSY	Active when HTRANS BUSY is driven on the AHB slave inputs and slave has asserted HREADY.
nseq	HTRANS=NONSEQ	Active when HTRANS NONSEQ is driven on the AHB slave inputs and slave has asserted HREADY.
seq	HTRANS=SEQ	Active when HTRANS SEQUENTIAL is driven on the AHB slave inputs and slave has asserted HREADY.
read	Read access	Active when HTRANS is SEQUENTIAL or NON-SEQUENTIAL, slave has asserted HREADY and the HWRITE input is low.
write	Write access	Active when HTRANS is SEQUENTIAL or NON-SEQUENTIAL, slave has asserted HREADY and the HWRITE input is high.

*Table 677.*AHB events

Event	Description	Note
hsize[5:0]	Transfer size	Active when HTRANS is SEQUENTIAL or NON-SEQUENTIAL, slave has asserted HREADY and HSIZE is BYTE (hsize[0]), HWORD (HSIZE[1]), WORD (hsize[2]), DWORD (hsize[3]), 4WORD hsize[4], or 8WORD (hsize[5]).
ws	Wait state	Active when HREADY input to AHB slaves is low and AMBA response is OKAY.
retry	RETRY response	Active when master receives RETRY response
split	SPLIT response	Active when master receives SPLIT response
spdel	SPLIT delay	Active during the time a master waits to be granted access to the bus after reception of a SPLIT response. The core will only keep track of one master at a time. This means that when a SPLIT response is detected, the core will save the master index. This event will then be active until the same master is re-allowed into bus arbitration and is granted access to the bus. This also means that the delay measured will include the time for re-arbitration, delays from other ongoing transfers and delays resulting from other masters being granted access to the bus before the SPLIT:ed master is granted again after receiving SPLIT complete. If another master receives a SPLIT response while this event is active, the SPLIT delay for the second master will not be measured.
locked	Locked access	Active while the HMASTLOCK signal is asserted on the AHB slave inputs. (Currently not used by L3STATand L4STAT)

50.3 Using the AHB trace buffer

The debug monitor GRMON3 has build-in support for using AHB trace buffer. For more information see chapter for using the trace buffer in the GRMON3 User's Manual [GRMON3].

50.4 Registers

50.4.1 Register address map

The trace buffer occupies 128 KiB of address space in the AHB I/O area.The following register addresses are decoded:

Table 678. Trace buffer address space

Address	Register
0x000000	Trace buffer control register
0x000004	Trace buffer index register
0x000008	Time tag counter
0x00000C	Trace buffer master/slave filter register
0x000010	AHB break address 1
0x000014	AHB mask 1
0x000018	AHB break address 2
0x00001C	AHB mask 2
0x010000 - 0x020000	Trace buffer
...0	Trace bits 127 - 96
...4	Trace bits 95 - 64
...8	Trace bits 63 - 32
...C	Trace bits 31 - 0

50.4.2 Trace buffer control register

The trace buffer is controlled by the trace buffer control register:

Table 679. 0x000000 - CTRL - Trace buffer control register

31 23	22 16	15	14 12	11 9	8	7 6	5	4	3	2	1	0
RESERVED	DCNT	BA	BSEL	RESERVED	PF	BW	RF	AF	FR	FW	DM	EN
0	0	1	0	0	0	*	0	0	0	0	0	*
r	rw	r	rw	r	rw	r	rw	rw	rw	rw	r	rw

31: 23 RESERVED

22: 16 Trace buffer delay counter (DCNT)

15 Bus select Available (BA) - Set to ‘1’ to indicate that the core has several buses connected. The bus to trace is selected via the BSEL field.

14: 12 Bus select (BSEL)
"000" - Main AHB Bus
"001" - DMA AHB bus
"010" - Debug AHB bus
"011" - Scrubber AHB bus

11: 9 RESERVED

8 Performance counter Filter (PF) - If this bit is set to ‘1’, the cores performance counter (statistical) outputs will be filtered using the same filter settings as used for the trace buffer. If a filter inhibits a write to the trace buffer, setting this bit to ‘1’ will cause the same filter setting to inhibit the pulse on the statistical output.

7: 6 Bus width (BW) - This value corresponds to log2(Supported bus width / 32)

5 Retry filter (RF) - If this bit is set to ‘1’, AHB retry responses will not be included in the trace buffer.

4 Address Filter (AF) - If this bit is set to ‘1’, only the address range defined by AHB trace buffer breakpoint 2’s address and mask will be included in the trace buffer.

3 Filter Reads (FR) - If this bit is set to ‘1’, read accesses will not be included in the trace buffer.

2 Filter Writes (FW) - If this bit is set to ‘1’, write accesses will not be included in the trace buffer.

Table 679. 0x000000 - CTRL - Trace buffer control register

1	Delay counter mode (DM) - Indicates that the trace buffer is in delay counter mode.
0	Trace enable (EN) - Enables the trace buffer

50.4.3 Trace buffer index register

The trace buffer index register indicates the address of the next 128-bit line to be written.

Table 680. 0x000004 - INDEX - Trace buffer index register

31 11	10 4	3 0
RESERVED	INDEX	0x0
0	NR	0
r	rw	r

31: 11	RESERVED
10: 4	Trace buffer index counter (INDEX).
3: 0	Read as 0x0

50.4.4 Trace buffer time tag register

The time tag register contains a 32-bit counter that increments each clock when the trace buffer is enabled. The value of the counter is stored in the trace to provide a time tag.

Table 681. 0x000008 - TIMETAG - Trace buffer time tag counter

31 0
TIME TAG VALUE
0
r

50.4.5 Trace buffer master/slave filter register

The master/slave filter register allows filtering out specified master and slaves from the trace.

Table 682. Trace buffer master/slave filter register

31 16	15 0
SMASK[15:0]	MMASK[15:0]
0	0
rw	rw

31: 16	Slave Mask (SMASK) - If SMASK[n] is set to ‘1’, the trace buffer will not save accesses performed to slave n.
15: 0	Master Mask (MMASK) - If MMASK[n] is set to ‘1’, the trace buffer will not save accesses performed by master n.

50.4.6 Trace buffer breakpoint registers

The DSU contains two breakpoint registers for matching AHB addresses. A breakpoint hit is used to freeze the trace buffer by clearing the enable bit. Freezing can be delayed by programming the DCNT field in the trace buffer control register to a non-zero value. In this case, the DCNT value will be decremented for each additional trace until it reaches zero and after two additional entries, the trace buffer is frozen. A mask register is associated with each breakpoint, allowing breaking on a block of addresses. Only address bits with the corresponding mask bit set to '1' are compared during breakpoint detection. To break on AHB load or store accesses, the LD and/or ST bits should be set.

Table 683. Trace buffer AHB breakpoint address register

31	2	1	0
BADDR[31:2]		0b00	
NR		0	
rw		r	

31: 2 Breakpoint address (BADDR) - Bits 31:2 of breakpoint address

1: 0 Reserved, read as 0

Table 684. Trace buffer AHB breakpoint mask register

31	2	1	0
BMASK[31:2]		LD	ST
NR		0	0
rw		rw	rw

31: 2 Breakpoint mask (BMASK) - Bits 31:2 of breakpoint mask

1 Load (LD) - Break on data load address

0 Store (ST) - Break on data store address

51 Boot ROM

51.1 Overview

The GR716B microcontroller contains a on-chip Boot ROM for low-level initialization and optional self-testing, standby and application loading. The Boot ROM contains instructions executed by the CPU. The Boot ROM may be configured via bootstraps signals controlled by the user at reset.

The Boot ROM prepares an execution environment suitable for an Application Software (ASW) to take over the system. This reduces the system initialization normally required by the application to perform.

It is possible to disable the Boot ROM via bootstraps signals, see section 7.1. When the Boot ROM is disabled, the reset address is determined by bootstrap signals.

51.2 ROM Architecture

51.2.1 Logical division of Boot ROM

This section describes the top-level structure of the Boot ROM. The design is logically divided in three main parts:

- Processor module initialization
- Standby mode
- Load sequence

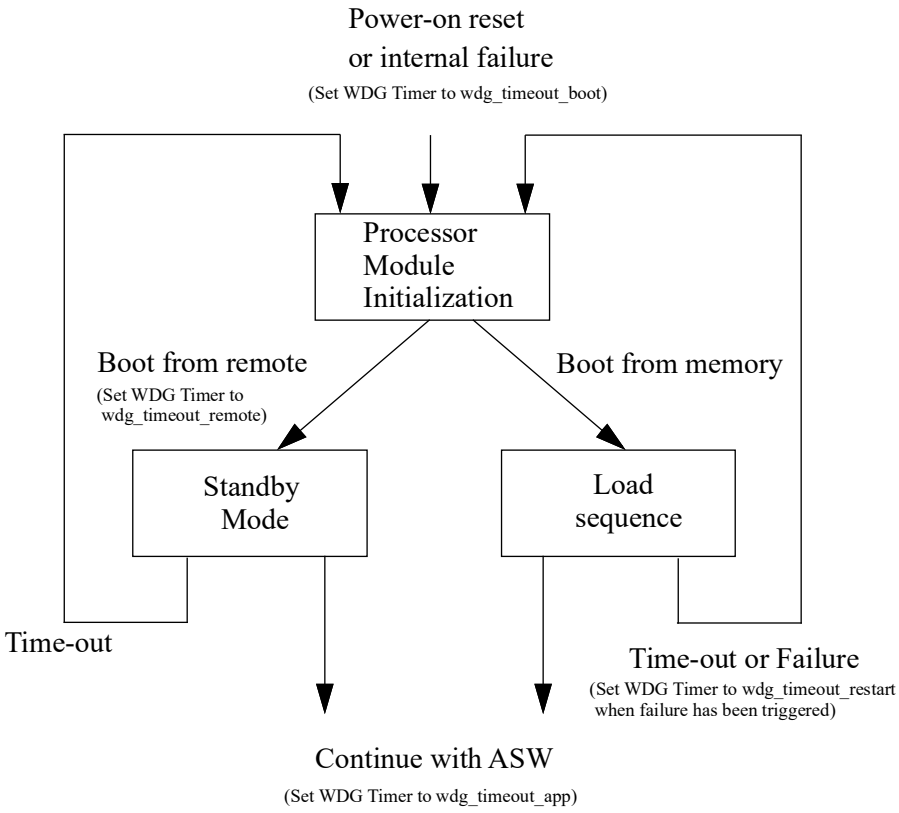


Figure 116. Boot ROM to-level architecture

The *Processor Module Initialization* sequence is triggered by reset condition. It is responsible for initializing and self-testing of on-chip instruction and data memory, writing the boot report. Processor module initialization has a mostly linear execution path, meaning that a minimum number of branch decisions which depend on system external factors are made. Initialization sequence is directly followed by the standby mode or load sequence.

The *Standby mode* is entered when the GR716B microcontroller is configured via bootstraps to be configured remotely via external interface, for example SpW, SPI, UART, etc. The watchdog timer is initialized.

The *Loading sequence* is responsible for validating, loading and executing an ASW image residing in application storage memory (ASM) or RAM. After the ASW load the ASW will be executed.

51.2.2 Properties of the Boot ROM

The table below gives an overview of certain properties of the Boot ROM. Some parameters are given in seconds based on a 50MHz system clock, however they must be scaler to the actual system clock to get the correct number of seconds.

*Table 685.*Run-time properties

Name	Value	Description
gptimer_prescaler	65535	The Boot ROM does not write to this register and assumes that it has the reset value for GPTIMER prescaler reload. The wdg_timeout_boot, wdg_timeout_restart, wdg_timeout_app and wdg_timeout_remote values are programmed by the Boot ROM. The watchdog timer reload value in GPTIMER is programmed by the Boot ROM using the formula (wdgX * 763 +1). Here, the 763 is the reload value required to get 1 second based on 50 MHz clock frequency.
wdg_timeout_boot	300	Timeout, in seconds, for the watchdog timer during start of internal boot. (At 50MHz system clock)
wdg_timeout_restart	1	Timeout, in seconds, for the watchdog timer when an internal error has occurred. (At 50MHz system clock)
wdg_timeout_app	600	Timeout, in seconds, set prior to application hand over from Boot ROM. (At 50MHz system clock)
wdg_timeout_remote	5*3600	Timeout, in seconds, waiting for remote access. (At 50MHz system clock)
i2c_prescaler	-	Precscaler set for 100KHz transfer for 50MHz system clock
i2c_deviceid	0x50	I2C device ID is locked to 0x50
mcfg1, mcfg2, mcfg3 - Initialization value for external memory controller		
mcfg1_r_ws	0xF	max read PROM wait states
mcfg1_bsize	0x6	Prom banks size of 512KiB
mcfg2_r_ws	0x4	max read SRAM wait states
mcfg2_bsize	0x6	Sram banks size of 512KiB
%psr, %wim and %tbr settings for applications and remote boot mode		
PSR_SUPER	0x1	Enable Supervisor mode
PSR_CWP	0x0	Default CWP
PSR_PIL	0x0	Processor interrupt level for application and remote access
PIL_EF	0x0	FPU disabled
fp_start	0x3000FF00	Frame pointer
sp_start	0x3000FEA6	Stack pointer
boot_image_header	0x3000FF48	Boot image header start
boot_report	0x3000FFF8	Boot report start

51.2.3 Clock configuration

This section specifies the clocks registers used and enabled after boot depending upon the boostraps.

Table 686: Clock configuration

Mode	PLL Config	Clock Unit	Description
External SPI ROM	N/a	CLKEN0.S0 CLKEN0.S1 [1)]	Boot from external SPI PROM with or without redundant source
External SRAM		CLKEN0.MC	Boot from external SRAM with or without redundant source
External ROM/PROM		CLKEN0.MC	Boot from external SRAM with or without redundant source
Dedicated memory interface (Only for SIP/ PBGA devices)		CLKEN0.NV	Boot from external NVRAM without redundant source
SpaceWire	PLLREF.CFG 3)	CLKEN1.SP	Remote boot via SpaceWire
SPI2AHB		CLKEN0.SA	Remote boot via SPI
CAN-FD		CLKEN1.C0	Remote boot via CANOpen
UART		CLKEN0.AU	Remote boot via UART

Note 1: Only enabled if boot from primary option fails

Note 2: Table only specify bits enabled by boot. All other bits will remain as is after reset.

Note 3: SpaceWire PLL option is configured by is determined by GPIO[63] and DUART_TX. For more information see table 22 in section 3.1.

51.2.4 Memory Layout

The Boot PROM usage of the memory resources in table 687.

*Table 687.*Boot software usage of the memory resources

Memory	**Properties**	**Context**
Boot memory	Read-only	Boot software executable and constant data
On-chip RAM	Volatile R/W	Boot software volatile runtime data and stack. Used by processor module initialization, load sequence and standby mode. Boot report will be written into the on-chip RAM above the stack.
External SPI Memory (SPIMCTRL)	Non-Volatile R/W	Application software to be copied into the on-chip RAM and executed from on-chip RAM (Application software can be available at two different areas for dual module redundancy check)
External PROM/SRAM (FTMCTRL0)	Non-Volatile R/W Volatile R/W	Application software, runtime and stack.
Dedicated memory interface (FTMCTRL1) Only available for SIP/PBGA devices	Non-Volatile R/W Volatile R/W	Application software, runtime and stack.

51.2.5 Boot report

Boot report entries are written by the Boot SW. The boot report in combination with the image header located in the local data ram contains the following information:

- Boot software version
- Internal Instruction and Data RAM test status
- ASW loaded and integrity status
- Start of execution (entry point)
- Application software id

The total size of the boot report allocated in the data memory is 8 bytes. The boot address always allocates the last 2 words in the local data memory. The boot report format:

Table 688. 0x30000FF8 - BOOTRPT0 - Boot Rom Report

31 10	9 7	6 5	4 2	1	0
Reserved	CPY	EXT	RMT	I	D
0	*	*	*	0	0
rw	rw	rw	rw	rw	rw

31: 10 Reserved.

9: 7 ASW source copy (CPY) - Status field for ASW source selected:

0x0 - None
0x1 - SRAM chip select 0 and 1
0x2 - PROM chip select 0 and 1
0x3 - SPI memory using dedicated external SPI memory interface
0x4 - SPI redundant memory interface

6: 5 External boot source (EXT) - Status field for external boot source:

0x0 - None
0x1 - External SRAM/MRAM using SRAM chip select 0
0x2 - External PROM using PROM chip select 0
0x3 - External SPI memory

4: 2 Remote access (RMT) - Status field for remote access:

0x0 - None
0x1 - SpaceWire remote access enabled
0x2 - UART remote access enabled
0x3 - CAN-FD
0x4 - SPI remote access enabled

1 Reserved

0 Reserved

Table 689. 0x30000FFC - BOOTRPT1 - Boot Rom Report

31 0
Reserved
0
rw

31: 0 Reserved

In case of using the ASW container the application can read and check the latest ASW header located in the data memory on address specified in table 685 and ASW header format in table 690.

51.2.6 Application software image format

Application software image files are located in ASM from where they are loaded by the Boot software Application loader. An image consists of an image header, a number of image section headers, and a variable number of data blocks. All addresses (entry point and data destination addresses) must be aligned to 32-bit words.The sections are of two types: data sections and WMEM sections. A

WMEM section provides a simple way to perform additional system initializations before handing over the system to application software. The format of the regular ASW and WMEM sections are described in Table 692 and Table 693 .

*Table 690.*Image header

Field	Type	Description
id	32-bit word	User defined section. The values does not affect Boot software execution.
ep	32-bit word	Application software entry point
stp	32-bit word	Application stack pointer
sections 0	image section header	Section layout defined in table below.
sections [1 ..15]	image section header	Section layout defined in table below.
cksum	16 -bit word	16-bit ISO checksum as defined in [ECSS-E7041], annex A. Calculated over all fields of the image header.
Reserved	16-bit word	Reserved, set to zero

*Table 691.*Image section header

Field	Type	Description
flags	32-bit word	Bit 0: ENABLE - This section shall be copied to RAM. Bit 1: RESERVED- Must be 0 Bit 2: WMEM : 0 -> This section is a regular section 1 -> This section is a WMEM section Bit 31..3: RESERVED - Must be 0
source	32-bit word	Location of source data block, expressed as number of 32-bit words relative to image header. (Note: Offset is not relative to section header.)
dest	32-bit word	Absolute destination address (As this filed is not used by WMEM sections, it can be set to 0 for WMEM headers)
length	32-bit word	Length of section data block in 32-bit words.
datacksum	16-bit word	Data block checksum over data[0]..data[length-1]. 16-bit ISO checksum as defined in [ECSS-E7041], annex A.
Reserved	16-bit word	Reserved

*Table 692.*Image data block for a regular section

Field	Type	Description
data[0]	32-bit word	Data word 0
data[1]	32-bit word	Data word 1
...	...	...
data[length-1]	32-bit word	Data word length-1

*Table 693.*Image data block for a WMEM section

Field	Type	Description
Address[0]	32-bit word	First address to be written
data[0]	32-bit word	Data word to be written at address[0]
Address[1]	32-bit word	Second address to be written
data[1]	32-bit word	Data word to be written at address[1]
...	...	...
data[length-1]	32-bit word	Data word length-1

When the application software is loaded the image header read is always stored in top of memory. The application software can determine its origin by reading the stored header in the local data memory. The image header start is always set to *'boot_image_header'* from the end of the last address in the local data memory.

51.3 Loader description

Load sequence is the summing-up of Boot software components executing after standby mode. The component includes enable clocks and pins, image loading and application software boot.

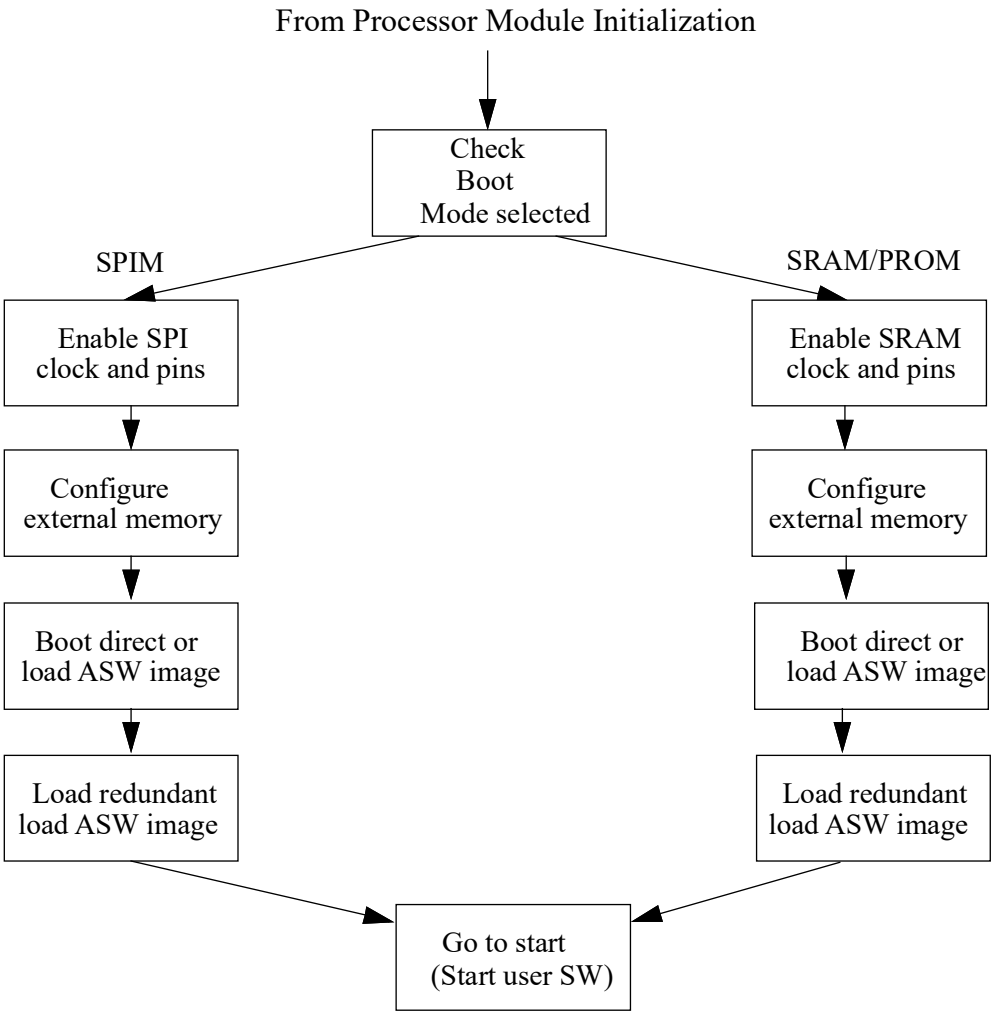


Figure 117. Load sequence detailed description

For more information see Table 685.

51.4 Standby description

After processor initialization has completed the Standby mode is started if selected by the bootstrap signals. The bootstrap configures which interface is used for the remote control. For each of the remote access options the boot software enable interface clock and pins. See sections below for more information about which pins are used for respective interface.

Full access is granted to the unit accessing the GR716B Microcontroller via the selected remote control and access interface. When remote control unit upload new software to the GR716B Microcontroller the remote unit can reset and re-start the processor via the interrupt controller unit, see chapter 40 for more information. The remote unit needs to instruct the GR716B Microcontroller to restart and start executing the uploaded software.

While in standby mode the processor is in power-down mode. The watchdog timer is activated during Standby mode.

51.4.1 CANFD (CANOpen)

A remote host can access the AHB DMA bus via the CAN interface described in chapter 26.

Information on which pins used to connect to the external CAN bus is described in Table 23.

51.4.2 SpaceWire - RMAP

A remote host can access the AHB DMA bus via the SpaceWire RMAP interface described in chapter 33.

Information on which pins used to connect to the external SpaceWire bus is described in Table 23.

51.4.3 SPI2AHB

A remote host can access the AHB DMA bus via the SPI2AHB interface described in chapter 43.

Information on which pins used to connect to the external SPI bus is described in Table 23.

51.4.4 UART

There are two AHBUART interfaces on the GR716B, only the AHBUART connected directly to the AHB DMA bus is used for the remote control in Standby mode.

A remote host can access the AHB DMA bus via the AHBUART interface described in chapter 48.

Information on which pins used to connect to the external UART bus is described in Table 23.

51.5 State at handover to application software

After the boot sequence has finished the following general state applies:

- GPTIMER prescaler is set to gptimer_prescaler
- Watchdog timer value is set to wdg_timeout_app or wdg_timeout_reomte and is kicked in less than 100 clock cycles from entry to ASW.
- Boot report is located at address boot_report
- ASW header is located at the address boot_image_header.
- All interfaces are disabled except for the interface requested to be enabled via bootstraps
- The clock-gate unit is configured according to the bootstrap configuration
- Frame pointer (register %fp) is set to the start before the boot report

- Stack pointer (register %sp) is set to frame pointer minus 96.
- PSR.S=1, PSR.ET=0, PSR.CWP=0, PSR.EF=0
- WIM=0x2
- Single vector trapping is disabled

For more information see Table 685.

51.6 Boot source requirements

This chapter specifies limitations to external interface or components during boot.

51.6.1 SpaceWire Remote access

Remote access via SpaceWire (without software support) requires the SpaceWire input frequency to be 10 MHz, 20 MHz, 25 MHz or 50 MHz.

51.6.2 External SPI Memory

External SPI memories will be clocked with system clock divided by 8 during the boot sequence. The external SPI memory is assumed to support read command 0x3 and have 3 or 4 address bytes as default.

51.6.3 External PROM/SRAM memory (FTMCTRL0)

The access time for external PROM shall be equal or less than 16 system clock cycles and for external SRAM the access time shall be equal or less than 4 system clock cycles.

51.6.4 Dedicated Memory Interface (FTMCTRL1)

Booting using the second memory controller is also supported in SIP/PBGA version of the device. The access time for external PROM shall be equal or less than 16 system clock cycles and for external SRAM/MRAM the access time shall be equal or less than 4 system clock cycles.

51.7 Protection schemes

The boot image has a number of protection schemes build in to check for erroneous boot software, hardware malfunction:

SRAM/PROM:

- BCH EDAC protection
- Scrubber (SRAM only)
- ASW load image protection with CRC protection
- Redundant ASW load image
- Watchdog timer

SPI Memory:

- BCH EDAC protection
- ASW load image protection
- Redundant ASW load image with CRC protection
- Watchdog timer

Remote Access:

- Watchdog timer

Internal Memories startup test (Instruction, Data and Register Window):

- Scrubber (Instruction and Data)

Boot ROM startup and access:

- Protection of unwanted access or start of boot due to erroneous trap handler
- Watchdog timer

51.7.1 BCH EDAC Protection

When the memory controller with EDAC enabled (SPIMCTRL/FTMCTRL) detects a correctable error, the data will be temporarily corrected and delivered onto the on-chip bus.

When using FTMCTRL with EDAC enabled for boot, an uncorrectable error detected during loading of an ASW will cause the boot ROM software to enter an error state, and the redundant boot source will not be attempted. Therefore, when the redundant memory boot option is used, it is recommended to disable memory EDAC when booting via FTMCTRL.

When using SPIMCTRL with EDAC enabled for boot, an uncorrectable error detected during loading of an ASW will cause the boot ROM software either to wait for a re-boot or to load a redundant ASW image. In this case, EDAC can remain enabled even when redundant memory is used. EDAC can be enabled without enabling error response in SPIMCTRL, which allows the redundant memory to be attempted.

51.7.2 Hardware Scrubber

The scrubber is used by the Boot ROM to clear internal instruction and data ram.

51.7.3 ASW load image

Application software load image correctness can be checked using 16-bit cyclic redundant code defined in [ECSS-E7041]. The load image also provides features to automatically copy instruction and data sections before executing ASW.

51.7.4 Redundant ASW load image

All externally memory boot options have the capability to boot from redundant memory in case of bad CRC is detected on first boot image. An error on the second will force the GR716B microcontroller to reboot and retry the first image.

PROM primary boot uses external ROM chip select signal 0 and redundant PROM uses ROM chip signal 1.

SRAM primary boot uses external ram chip select signal 0 and redundant SRAM uses RAM chip signal 1.

SPI memory boot automatically selects SPI memory controller 0 as primary boot and SPI memory controller 1 as redundant boot option.

51.7.5 Watchdog Timer

A watchdog timer will be enabled to detect erroneous behavior during boot sequence. The Watchdog Timer is reloaded at reset and during start of the boot ROM and at the end of boot ROM to make sure the ASW or accesses from the external interface can be executed or received.

The watchdog have different settings and reload value depending boot software executed and boot option selected by user.

Watchdog timeout value and state:

- During initialization of boot software the watchdog is set to a value ’*wdg_timeout_boot*’ long enough to cope with memory and register test. The watchdog is always enabled at boot program start because the boot software might have been called from an unwanted trap.
- For application booting from external memory the watchdog is set to a value ’*wdg_timeout_app*’ long enough to cope with starting or loading an application from an external memory.
- For application booting via remote access the watchdog is set to a value ’*wdg_timeout_remote*’. The value is set to a ’extrem’ long value to be able to cope with slow transfers or interfaces. The watchdog timer reload register is accessible via remote interface and in case where software

upload is extremely slow the remote software should be able to extend or reload the watchdog counter by writing to the watchdog timers registers.

- When a failure is encountered during initialization of the microcontroller which can't be resolved the boot software requests a system reboot by setting the watchdog timer to a '*wdg_timeout_restart*'.

52 Real-Time Accelerator (RTA)

52.1 Overview

This chapter describes the Real-Time execution Accelerator (RTA) block integrated in to GR716B.

The RTA module is an independent LEON3FT subsystem and can execute software in parallel with the main LEON3FT processor in GR716B. The RTA subsystem can access 32KiB of tightly coupled memory protected by EDAC to ensure single cycle instruction execution. The RTA implements 2 sets of registers window and support interrupts.

The RTA shares the address map with the main LEON3FT processor in section 2.10.

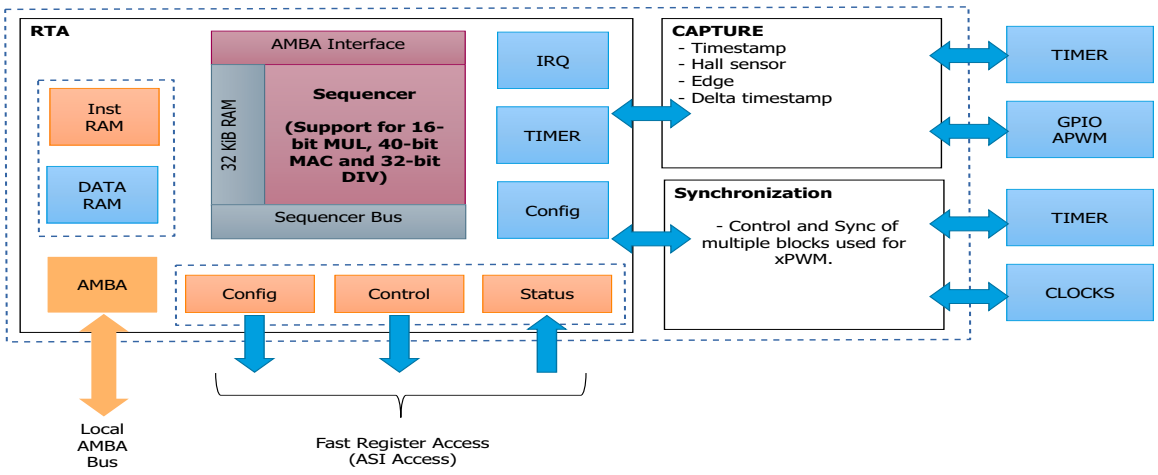


Figure 118. Simplified architecture and functional block diagram of the RTA and connections

The Real-Time-Accelerator (RTA) includes:

- LEON3FT processor core with 2 register windows and support for 16*16-bit multiplier and 40 bit accumulator. FPU is not available in RTA.
- Separate Data (16KiB) and instruction (16KiB) memory for single cycle execution of instructions
- Local System timer (GPTIMER)
- Local IRQMP interrupt controller with start/stop/reset control
- Separate Debug Unit with instruction trace
- RTA Task Manager (RTM) and interrupt time stamp functionality
- AHB/AHB bridge with access control to separate the RTA system from the main GR716B system
- AHB bridge with access to following peripherals (refer table 20 for memory map)
  - UART (all six UART Serial Interface)
  - SPI controller (both controller)
  - LVDS IO
  - General Purpose I/O port 0 to 31
  - General Purpose I/O port 32 to 64

- I2C-master (both controller)
- ADC (all four interface)
- FPGA Scrubber
- DAC (all four interface)
- I2C-slave (both controller)
- APWM system timer and synchronization
- APWM function timer and synchronization
- Analog comparator A
- Analog comparator B
- Protection shutdown A, B and C
- External synchronization
- PWM modulator DAC A, B, C and D
- Clock error detection (both 0 and 1 units)
- APWM A0 Timer and synchronization
- FIR filter including clock and synchronization unit
- APWM Timestamp
- All APWMAB, APWMCF and APWMG controllers

Each RTA have separately clock and reset control possibility. See section 28.

The software execution is started and controlled via the local interrupt controller. See section 40.2.7 on dynamically controlling software execution.

52.2 RTA Status and mailbox Register

RTA status and mailbox register for simple inter-task communication from the RTA to the main LEON3FT processor.

Any status bit(s) in the register RTAx.STAT can generate an interrupt to the main processor (Interrupt line 60 for RTA0 and 61 for RTA1). Application needs to set the corresponding level and enable bit in the registers RTAx.STAT.LVL and RTAx.STAT.MASK.

Table 694. RTA status and mailbox register base addresses (TBD)

AMBA address	Register	Acronym
0x62040000 0x72040000	RTAx status and mailbox status register	RTA0.STAT.REG RTA1.STAT.REG
0x62040004 0x72040004	RTAx status and mailbox interrupt level detection configuration register	RTA0.STAT.LVL RTA1.STAT.LVL
0x62040008 0x72040008	RTAx status and mailbox interrupt register	RTA0.STAT.IRQ RTA1.STAT.IRQ
0x6204000C 0x7204000C	RTAx status and mailbox mask register for interrupt generation	RTA0.STAT.MASK RTA1.STAT.MASK

Table 695. 0x62040000 / 0x72040000 - RTAx.STAT - RTA status and mailbox register status register

31 28	27	26	25	24 23 22 21 20 19 18 17 16 15 14 13 12	11	10	9	8	7	6	5	4	3	2	1	0
USR	LW	LI	LR	R	DM	DE	FC	IC	BP	DU	SU	IP	ID	PD	HA	ER
0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
rw	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Table 695. 0x62040000 / 0x72040000 - RTAx.STAT - RTA status and mailbox register status register

31: 28	User defined status bits (USR)
27	Local RTAx Watch Dog Timer Alarm (LW) - Local watchdog has triggered and generated local interrupt
26	Local Interrupt Alarm (LI) - A local interrupt have occurred. see table 699 for interrupt assignments
25	Local RTAx interrupt resume Alarm (LR) - Processor has been resumed
24: 11	Reserved
11	DSU mode (DM)
10	DSU Enable (DE)
9	Fault Counter (FC)
8	Instruction Counter (IC)
7	BP missed (BP)
6	DU Counter (DU)
5	SU state (SU)
4	Interrupt pending (IP)
3	IDLE state (ID)
2	Power down state (PD)
1	Halt (HA)
0	Error (ER)

Note When corresponding bit are set in the RTAx.STAT.LVL and RTAx.STAT.MASK register an interrupt will be generated

Table 696. 0x62040004 / 0x72040004 - RTAx status and mailbox interrupt level detection configuration register

31 28	27	26	25	24 23 22 21 20 19 18 17 16 15 14 13 12	11	10	9	8	7	6	5	4	3	2	1	0
USR	LW	LI	LR	R	DM	DE	FC	IC	BP	DU	SU	IP	ID	PD	HA	ER
0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
rw	rw	rw	rw	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Table 697. 0x62040008 / 0x72040008 - RTAx status and mailbox interrupt register

31 28	27	26	25	24 23 22 21 20 19 18 17 16 15 14 13 12	11	10	9	8	7	6	5	4	3	2	1	0
USR	LW	LI	LR	R	DM	DE	FC	IC	BP	DU	SU	IP	ID	PD	HA	ER
0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
wc	wc	wc	wc	r	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc

Table 698. 0x6204000C / 0x7204000C - RTAx status and mailbox mask register for interrupt generation

31 28	27	26	25	24 23 22 21 20 19 18 17 16 15 14 13 12	11	10	9	8	7	6	5	4	3	2	1	0
USR	LW	LI	LR	R	DM	DE	FC	IC	BP	DU	SU	IP	ID	PW	HA	ER
0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
rw	rw	rw	rw	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

52.3 Interrupts (Local to RTA)

The table below indicates the interrupt assignments. All interrupts are handled by the interrupt controller and forwarded to the RTA LEON3FT processors. The controller along with local timer supports soft watchdog (event creates an interrupt). For more configuration and option see chapter 40. *There are differences between the local interrupt controller and the main interrupt controller, the interrupt remap function is not available (and not necessary) in the RTA local interrupt controller, there is only one soft watchdog timer in the RTA local interrupt controller.*

*Table 699.*Bus Interrupt line assignments.

Interrupt Line	RTA driven interrupt line	Interrupt from main system bus Revision 1	Interrupt from main system bus Revision 0
0	n/a - not used		
1	GPTIMER - Interrupt 1 from the local subtimer0	Interrupt line from the main system bus can trigger interrupt to RTA. Refer table 21 and section 40 for more information.	Interrupt line from the main system bus can trigger interrupt to RTA. Refer table 21 and section 40 for more information.
2	GPTIMER- Interrupt 2 from the local subtimer1		
3	GPTIMER - Interrupt 3 from the local subtimer2		
4	GPTIMER - Interrupt 4 from the local subtimer3		
5	RTM0 - Interrupt from RTA Time Manager 0 (RTM)	Main system bus cannot trigger RTA interrupt line 5, 6 and 7. This is modified in Revision 1 of device.	
6	RTM1- Interrupt from RTA Time Manager 1 (RTM)		
7	RTM2 - Interrupt from RTA Time Manager 2 (RTM)		
8	unused	Interrupt line from the main system bus can trigger interrupt to RTA. Refer table 21 and section 40 for more information.	
9	DLRAM, ILRAM - Interrupt from On-chip RAM connected with RTA		
10	AHBSTAT - Interrupt from local AHB Status unit		
11-31	unused		

Interrupts within an RTA subsystem do not propagate to the main system or to other RTAs. However, interrupts from outside the RTA subsystem propagate into the RTAs and are latched by the interrupt controllers. For example, the main GPTIMER interrupts are latched into the RTA interrupt controllers.

Interrupt lines can be masked by the local interrupt controller. A general strategy is to unmask as few interrupt sources as possible and, at the system level, ensure that no other sources of these interrupts exist.

52.4 General Purpose Timer Unit (Local to RTA)

Each RTA consist of a local timer General Purpose Timer Unit (GPTIMER). The GPTIMER provides a common prescaler and four decrementing timers. The unit is capable of asserting interrupts on timer underflow. For more configuration and option see chapter 36, the local RTA timer only supports four subtimers but the main timer supports seven subtimers. This local timer do not support a hard watchdog trigger (where an event trigger a system level reset) but supports soft watchdog event (an interrupt is triggered using the local interrupt controller after a watchdog event). The RTA supports soft watchdog events from local subtimer 3 (GPTIMER).

Each RTA has two tick outputs taken from the GPTIMER. RTA tick output 0 is asserted when subtimer 1 underflows, RTA tick output 1 is asserted when subtimer 2 underflows.

52.5 RTA Task Manager (RTM) and interrupt time stamp functionality

A central block for real-time RTA execution is the RTA Task Manager (RTM). It can be viewed as a user-configurable link between hardware real-time events, from peripherals such as analog PWM functions, and start of RTA task routines. Task priority is also handled in the RTM.

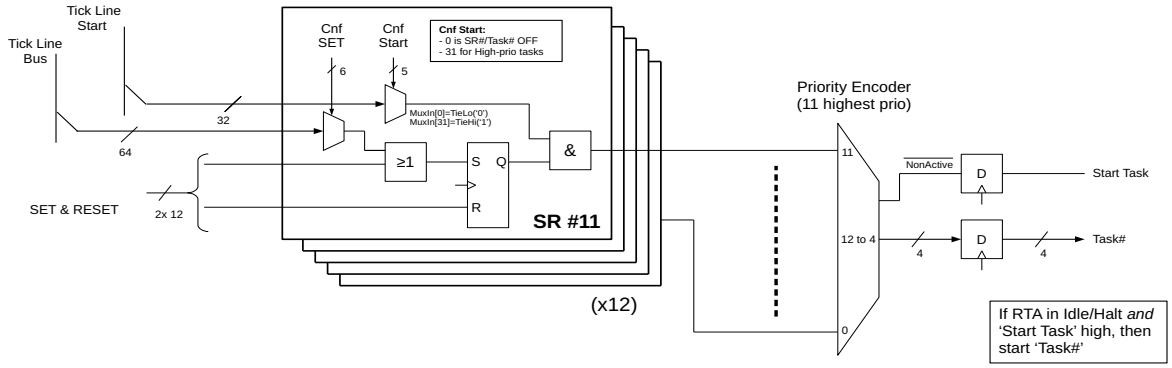


Figure 119. Simplified schematic view of RTA Task Manager (RTM). There are 12 task blocks (SR#0 to SR#11), and three RTM for each RTA.

Interrupt handling in a deeply pipelined architecture, such as LEON3FT, would require unnecessarily long time for storing and re-storing all states in the pipeline structure and other registers to and from the stack. When handling applications like DC/DC controllers, it is critical that the starts and stops of the software execution are very short and efficient. Therefore, the RTA Task Manger (RTM) only executes code in the LEON3FT in 'one execution level', without needing conventional interrupt handling. Before starting the next task, it requires the LEON3FT to be in Idle/Halt state. Then, it starts the next task with highest hardware priority number based on all activated signals into the Priority Encoder in figure 119. Thus, when more than one SR# blocks request task execution into the Priority Encoder, it gives highest priority to the SR#11 block, second highest to SR#10, and so on downto lowest priority to SR#0.

52.5.1 Configuration of RTM

Task priority for a certain task routine is selected by configuring the task to the right SR# block. Each SR# block is directly connected to the same input number of the hardware Priority Encoder, which defines the priority of which RTA routine to start. So the priority configuration is simply done by setting up Cnf SET of the right SR# block to the desired TickLine# that will trig (start) the RTA task routine. This SR# then effectively is the priority number for the task. This number can also serve as being Task# (=SR#, configured to one TickLine#), see to the right in Figure 119.

If a certain RTA routine, for some (unknown) reason, needs to be started from more than one TickLine#, another SR# with yet a Cnf SET configuration can be used. Then, note that more than one Task# is assigned to one RTA routine, which is no problem from RTA execution (availability) point of view, since there is only 'one execution level' in an RTA. But it may be good software-design practice to then keep good track of how all routine variables, other resources, etc, are used/re-used in different real-time execution sequences of that RTA routine.

For the RTA to be able to execute the right routines, the start-address table also needs to be set up for each Task#. Furthermore, Cnf Start can be set to require an RTA start condition, or set to its maximum

value (no start condition), which is further explained below. Finally, when Cnf Start is set to zero, this SR# block is turned off.

52.5.2 Support of Strict Time Schedule still allowing Asynchronous Task Requests

A vital property of the RTM is that it supports start of each task on strict time schedule, based on Tick-Line#, according to the following conditions:

- A 'High-prio task' (Cnf Start=max), with the highest Encoder Priority, starts immediately when: the configured Cnf SET TickLine# is activated and the RTA is in Idle/Halt state. The definition of a High-prio task is that Cnf Start is set to an unconditional '1' into the AND gate in figure 119.
- A 'Low-prio task' does not start until the configured Cnf Start TickLine# activates at the same time as the RTA is in Idle/Halt state. The definition of a Low-prio task is that Cnf Start is set to a Tick-Line# (non-zero, non-max), which effectively can be viewed as a 'time gating' when the Low-prio task is allowed to start.

A general comment, on the (manual) time scheduling work to be done in the building of any real-time system, is that it becomes more straightforward if never more than one High-prio task signal is allowed to be active at a time. Anyway, in whatever way this scheduling work is done, the Priority Encoder will always start the task with the highest SR# (Task#) that has its signal set to high into the Priority Encoder, whenever the RTA returns to Idle/Halt. So whatever real-time system that functions correctly under these rules will work with the RTM on GR716B.

Low-prio tasks with RTA requests (Cnf SET TickLine#) from different kinds of real-time applications such as from Motor Controllers, etc, will come in asynchronously relative to the strict time schedule set up for the High-prio tasks such as DC/DC Controllers. Therefore, all such RTA requests must be 'synchronized' into the High-prio task time schedule. This is done by another tick, Cnf Start *TickLine Start#*, which shall be generated by a Timer inside one of the already existing High-prio applications running on this RTA. Thereby, it will be ensured that all Low-prio tasks will start on the conditions of the (time critical) High-prio tasks.

When a Cnf Start TickLine# enables the start of Low-prio tasks, many or even maybe all Low-prio tasks may be configured to start on that TickLine#. This means that it will be the Priority Encoder that determines the order of execution for all Low-prio tasks that have made an RTA request. Hence, since there is no time schedule or other time-sharing mechanisms controlling priority in-between the Low-prio tasks, the only effective priority that will exist is the Encoder Priority. Therefore, this priority order is vital to carefully work through and assign correctly inter-mutually to all the Low-prio tasks.

This work can actually be trickier than the scheduling of High-prio tasks, because the maximum delays to RTA service for tasks assigned to the lowest priorities need to be calculated, even though there may be uncertainties in how often requests come from Low-prio tasks with higher priority. This kind of scheduling problem is not at all unique for this real-time system in GR716B though, so an experienced system designer will finally find a way to a suitable solution anyhow.

In general, assigning higher priority to High-prio tasks than to Low-prio tasks should be regarded good design practice, even though this actually should not matter in normal execution schedules, since High-prio tasks always should have their own time slots fixed scheduled. Anyhow, this would be to ensure that any High-prio task will always get priority over all Low-prio tasks, e.g. if its start was delayed for any unforeseen single-event reason, such that the start has slided into a time point where a Cnf Start TickLine Start# for Low-prio tasks happens to be activated. In any such case, good design practice should configure the priority to give all High-prio tasks priority before all Low-prio tasks.

52.5.3 User Code Requirements: Return to Idle/Halt and Task-Flag RESET

According to above, Low-prio tasks only will start during the clock cycle when the configured Cnf Start TickLine Start# is active and the RTA is Idle/Halt. This gives the user opportunity to schedule Low-prio asynchronous tasks into time slots where High-prio tasks are not scheduled to run. Thus, it

will prohibit asynchronous tasks from starting at any time point, e.g. just before a High-prio task is scheduled to start, which otherwise would mean that the asynchronous task can keep running far into the High-prio task's scheduled time slot before returning to Idle/Halt. That would most likely be a disaster for the High-prio task application.

Hence, a critical requirement on all Low-prio asynchronous tasks is that their execution time must be shorter than the available time slot in-between the (strictly scheduled) High-prio tasks. Consequently, it is the software designer's responsibility to write each task routine such that it returns to Idle/Halt within a predetermined max execution time (to be defined during system design of the real-time system). This is equally important to fulfill for all task routines – high-prio as well as low-prio routines – without exceptions, for exactly all task routines within the whole RTA execution.

Therefore, long task routines may need to be executed in parts, where each part returns to Idle/Halt within the max predetermined time, and will be re-started one or several times by the Task Manager (RTM), to finish the whole task routine. Hence, the task flag, SR#, should not be RESET until the last part of the whole routine is executed.

Carefully note that it is the task-routine user code that has the responsibility to RESET its task flag, SR#. The reason why that is deliberately left to the user is that it needs to be handled in rather different ways in different real-time applications with different software situations. In a high-priority (short) task routine, typically in DC/DC applications, it will be straightforward to simply RESET the task flag, SR#, at the end of the task routine. On the other hand, e.g. controlling a BLDC motor, it might be longer tasks to be executed, but less time critical. Then, the method of running the whole task routine in shorter parts, returning to Idle/Halt in-between, may need to be used.

Another example is if the Cnf SET TickLine# is re-activated already during the execution of this task’s routine, which might be acceptable in some real-time systems but not in others. Then, in some systems RESET operation may be required before the re-activation to catch the new one, but in others it may be required to have RESET last in the task routine.

Hence, to be able to provide the best user flexibility it seems necessary to leave the RESET operation of the task flag, SR#, to the task-routine user code.

Each RTA subsystem has 3 separate RTM systems named RTM0, RTM1 and RTM2. Each individual RTM system can be configured to use up to 12 tasks. The ticklines are identical to RTA0 and RTA1.

The Tickline bus source for each of the three RTMs are listed in table 700, 701 and 702. Required individual ticklines are selected using the register RTMTC.CNF_SET listed in table 709.

All the RTMs have a common Tickline Start bus which is listed in table 703. Individual ticklines are selected using the register RTMTC. CNF_START 709.

Table 700. Tickline BUS RTM0 (*TickLines0*)

Tickline	Description	Documented in
0 to 3	APWM DAC tick output 0 to 3 (also system interrupt outputs)	Section 54.2.2
4 to 11	FIR 0 to 7 tick output (also system interrupt outputs)	Section 28.1
12 to 31	ACOMP_SYNCH 0 to 19 ##TBD coming in asynchronously	Section 16
32 to 38	Timer27_1 tick output 0 to 6	Section 53.10
39 to 45	Timer27_0 tick output 0 to 6	
46 to 53	Timer32_A tick output 0 to 7	Section 53.9
54 to 61	Timer32_B tick output 0 to 7	
62 to 63	These two tick lines have the constant value 0. (unused)	This table.

Table 701. Tickline BUS RTM1 (*TickLines1*)

Tickline	Description	Documented in
0 to 6	APWM_CF2 timer tick output 0 to 7	Section 53.4
7 to 13	APWM_CF1 timer tick output 0 to 7	
14 to 20	APWM_CF0 timer tick output 0 to 7	
21 to 27	APWM_AB3 timer tick output 0 to 7	Section 53.2
28 to 34	APWM_AB2 timer tick output 0 to 7	
35 to 41	APWM_AB1 timer tick output 0 to 7	
42 to 48	APWM_AB0 timer tick output 0 to 7	
49 to 55	APWM_A timer tick output 0 to 7	Section 53.3
56 to 63	These eight tick lines have the constant value 0. (unused)	This table.

Table 702. Tickline BUS RTM2 (*TickLines2*)

Tickline	Description	Documented in
0 to 3	DAC0-3 synchronization output	Section 15.5.4
4 to 7	ADC 0 sampling buffer 0-3 interrupt output	Section 12.3.9
8 to 11	ADC 1 sampling buffer 0-3 interrupt output	
12 to 15	ADC 2 sampling buffer 0-3 interrupt output	
16 to 19	unused	This table.
20	Timer unit 0 counter 1 underflow (Main GPTIMER0)	Section 35
21	Timer unit 0 counter 2 underflow (Main GPTIMER0)	
22	Timer unit 0 counter 3 underflow (Main GPTIMER0)	
23	Timer unit 0 counter 4 underflow (Main GPTIMER0)	
24	Timer unit 1counter 1 underflow (Main GPTIMER1)	Section 36
25	Timer unit 1counter 2 underflow (Main GPTIMER1)	
26	Timer unit 1counter 3 underflow (Main GPTIMER1)	
27	Timer unit 1counter 4 underflow (Main GPTIMER1)	
28	RTA0 Timer unit counter 1 underflow (RTA0 GPTIMER)	Section 52.4
29	RTA0 Timer unit counter 2 underflow (RTA0 GPTIMER)	
30	RTA1 Timer unit counter 1 underflow (RTA1 GPTIMER)	
31	RTA1 Timer unit counter 2 underflow (RTA1 GPTIMER)	
32 to 47	TimeStamp(s) interrupts (0 to 15)	Section 53.12
48 to 63	GPIO0-15 via Schmitt trigger. Note that GPIO Schmitt trigger input paths are always active regardless of GPIO MUX configuration in SYS.CFG registers. For example, even a GPIO configured as output will cause the Schmitt trigger input signals to toggle.	This table.

Table 703. Tickline Start BUS RTM (*TickLineStart*, common to all RTMs)

Tickline	Description	Documented in
0	Disabled (Input is tied low, when selected the corresponding task block is disabled.)	This table.
1 to 4	Timer32_A tick output 7 to 4	Section 53.9
5 to 8	Timer32_B tick output 7 to 4	
9 to 15	Timer27_0 tick output 0 to 6	Section 53.10
16 to 22	Timer27_1 tick output 0 to 6	
23 to 24	Disabled (Input is tied low, when selected the corresponding task block is disabled)	This table.
25	RTA0 Timer unit counter 1 underflow (RTA0 GPTIMER)	Section 52.4
26	RTA0 Timer unit counter 2 underflow (RTA0 GPTIMER)	
27	RTA1 Timer unit counter 1 underflow (RTA1 GPTIMER)	
28	RTA1 Timer unit counter 2 underflow (RTA1 GPTIMER)	
29	Timer unit 1 counter 1 underflow (Main GPTIMER1)	Section 35
30	Timer unit 1 counter 2 underflow (Main GPTIMER1)	Section 36
31	High priority (Input is tied high)	This table.

Each individual RTM system can be configured to use up to 12 tasks.

Table 704. RTM0, RTM1 and RTM2 status and configuration register (TBD)

AMBA address	Register
0x62060000	RTM Interrupt register
0x62060004	RTM Interrupt Status register
0x62060008	RTM Interrupt Mask register
0x6206000C	RTM Interrupt Edge register
0x62060010	RTM Task 0 Configuration
0x62060014	RTM Task 1 Configuration
0x62060018	RTM Task 2 Configuration
0x6206001C	RTM Task 3 Configuration
0x62060020	RTM Task 4 Configuration
0x62060024	RTM Task 5 Configuration
0x62060028	RTM Task 6 Configuration
0x6206002C	RTM Task 7 Configuration
0x62060030	RTM Task 8 Configuration
0x62060034	RTM Task 9 Configuration
0x62060038	RTM Task 10 Configuration
0x6206003C	RTM Task 11 Configuration
0x62060040	RTM Task Set Configuration
0x62060044	RTM Task Clear Configuration
0x62060080	RTM Status 0
0x62060084	RTM Status 1
0x62060088	RTM Status 2
0x620600E0	Capability register

Note 1: All the RTM systems has the same registers with different address offset

RTA0:

RTM0 AMBA address starts at 0x62060000

RTM1 AMBA address starts at 0x62070000

RTM2 AMBA address starts at 0x62080000

RTA1:

RTM0 AMBA address starts at 0x72060000

RTM1 AMBA address starts at 0x72070000

RTM2 AMBA address starts at 0x72080000

Table 705. 0x00 - RTMINT - RTM Interrupt register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED						P	TS	TQ 11	TQ 10	TQ 9	TQ 8	TQ 7	TQ 6	TQ 5	TQ 4	TQ 3	TQ 2	TQ 1	TQ 0	TN 11	TN 10	TN 9	TN 8	TN 7	TN 6	TN 5	TN 4	TN 3	TN 2	TN 1	TN 0
0x0						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r						rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*

31: 26 RESERVED

25 Configuration parity error status (P). This parity check covers the 14 registers RTMTC, TASKSET and TASKCLR. Becomes 1 when a configuration parity error is detected.

24 TS: TaskStart

23:12 TQ#: TickoQ. Trigger without TaskStart (TBC)

Table 705. 0x00 - RTMINT - RTM Interrupt register

11:0 TN#: Interrupt line trigger corresponding to Task# (as per the SR#11 to SR#0 task block trigger) and TaskStart

For test-purposes, bits 25:0 in this register are writable. The written data overrides the normal data source for exactly one system clock cycle. This provides a means for triggering the associated interrupt from software.

NOTE: The fields of this register are updated at every clock cycle, depending on the corresponding signals. These values are not latched and polling routines could easily miss signal transitions. To monitor the status of a signal it is preferable to use the Interrupt Status register.

Table 706. 0x04 - RTMISTAT - RTM Interrupt Status register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED						P	TS	TQ 11	TQ 10	TQ 9	TQ 8	TQ 7	TQ 6	TQ 5	TQ 4	TQ 3	TQ 2	TQ 1	TQ 0	TN 11	TN 10	TN 9	TN 8	TN 7	TN 6	TN 5	TN 4	TN 3	TN 2	TN 1	TN 0
0x0						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r						wc	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc

31: 26 RESERVED

25: 0 Bit *x* of this register is set to 1 whenever an interrupt has been generated due to bit *x* of RTMINT. Each bit in this stays set to 1 until software clears it by writing 1.

Write 1 to clear.

Table 707. 0x08 - RTMMASK - RTM Interrupt Mask register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED						P	TS	TQ 11	TQ 10	TQ 9	TQ 8	TQ 7	TQ 6	TQ 5	TQ 4	TQ 3	TQ 2	TQ 1	TQ 0	TN 11	TN 10	TN 9	TN 8	TN 7	TN 6	TN 5	TN 4	TN 3	TN 2	TN 1	TN 0
0x0						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

31: 26 RESERVED

25: 0 Bit *x* of this register controls whether the event corresponding to bit *x* of RTMINT can generate an interrupt or not. When bit *x* is 0, that event will not generate interrupts.

Table 708. 0xC - RTMEDGE - RTM Interrupt Edge register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED						P	TS	TQ 11	TQ 10	TQ 9	TQ 8	TQ 7	TQ 6	TQ 5	TQ 4	TQ 3	TQ 2	TQ 1	TQ 0	TN 11	TN 10	TN 9	TN 8	TN 7	TN 6	TN 5	TN 4	TN 3	TN 2	TN 1	TN 0
0x0						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

31: 26 Reserved

25: 0 Bit *x* of this registers controls the polarity of interrupt generation from bit *x* of the RTMINT register.
0 => falling edge: interrupt generated when bit *x* changes from 1 to 0
1 => rising edge: interrupt generated when bit *x* changes from 0 to 1

Table 709. 0x10-3C, RTMTC - RTM Taskx Configuration

31	11	10	6	5	0
RESERVED		CNF_START		CNF_SET	
0x0		0x0		0x0	
r		rw		rw	

31: 11 RESERVED

10:6 CNF_START: Select one tickline start from 32 ticklines. TBD Refer table.

- 0 is SR#/Task# OFF

- 31 for High-priority tasks

5:0 CNF_SET: Select one tickline from 64 tickline bus. TBD Refer table.

Note: Register description applies for RTM Task 0-11 Configuration registers.

Table 710. 0x40 - TASKSET - RTM Task Set Configuration

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																				TS 11	TS 10	TS 9	TS 8	TS 7	TS 6	TS 5	TS 4	TS 3	TS 2	TS 1	TS 0
0x0																				0	0	0	0	0	0	0	0	0	0	0	0
r																				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

31:12 RESERVED

11:0 TS#: Task set corresponding to SR#11 to SR#0 task blocks.

Table 711. 0x44 - TASKCLR - RTM Task Clear Configuration

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																				TC 11	TC 10	TC 9	TC 8	TC 7	TC 6	TC 5	TC 4	TC 3	TC 2	TC 1	TC 0
0x0																				0	0	0	0	0	0	0	0	0	0	0	0
r																				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

31:12 RESERVED

11:0 TC#: Task clear corresponding to SR#11 to SR#0 task blocks.

Table 712. 0x80 - RTMSTAT0 - RTM Status0 register

31	5	4	0
RESERVED		TNH	
0x0		0x0	
r		r	

31: 5 RESERVED

4:0 TNH: Task Number Hold (Current Active Task Number Selected)

Table 713. 0x84 - RTMSTAT1 - RTM Status1 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																				TQ 11	TQ 10	TQ 9	TQ 8	TQ 7	TQ 6	TQ 5	TQ 4	TQ 3	TQ 2	TQ 1	TQ 0
0x0																				0	0	0	0	0	0	0	0	0	0	0	0
r																				r	r	r	r	r	r	r	r	r	r	r	r

31: 12 RESERVED

11:0 TQ#: TickoQ. Trigger without TaskStart (TBC) Next Waiting Task Number

Table 714. 0x88 - RTMSTAT2 - RTM Status2 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																				TH 11	TH 10	TH 9	TH 8	TH 7	TH 6	TH 5	TH 4	TH 3	TH 2	TH 1	TH 0
0x0																				0	0	0	0	0	0	0	0	0	0	0	0
r																				r	r	r	r	r	r	r	r	r	r	r	r

31: 12 RESERVED

11:0 TH: Ticko High Priority Hold (TBC) Next Waiting Task Number

52.5.4 RTM Local register interface information

Table 715. 0xE0 - INFO - RTM Local register interface information

31 24	23 16	15 8	7 0
IRQ	CFG	STAT	IRQ
0x1A	0x0E	0x03	0x1A
r	r	r	r

31: 24 IRQ: Number of Interrupts (same as 7:0)

23: 16 CFG: Number of Configuration registers

15:8 STAT: Number of Status registers

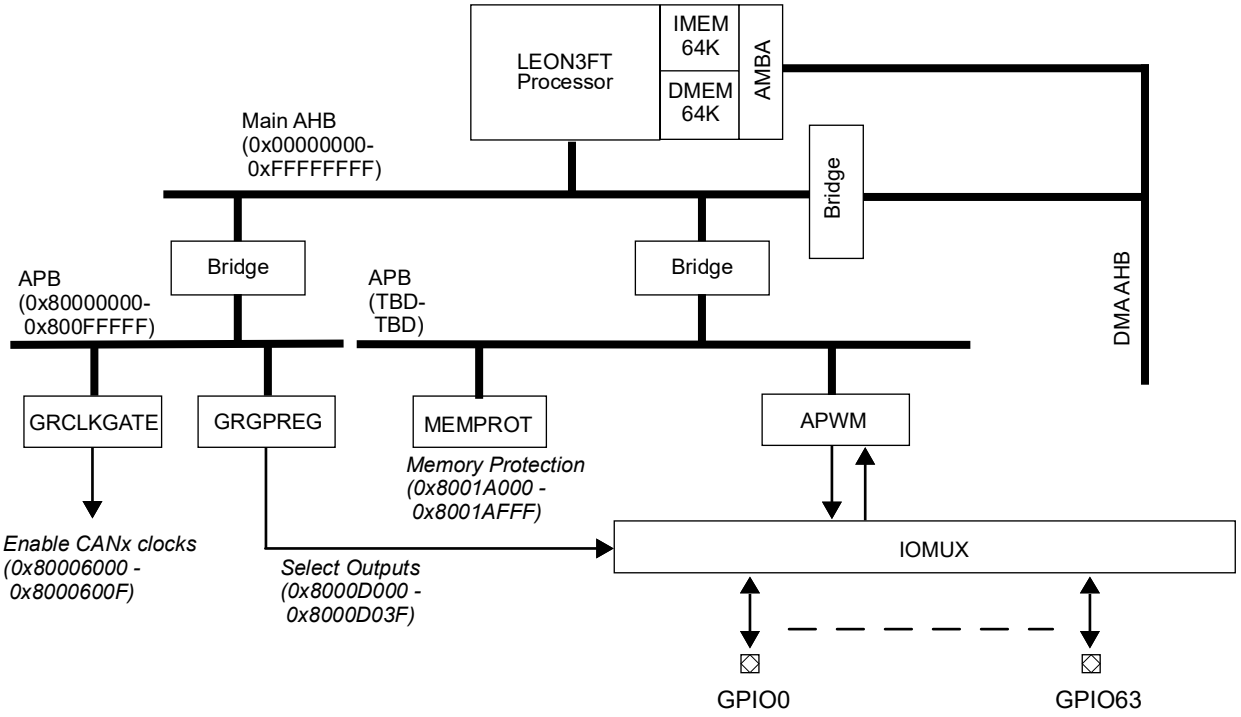
7:0 IRQ: Number of Interrupts

53 Analog Applications Pulse Width Modulation (APWM)

The GR716B microcontroller contains an Analog applications PWM (APWM) unit, which contains several different types of PWM functions. The main purpose of the APWM unit is to support DC/DC and motor control applications and it can be used in many other application areas.

The APWM controller unit controls its own external pins and has a unique AMBA address described in section 2.10. The APWM unit is located on APB bus in the address range from **0x81000000 to 0x8150f0ff**. See APWM unit connections in figure 120. It shows memory locations and functions used for APWM configuration and control.

Figure 120. GR716B APWM bus and pin connection



The secondary clock gating unit **GRCLKGATE** described in chapter 28 is used to enable/disable the APWM units. The unit **GRCLKGATE** can also be used to perform reset of the APWM unit. Software must enable clock and release reset described in chapter 28 before APWM configuration and transmission can start.

External IO selection per APWM unit is made in the system IO configuration register (**GRGPREG**) in the address range from 0x8000D000 to 0x8000D03F. See chapter 28 for further information.

The **APWM** unit control its own external pins and has a unique AMBA address described in section 2.10. The APWM configuration and status registers are described in section 53.8

System can be configured to protect and restrict access to individual APWM units in the **MEMPROT** unit. See chapter 47 for more information.

53.1 Overview of APWM unit

To support the area of switching power supply (DC/DC) and motor control applications, a set of pulse width modulator (PWM) hardware blocks have been implemented in GR716B, see figure 121 and sections 2.2.12 and 2.2.13.

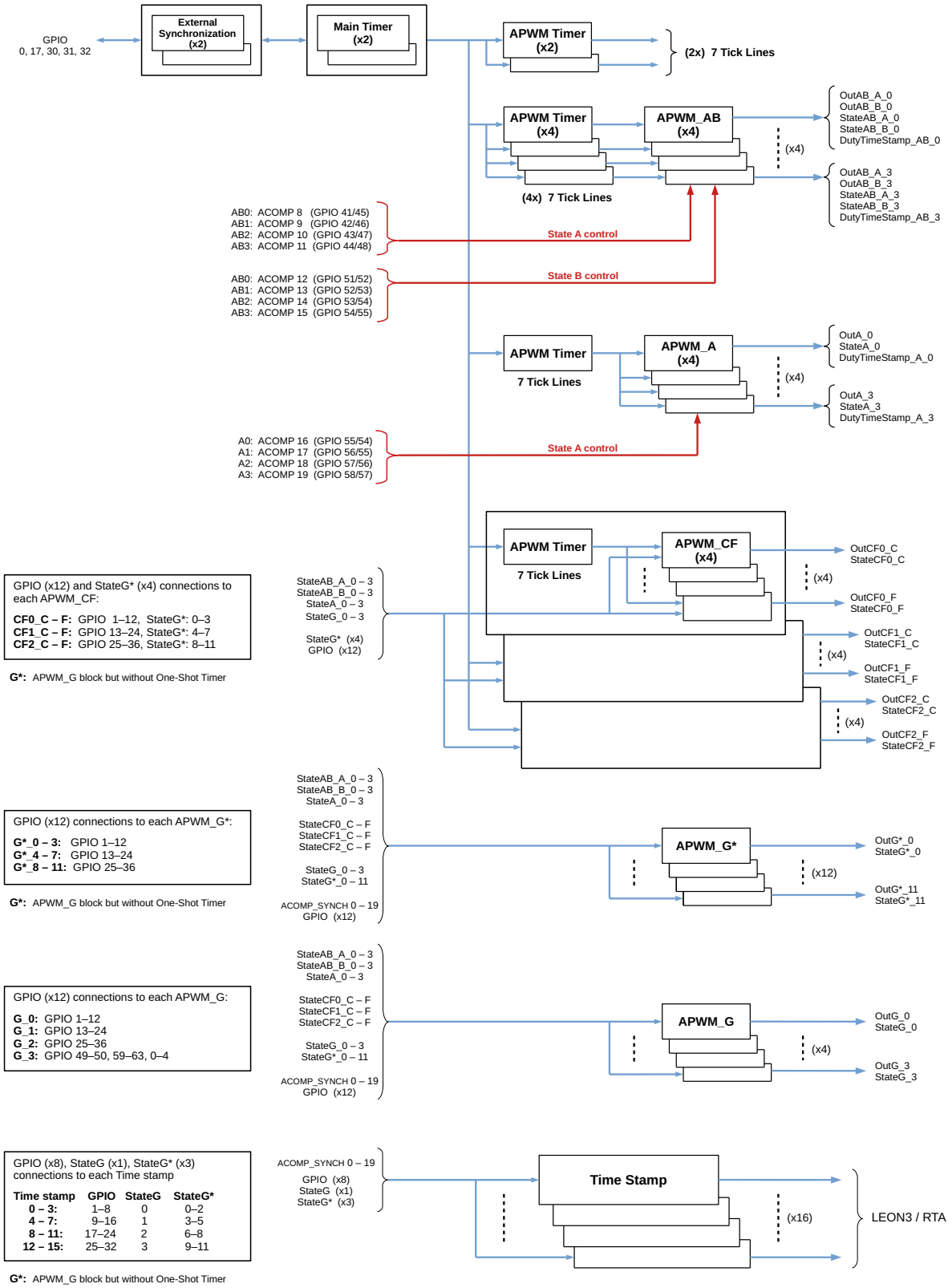


Figure 121. Overview of APWM hardware functions. All APWM output blocks to the right can be real-time synchronized to each other, and the Main Timers at the top can be externally synchronized to or from external circuitry, e.g., another GR716B device.

For DC/DC control applications, the APWM hardware blocks provide support for:

- 4 independent DC/DC controllers in peak-current control mode (APWM_AB), and more controllers in voltage or frequency control mode (APWM_A/AB/CF).
- Overcurrent detection and protection shutdown (ACOMP and Alarm Matrix).
- Max duty-cycle limiting.
- Power switch turn-on delay, configurable cycle-by-cycle from RTA.
- Leading edge blanking (LEB), configurable cycle-by-cycle from RTA.
- Extended PWM outputs (APWM_CF and APWM_G), supporting various full-bridge DC/DC converter topologies.

For motor control applications, the APWM hardware blocks provide support for:

- 4 BLDC or PMSM motors or 6 micro stepper motors, in PWM control mode (APWM_CF), and more motors in immediate GPIO control mode.
- 4 analog channels with simultaneous ADC sampling to facilitate motor control.

The number of supported controllers can be combined as different number of DC/DC controllers and motor controllers. And in the case where only APWM_AB blocks (x4) are used for 4 basic DC/DC converters (e.g., buck, boost, flyback), the APWM_CF blocks (x12) can be used to support 4 motors.

The APWM_AB block is designed to generate two gate-driver signals suitable to control the two power switches in a half-bridge switch constellation. The APWM_A block is a simplified version of APWM_AB. It has only one output and can be used in application areas such as low-power supply converters, and current or voltage signal sources. When motor PWM control is to be implemented, the APWM_CF blocks can be used to control half-bridges for standard BLDC and PMSM motors. Each APWM_CF is a general PWM block configurable to work stand alone or as an extension of APWM_A /AB. The APWM_G block, working as extension on the other APWM blocks, supports control of various complex DC/DC converter topologies, including full-bridge converters with synchronous rectification such as phase-shifted full bridge.

Each APWM Timer (24bit) in figure 121 can be configured to its own period cycle, and to its own start time point in a Main Timer cycle (32bit). When running more than one DC/DC converter, it can be of interest to run all power converters on integer multiples of each others' switching frequencies. Otherwise, there may be a significant risk of EMC problems from undesired intermediate frequencies, i.e., the difference(s) between switching frequencies appearing as low-frequency disturbance(s), which can be tricky to find and solve in switching power systems in general. The configurability of these timers allows for programming of any desired integer-multiple of frequencies and relative time positions of power-cycle starts for each DC/DC controller individually.

Each timer can generate real-time tick/trigger signals, which can be configured to start RTA task routines, ADC conversions, DAC writings, etc. A real-time tick/trig can be set to any time position in the cycle of the timer, and it will be cycle-repeatable and deterministic on system clock cycle-by-cycle level.

53.2 APWM_AB description

The APWM_AB hardware block is designed to generate two gate-control signals, Out A and Out B in figure 122, suitable to control the two power switches in a half-bridge switch constellation. The duty cycle can be controlled from peak-current analog comparator, ACOMP State A, or from RTA. Also the frequency from APWM Timer can be reprogrammed PWM cycle-by-cycle from RTA. Hence, all DC/DC topologies, which run in current or voltage control mode with variable duty cycle or frequency, are supported.

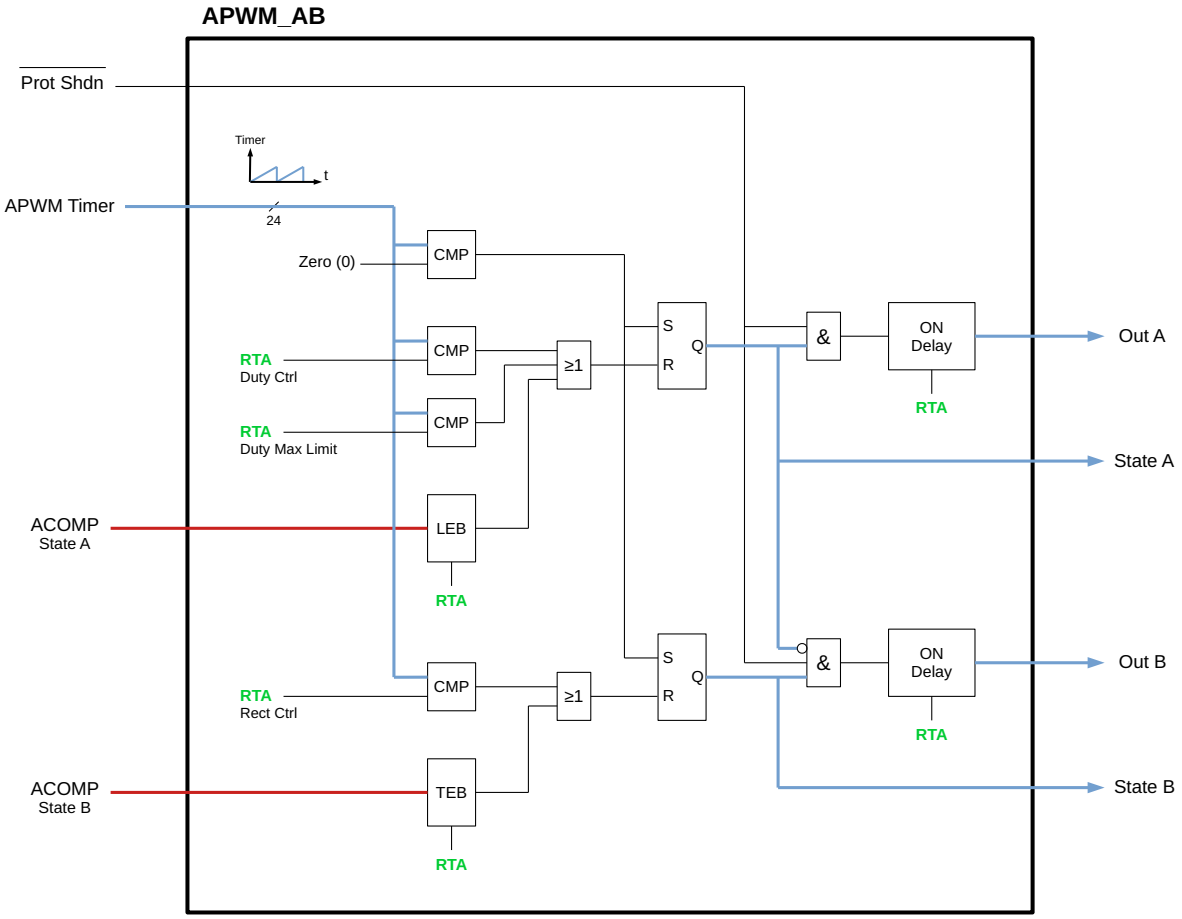


Figure 122. Functional diagram of APWM_AB

53.2.1 Operation

A half-bridge switch constellation is used in many switching power-converter topologies such as in step-up (boost) converters and step-down (buck) converters. The latter will be used as application example in the detailed presentation of the APWM_AB operation here, see figure 123. More examples of APWM_AB application connection diagrams for commonly used DC/DC topologies are presented in section 53.13.1.

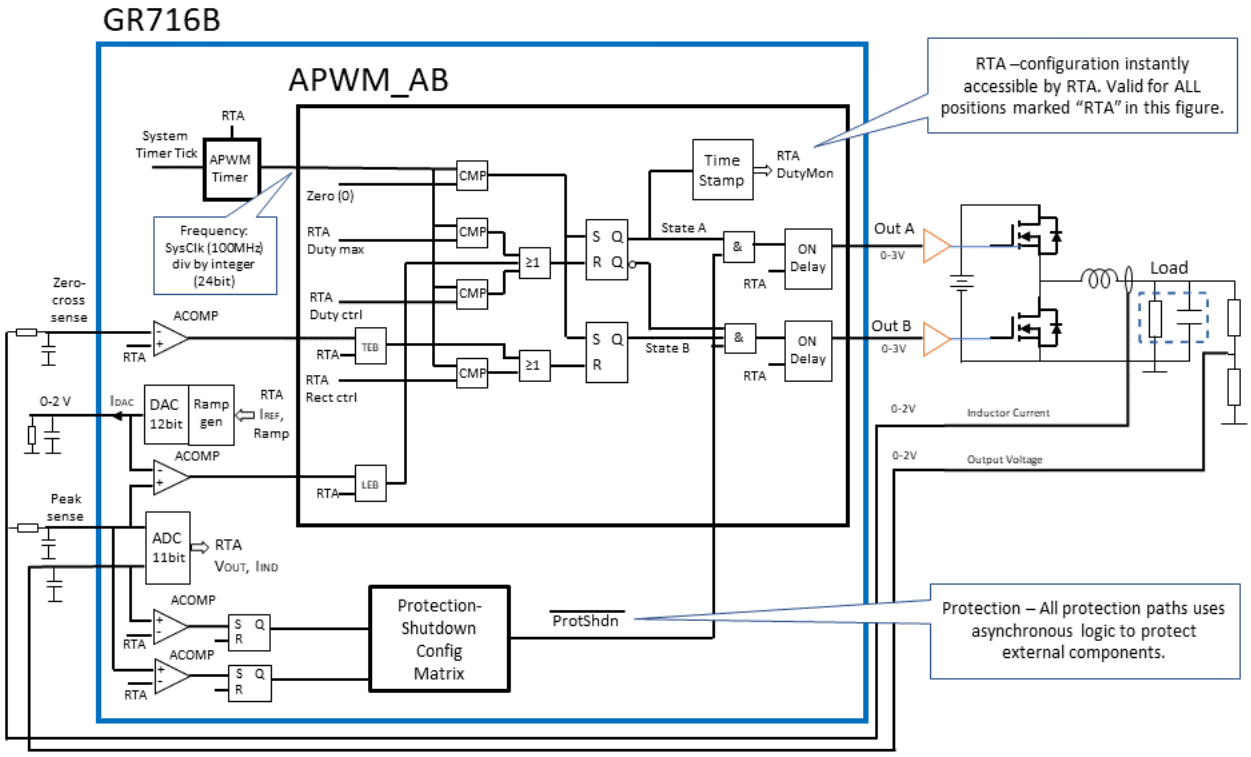


Figure 123. Simplified application schematic of a step-down (buck) converter controlled by APWM_AB.

The Out A signal is connected to the active ON duty-cycle switch, which connects the inductor to the input power bus, and thereby increases the inductor current, i.e., increases its stored energy. The second switch is used for synchronous rectification during the switch-cycle OFF time, when the stored inductor energy is discharging into the large output capacitor. In cases where the second switch is implemented, it is connected to the Out B signal. This switch can be replaced by a diode when power efficiency is of less importance. The main benefit of using the second switch is that it provides lower power dissipation compared to a diode, thereby improving the power efficiency of the converter. For applications in the order of 10 A and higher output current, this switch is normally beneficial to implement, but for the order of 1 A and lower it is less space and cost efficient compared to the benefits.

The ON Delay block provides programmable dead-time between switching of the two power switches, which is reprogrammable PWM cycle-by-cycle from RTA. The main use case is usually to ensure enough time margin for skew between the two PCB gate drivers for Out A and B, where a dead-time in OFF state must be guaranteed at the two power MOSFET gates. Another use case can be to continuously maintain maximum power efficiency under all operating conditions, e.g., change of switch timing characteristics over temperature, load current, etc. Also various switching schemes based on optimum zero-voltage switching are supported by continuous control of the ON Delay block.

The duty-cycle regulation is based on peak-current control. The current-sense signal goes into the on-chip analog comparator, ACOMP for State A, which has the reference level set by an RTA-programmable DAC output. The DAC provides a sourcing-current output, where the user should put a resistor to ground on PCB to give the resulting DAC-output voltage. The DAC also has built-in RTA-programmable digital ramp generation.

The current-sense signal goes into an ADC input, in parallel with the ACOMP input, so it can be measured for monitoring purposes and made available to the RTA for control-law calculations, etc. When ADC measurement is to be done, the ADC track turn-on event can disturb the GPIO pin, see 12.3.15

to 12.3.16. Therefore, in the case current measurement should be done as early as possible in the duty-cycle period, the track turn-on time point is preferably configured at the power-switch turn-on point.

The following leading-edge blanking (LEB) time should be configured long enough to cover the ADC tack transient and the switching of the power switches, which may be in the range 0.1-0.5 us. The track time should be long enough to ensure accurate settling of the ADC S/H capacitor, which is 0.3-0.7 us after LEB ends, depending on settling during LEB time and required accuracy. In the case current measurement should be done in the middle of the duty-switch on time, where the inductor current is approximately the average of the on-time current, the Tick time point for ADC start needs to be continuously updated with the right track start point. Moreover, ADC S/H capacitor precharge to ground should be considered, see 12.3.15.

The regulation of the converter output voltage is typically performed by first taking an ADC sample of the output voltage. The RTA executes the control-law calculations based on this output-voltage sample and any earlier samples. This results in a new current-reference level written to the DAC before the analog comparator will reach its peak-current trig level in the upcoming power switching cycle. Note that this trig level will have no effect until the leading edge blanking (LEB) time has past, so it is essential to configure LEB correctly. This whole ADC-RTA-DAC sequence is continuously repeated once per power-switching cycle, which provides continuous regulation of the output voltage.

Duty-cycle regulation can, instead of peak-current control, be performed directly from the RTA by using ‘Duty Ctrl’ in figure 123. This is the case when voltage-mode control is to be implemented instead of peak-current control. Moreover, in special transient situations, running in peak-current mode, it may be handy to be able to control (limit) the duty cycle temporarily, to handle transient peaks by non-linear regulation possible to implement only digitally in RTA software.

In addition to the above duty-cycle control methods, there is also a maximum duty-cycle limit that can be configured, see ‘Duty Max Limit’. It can be useful in converter implementations that require a guaranteed minimum OFF time per switch cycle, e.g., to re-charge a bootstrapped highside gate-driver supply. Moreover, some converter topologies, e.g., step-up (boost) converters, must be prohibited from running all the way up to 100% duty cycle, to avoid power-switch stress or damage from regulation-transient overcurrents.

When synchronous rectification is used, it can be configured to run continuously (Out B inverse of Out A); or, diode emulation can be used when back-feeding of power onto the input power bus should be avoided. In diode-emulation mode, either ‘Rect Ctrl’ by RTA or Zero-crossing sense by ACOMP can be used to turn off the rectification switch (Out B), which should happen at the current zero-crossing point. When ‘Rect Ctrl’ is used, a new value can be written PWM cycle-by-cycle from RTA. When zero-crossing sense by ACOMP is used, it is essential to handle current-sense parasitics on PCB carefully and configure trailing edge blanking (TEB) time correctly.

Protection functions in hardware are provided in the form of overvoltage and overcurrent analog comparators, see the two ACOMPs connected to SR-latches in figure 123. They are connected to the central Alarm and Shutdown Matrix in GR716B. Here, any protection alarm signals from DC/DC converters, or other critical related functions on the chip or PCB design, can be configured to shut down one or more DC/DC converters and other related functions, to ensure safe and simultaneous shut down of each whole application. In this ACOMP use case, the disturbance tolerant (SET hard) configuration mode can be recommended, to avoid unnecessary faulty shutdowns of applications.

Note that these protection ACOMPs re-use the same sense package pins as the regulation sense pins, so they cannot protect against hardware failure inside the power converter itself, e.g., pin failure on GR716B. Hence, these ACOMPs can typically be regarded as protection against excessive load current caused by temporary load failures, or against abnormal input voltage, e.g., caused by fuse blows on power system level. They can also protect against faulty or unexpected behavior of the control-law software in RTA. In cases where additional protection would be required, e.g., protection against hardware failures inside the power converter itself, independent hardware will have to monitor the converter, and safely shut it down with an independent set of power switches other than the two power

switches discussed in this application example. This independent hardware, including the independent set of power switches, need to be supplied by independent auxiliary supply on power system level.

53.2.2 Architecture

The current value of the APWM_Timer input is readable through the APWM_AB Timer Status register. There is an APWM_TIMER block for each APWM_AB block. The timer is configurable and its operation is described in section 53.10.

The description below refers to figure 122.

The Out A and Out B are never asserted at the same time.

Out A

When the 24 least significant bits of APWM_Timer match the field SET of the APWM_AB Set Timer Value Configuration register, the S input of the SR flip-flop labeled State A is asserted. This in turn asserts the State A signal at the next clock cycle, which remains at that value until a reset condition occurs. There are two types of reset conditions:

- Duty Cycle Reset: Triggered when the 24 least significant bits of APWM_TIMER (APWM_AB) match the RESET field of the APWM_AB Duty Max Configuration register or the RESET field of the APWM_AB Duty Ctrl Configuration register.
- Asynchronous Reset: Triggered by one of the two ACOMP State A inputs. The selection of the input and its corresponding value are configured in the APWM_AB State A Asynchronous Clear Configuration register. The Leading Edge Blanking (LEB) block allows for these resets to be ignored until the LEB block completes its countdown.

When the S input is asserted, a 12-bit down-counter in the LEB block starts counting down from the PRESET value in the APWM_AB Leading Edge Blanking Configuration register. It decrements by one every clock cycle and all asynchronous reset conditions related to State A are ignored until it reaches zero.

The LEB block also outputs a signal, LEB_to_ACOMP, which behaves according to the settings in the SEL field of the APWM_AB Leading Edge Blanking Configuration register.

When State A is asserted, its value is transferred to Out A with a certain delay through the ON Delay block. This delay is configurable through the field DELAY of the APWM_AB ON Delay Out A Configuration register. When State A is deasserted, Out A is also immediately deasserted.

The APWM_AB Time Stamp Status register stores the value of the APWM_Timer input whenever the State A signal has a transition high-to-low.

Important: The output signal Out A, is immediately set to zero whenever the input signal $\overline{\text{Prot Shdn}}$ is asserted (Prot Shdn = 0), overriding other conditions.

Out B

As per State A, when the 24 least significant bits of APWM_Timer match the field SET of the APWM_AB Set Timer Value Configuration register, the S input of the SR flip-flop labeled State B is asserted. This in turn asserts the State B signal at the next clock cycle, which remains at that value until a reset condition occurs. There are two types of reset conditions:

- Duty Cycle Reset: Triggered when the 24 least significant bits of the APWM_TIMER input match the RESET field of the APWM_AB Rect Ctrl Configuration register.
- Asynchronous Reset: Triggered by one of the two ACOMP State B inputs. The selection of the input and its corresponding value are configured in the APWM_AB State B Asynchronous Clear Configuration register. The Trailing Edge Blanking (TEB) block allows for these resets to be ignored until the TEB block completes its countdown.

When the S input is asserted, a 12-bit down-counter in the TEB block starts counting down from the value set in the APWM_AB Trailing Edge Blanking Configuration register. It decrements by one per clock cycle and all asynchronous reset conditions related to State B are ignored until it reaches zero.

The TEB block also outputs a signal, LEB_to_ACOMP, which behaves according to the settings in the SEL field of the APWM_AB Trailing Edge Blanking Configuration register.

When State A is not asserted and State B becomes asserted, Out B is asserted after a delay controlled by the ON Delay block. This delay can be adjusted using the DELAY field in the APWM_AB ON Delay Out B Configuration register. When State B is deasserted, Out B is immediately deasserted as well.

Important: The output signal Out B, is immediately set to zero whenever the input signal $\overline{\text{Prot Shdn}}$ is asserted (Prot Shdn = 0), overriding other conditions.

53.2.3 Registers

The core is programmed through registers mapped into APB address space.

*Table 716.*APWM_AB registers

APB address offset	Register
0x81200000	APWM_TIMER (APWM_AB0) Start [1)]
0x81201000	APWM_AB0 Status register
0x81201004	APWM_AB0 Interrupt Status register
0x81201008	APWM_AB0 Interrupt Mask register
0x8120100C	APWM_AB0 Interrupt Edge register
0x81201010	APWM_AB0 Duty Ctrl Configuration register
0x81201014	APWM_AB0 State A Asynchronous Clear Configuration register
0x81201018	APWM_AB0 Rect Ctrl Configuration register
0x8120101C	APWM_AB0 State B Asynchronous Clear Configuration register
0x81201020	APWM_AB0 Leading Edge Blanking Configuration register
0x81201024	APWM_AB0 Trailing Edge Blanking Configuration register
0x81201028	APWM_AB0 ON Delay Out A Configuration register
0x8120102C	APWM_AB0 ON Delay Out B Configuration register
0x81201030	APWM_AB0 Duty Max Configuration register
0x81201034	APWM_AB0 Set Timer Value Configuration register
0x81201080	APWM_AB0 Time Stamp Status register
0x81201084	APWM_AB0 Timer Status
0x81202000	APWM_TIMER (APWM_AB1) Start [1)]
0x81203000	APWM_AB1 Start [2)]
0x81204000	APWM_TIMER (APWM_AB2) Start [1)]
0x81205000	APWM_AB2 Start [2)]
0x81206000	APWM_TIMER (APWM_AB3) Start [1)]
0x81207000	APWM_AB3 Start [2)]

Note 1: Refer section

Note 2: APWM AB 1, 2 and 3 has the same registers as APWM AB 0 with a different offset

53.2.4 APWM_AB Status Register

Table 717. 0x00 - APWM_AB Status register

31	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED	E	R	R	R	R	R	R	OB	OA	TEBAC	GPFA	GPSH	DACF	DACSH	RECT	R	S	LEBAC	LEBSR	PS	SB	SA
NR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
r	rw*	r	r	r	r	r	r	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*

NOTE: The rw fields of this register can be written in order to trigger an interrupt, as per the conditions described in the APWM_AB Interrupt Status register.

NOTE: The fields of this register are updated at every clock cycle, depending on the corresponding signals. These values are not latched and polling routines could easily miss signal transitions. To monitor the status of a signal it is preferable to use the Interrupt Status register.

21 Error (E) - There has been an uncorrectable error in one of the configurations registers. For the details of the protection scheme, see APWM_AB Check Bits registers.

20 RESERVED

19 RESERVED

18 RESERVED

17 RESERVED

16 RESERVED

15 RESERVED

14 Out B(OB) - Value of the output Out B

13 Out A (OA) - Value of the output Out A

12 TEB to ACOMP (TEBAC) - value of the TEB to ACOMP signal output of the TEB block in figure 122

11 ACOMP GPIO Fast (GPFA) - value of the input ACOMP GPIO Fast in figure 122

10 ACOMP GPIO Set Hard (GPSH) - value of the input ACOMP GPIO Set Hard in figure 122

9 ACOMP DAC Fast (DACFA) - value of the input ACOMP DAC Fast in figure 122

8 ACOMP DAC Set Hard (DACSH) - value of the input ACOMP DAC Set Hard in figure 122

7 RECT - Triggered when the 24 least significant bits of the APWM_TIMER input match the RESET field of the APWM_AB Rect Ctrl Configuration register

6 Reset (R) - value of the signal input R to the State A SR flip flop in figure 122

5 Set (S) - value of the signal input S to the State A SR flip flop in figure 122

4 LEB to ACOMP (LEBAC) - value of the LEB to ACOMP signal output of the LEB block in figure 122

3 LEB to SR (LEBSR) - If set to 1, the LEB block is not preventing asynchronous resets due to the ACOMP inputs

2 Protection Shutdown (PS) - value of signal “Prot Shdn” in figure 122

1 State B (SB) - value of signal State B

0 State A (SA) - value of signal State A

53.2.5 APWM_AB Interrupt Status register

Table 718. 0x04 - APWM_AB Interrupt Status register

31	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED	E	TEBNE	TEBPA	TEBRE	LEBNE	LEBPA	LEBRE	OB	OA	TEBAC	GPFA	GPSH	DACF	DACSH	RECT	R	S	LEBAC	LEBSR	PS	SB	SA
NR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
r	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc

Table 718. 0x04 - APWM_AB Interrupt Status register

A bit in this register (and an interrupt) will only be asserted if the corresponding bit in the APWM_AB Interrupt Mask register is set to 1. When this condition is met, the bit will be asserted whenever the corresponding bit in the APWM_AB Status register changes to match the value of the same bit in the APWM_AB Interrupt Edge register.

Write 1 to clear.

53.2.6 APWM_AB Interrupt Mask register

Table 719. 0x08 - APWM_AB Interrupt Mask register

31	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED	E	TEBNE	TEBPA	TEBRE	LEBNE	LEBPA	LEBRE	OB	OA	TEBAC	GPFA	GPSH	DACF	DACSH	RECT	R	S	LEBAC	LEBSR	PS	SB	SA
NR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

See APWM_AB Interrupt Status register.

53.2.7 APWM_AB Interrupt Edge register

Table 720. 0xC - APWM_AB Interrupt Edge register

31	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED	E	TEBNE	TEBPA	TEBRE	LEBNE	LEBPA	LEBRE	OB	OA	TEBAC	GPFA	GPSH	DACF	DACSH	RECT	R	S	LEBAC	LEBSR	PS	SB	SA
NR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

See APWM_AB Interrupt Status register.

53.2.8 APWM_AB Duty Ctrl Configuration register

Table 721. 0x10 - APWM_AB Duty Ctrl Configuration register

31 27	26 24	23 0
RESERVED		RESET
NR		0
r		r

23:0 RESET - Determine the value of the 24 least significant bits of the APWM_TIMER input at which the R input of the SR flip-flop State A shall be asserted.

53.2.9 APWM_AB State A Asynchronous Clear Configuration register

Table 722. 0x14 - APWM_AB State A Asynchronous Clear Configuration register

31	4	3	2	1	0
RESERVED		RES	EAC	SEL	
NR		0	0	0	
R		rw	rw	rw	

3 RES - Only write 0 to this field.

2 Enable Asynchronous Clear (EAC) - Set to 1 to enable clearing the State A signal using the signals selected through the field SEL of this register

1:0 Select which signal and which value of it can trigger an asynchronous clear of the State A signal

“00” -> ACOMP DAC Set Hard = 1

“01” -> ACOMP DAC Set Hard = 0

“10” -> ACOMP DAC Fast = 0

“11” -> ACOMP DAC Fast = 1

53.2.10 APWM_AB Rect Ctrl Configuration register

Table 723. 0x18 - APWM_AB Rect Ctrl Configuration register

31 27	26 24	23 0
RESERVED		RESET
NR		0
r		r

23:0 RESET - Determine the value of the 24 least significant bits of the APWM_TIMER input at which the R input of the SR flip-flop State B shall be asserted.

53.2.11 APWM_AB State B Asynchronous Clear Configuration register

Table 724. 0x1C - APWM_AB State B Asynchronous Clear Configuration register

31 4	3	2	1 0
RESERVED	RES	EAC	SEL
NR	0	0	0
R	rw	rw	rw

3 RES - Only write 0 to this field.

2 Enable Asynchronous Clear (EAC) - Set to 1 to enable clearing the State B signal using the signals selected through the field SEL of this register

1:0 Select which signal and which value of it can trigger an asynchronous clear of the State B signal

“00” -> ACOMP GPIO Set Hard = 1

“01” -> ACOMP GPIO Set Hard = 0

“10” -> ACOMP GPIOFast = 0

“11” -> ACOMP GPIOFast = 1

53.2.12 APWM_AB Leading Edge Blanking Configuration register

Table 725. 0x20 - APWM_AB Leading Edge Blanking Configuration register

31 24	23 14	13 12	11 0
RESERVED		SEL	PRESET
NR		0	0
r		rw	rw

13:12 SEL - Configured the behavior of the output LEB to ACOMP.

00 - LEB to ACOMP is always set to 1

01 - LEB to ACOMP is set to 0 for the time interval from when the LEB down-counter starts counting until it reaches zero. It is set to 1 otherwise.

10- LEB to ACOMP is set to 0 for the time interval from the negative edge of State B to when the LEB down-counter reaches zero. It is set to 1 otherwise.

11 - LEB to ACOMP is set to 0 for the time interval from when the LEB down-counter starts counting until it reaches zero. It is then set to 1 if the State A signal is set to1 and it is kept to 1 until the State A signal goes to 0.

11:0 PRESET - Leading Edge Blanking Down-counter Preset value

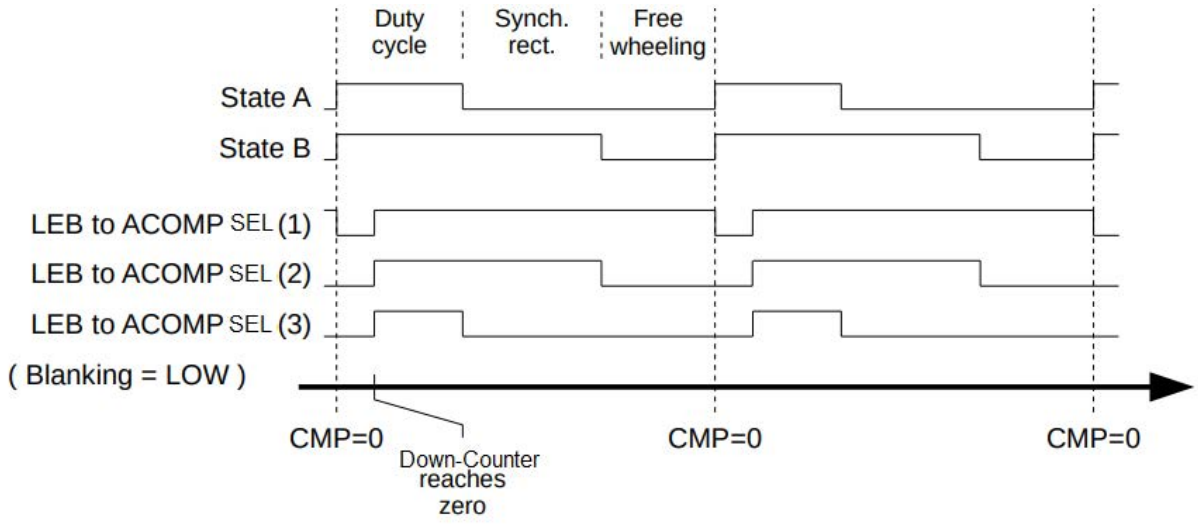


Figure 124. LEB to ACOMP signal wave diagram

53.2.13 APWM_AB Trailing Edge Blanking Configuration register

Table 726. 0x24 - APWM_AB Trailing Edge Blanking Configuration register

31 24	23 14	13 12	11 0
RESERVED		SEL	PRESET
NR		0	0
r		rw	rw

13:12 SEL - Configured the behavior of the output TEB to ACOMP.

00 - TEB to ACOMP is always set to 1

01 - TEB to ACOMP is set to 0 for the time interval from when the TEB down-counter starts counting until it reaches zero. It is set to 1 otherwise.

10- TEB to ACOMP is set to 0 for the time interval from the negative edge of State A to when the TEB down-counter reaches zero. It is set to 1 otherwise.

11 - TEB to ACOMP is set to 0 for the time interval from when the TEB down-counter starts counting until it reaches zero. It is then set to 1 if the State Bsignal is set to1 and it is kept to 1 until the State B signal goes to 0.

11:0 PRESET - Trailing Edge Blanking Down-counter Preset value

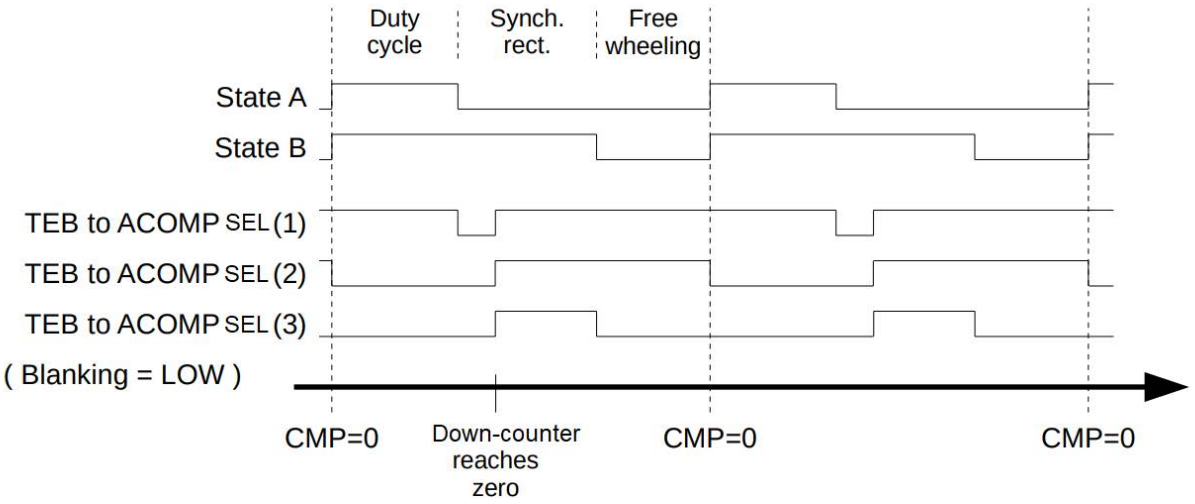


Figure 125. TEB to ACOMP signal wave diagram

53.2.14 APWM_AB ON Delay Out A Configuration register

Table 727. 0x28 - APWM_AB ON Delay Out A Configuration register

31 12	11 0
RESERVED	DELAY
NR	0
r	rw

11:0 DELAY - When State A is asserted, its value is transferred to Out A with after DELAY clock cycles

53.2.15 APWM_AB ON Delay Out B Configuration register

Table 728. 0x2C - APWM_AB ON Delay Out B Configuration register

31 12	11 0
RESERVED	DELAY
NR	0
r	rw

11:0 DELAY - From the moment when State A is not asserted and State B is asserted, Out B is asserted after a delay of DELAY clock cycles.

53.2.16 APWM_AB Duty Max Configuration register

Table 729. 0x30 - APWM_AB Duty Max Configuration register

31 27	26 24	23 0
RESERVED		RESET
NR		0
r		r

23:0 RESET - Determine the value of the 24 least significant bits of the APWM_TIMER input at which the R input of the SR flip-flop State A shall be asserted.

53.2.17 APWM_AB Set Timer Value Configuration register

Table 730. 0x34 - APWM_AB Set Timer Value configuration register

31 27	26 24	23 0
RESERVED		RESET
NR		0
r		r

23:0 SET - Determine the value of the 24 least significant bits of the APWM_TIMER input at which the S input of the SR flip-flop State A shall be asserted.

53.2.18 APWM_AB Time Stamp Status register

Table 731. 0x80 - APWM_AB Time Stamp Status register

31 27	26 24	23 0
RESERVED	VALUE_3	VALUE_24
NR	0	0
r	r	r

Table 731. 0x80 - APWM_AB Time Stamp Status register

27:24	VALUE_3 - Stores the value of the 3 most significant bits of the input APWM_TIMER the last time the State A signal had a transition high-to-low.
23:0	VALUE_24 - Stores the value of the 24 least significant bits of the input APWM_TIMER the last time the State A signal had a transition high-to-low

53.2.19 APWM_AB Timer Status register

Table 732. 0x84 - APWM_AB Timer Status register

31 27	26 24	23 0
RESERVED	VALUE_3	VALUE_24
NR	0	0
r	r	r

27:24	VALUE_3 - current value of the 3 most significant bits of the input APWM_TIMER
23:0	VALUE_24 - current value of the 24 least significant bits of the input APWM_TIMER

53.3 APWM_A description

The APWM_A block is a simplified version of APWM_AB, and has only one PWM output, see figure 126. It can be used in application areas such as low-power DC/DC converters, and current or voltage sources at medium-power signal level, in general for maximum output power in the range 0.01 to 10 W. For low-power DC/DC converters, the APWM_A and its ACOMP run in peak-voltage mode without RTA, ADC and DAC. It gives an added GR716B consumption per controller of about 1mA from 1.8V supply, and about 0.5mA from 3.3V supply, which should be negligible compared to any other losses in the converter. Instead of alternative solutions such as non-SET-hard radiation-tolerant LDO ICs, these SET-hard APWM_A based converters can provide robust solutions for low-power DC/DC converters. An example of application area is to generate the auxiliary supply voltages to FPGAs, where the need of supply current is low.

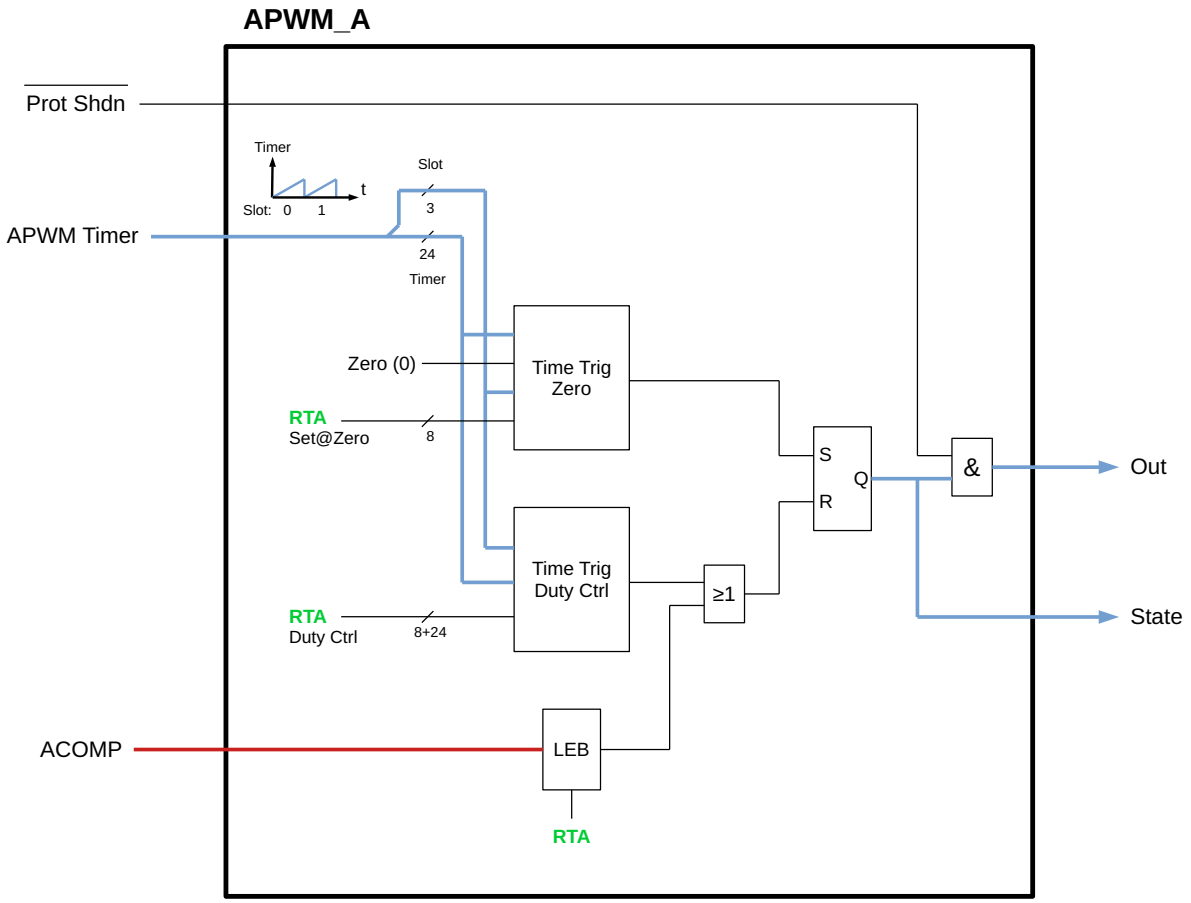


Figure 126. Functional diagram of APWM_A

A feature for APWM_A, together with its APWM Timer, is that they can be configured to work in up to 8 time slots, see the 3bit Slot count in figure 126. APWM Timer Slot configuration equal to 7 gives Slot count from 0 up to 7 and restart from 0 again. Configuration equal to 1 gives Slot count from 0 to 1 and restart from 0. Configuration equal to 0 gives Slot count equal to 0 continuously.

The APWM_A blocks have a common APWM Timer and can easily be configured to work in a certain time-slot sequence, where some APWM_A can work in sequence, some simultaneously, and some can use more than one time slot, etc. Furthermore, when multiple APWM_CF blocks are combined with one APWM_A block, each APWM_CF can control an individually regulated supply voltage. Thus, one common ACOMP, through its APWM_A, can be used to regulate a unique supply output voltage in each time slot of up to 8 slots, using up to 8 APWM_CF blocks. Examples of low-power DC/DC converters are presented in section 53.13.2 and 53.13.3.

The examples are focused on low-power DC/DC converters, but any time-varying medium-power signal voltage or current can be regulated by APWM_A. A benefit of using APWM_A instead of APWM_DAC, see section 54, is the well controlled (higher) power efficiency. The reason is that the number of switchings per second are exactly controlled by the APWM Timer, whereas for APWM_DAC, switchings per second can vary a lot and become very many. In medium-power applications this could give a disastrous drop of efficiency due to uncontrolled and high switching losses, and possibly even cause permanent damage to the power-switch devices due to overheating.

A drawback with APWM_A control is that there will be a significantly higher output switching ripple at the switching frequency. Therefore, a low-pass filter with significantly lower cut-off frequency is needed, which will give significantly lower signal bandwidth in APWM_A than APWM_DAC designs. However, when APWM_A is used to generate signal output voltage or current, there may not

be any strict requirement on peak overshoots in the same way as there are peak requirements for DC/DC converter outputs. Therefore, a network of various resistors, inductors and capacitors could be used in these APWM_A designs to obtain improved output ripple, bandwidth, etc, which will reduce the bandwidth drawback of APWM_A. Anyhow, it will depend on the specific application what type of PWM solution that is best in each application case.

53.3.1 Operation

When the analog comparator, ACOMP, is used to control (turn off) the duty cycle in a DC/DC converter, it will typically be configured to trig on a fixed output voltage level. Each PWM cycle, it turns off the duty-cycle switch instantly when the configured reference voltage level is reached, which effectively becomes a peak-voltage mode controller. Another way to control (turn off) the duty cycle is to use the 'Duty Ctrl' register in APWM_A, which can be updated PWM cycle-by-cycle from RTA. In this case, the application output signal needs to be measured by ADC, the RTA calculates the required duty cycle, which is written to the 'Duty Ctrl' register. In the following descriptions of APWM_A, control by the 'Duty Ctrl' register is regarded a complementary feature to be implemented if desired, and the main control will be implemented by ACOMP peak control. An essential reason for this control choice, when using APWM_A, is to off-load the RTAs and ADCs to free their capacity for other applications.

In cases where ADC measurements are needed on the same GPIO pin as ACOMP is used, the ADC track turn-on event can disturb the GPIO pin, see section 12.3.15 to 12.3.16. Therefore, the track turn-on time point is preferably configured to be at the power-switch turn-on point, or earlier, e.g., at the duty-cycle max limit if such is configured in the 'Duty Ctrl' register. The following leading-edge blanking (LEB) time should be configured long enough to cover the ADC Track ejection transients on the GPIO pin, which may be significant during about 0.1-0.5 us depending on GPIO-pin RC network on PCB. The Track time should be long enough to ensure accurate settling of the ADC S/H capacitor, which may be about 0.5us after LEB ends. For example, the Track time could be fixed to 1.0us, and then LEB optimized arbitrarily within 0.1-0.5 us (also considering settling of ACOMP pin).

Another ADC strategy could be to start the Track time at one power-switch turn-on point (or at preceding duty-cycle max limit) and let it continue until just before the next power-switch turn-on point. Then, the A/D conversion is executed during the next PWM period. The drawback of this ADC Track method is that it will occupy the ADC one whole PWM cycle plus the A/D conversion time, but the benefits are that it will disturb the APWM_A converter operation the least, and give the most accurate ADC result due to the best possible Track settling.

In section 53.13.2 and 53.13.3, there are examples presented how APWM_A can control cost-efficient low-power DC/DC converter topologies, based on low-cost bipolar transistors compared to expensive rad-hard MOSFETs. Note that these converters contain only short time constants and delays compared to the PWM period, except for the converter output capacitor, C_{OUT}, which is designed large enough to maintain a constant output voltage with low ripple throughout a PWM period. Thereby, since GR716B is SET hard, such topologies will be SET hard by design, because the output capacitor will low-pass filter the SETs from all short time constants in these topologies.

53.3.2 Architecture

The current value of the APWM_Timer input is readable through the APWM_A Timer Status register. The timer APWM_TIMER (APWM_A) is shared between the four APWM_A blocks. The timer is configurable and its operation is described in section 53.10.

When the 24 least significant bits of APWM_Timer are zero and the 3 most significant bits match one of the values selected in the APWM_A Set Select Configuration register, the S input of the SR flip-flop is asserted. This in turn asserts the State signal at the next clock cycle, which remains at that value until a reset condition occurs. There are two types of reset conditions:

- Duty Cycle Reset: Triggered when the 24 least significant bits of APWM_TIMER (APWM_A) match the RESET field of the APWM_A Duty Cycle Configuration register, and the 3 most sig-

nificant bits match one of the values selected in the APWM_A Reset Select Configuration register.

- Asynchronous Reset: Triggered by one of the two ACOMP inputs. The selection of the input and its corresponding value are configured in the APWM_A Asynchronous Clear Configuration register. The Leading Edge Blanking (LEB) block allows for these resets to be ignored until the LEB block completes its countdown.

When the S input is asserted, a 12-bit down-counter in the LEB block starts counting down from the PRESET value in the APWM_A Leading Edge Blanking Configuration register. It decrements by one every clock cycle and all asynchronous reset conditions are ignored until it reaches zero.

The LEB block also outputs a signal, LEB_to_ACOMP, which behaves according to the settings in the SEL field of the APWM_A Leading Edge Blanking Configuration register.

Important: The output signal, OUT, is immediately set to zero whenever the input signal $\overline{\text{Prot Shdn}}$ is asserted (Prot Shdn = 0), overriding other conditions.

The APWM_A Time Stamp Status register stores the value of the APWM_Timer input whenever the State signal has a transition high-to-low.

53.3.3 Registers

The core is controlled through registers mapped into APB address space.

*Table 733.*APWM_A registers

APB address offset	Register
0x81100000	APWM_TIMER (APWM_A) Start
0x81101000	APWM_A0 Status register
0x81101004	APWM_A0 Interrupt Status register
0x81101008	APWM_A0 Interrupt Mask register
0x8110100C	APWM_A0 Interrupt Edge register
0x81101010	APWM_A0 Set Select Configuration register
0x81101014	APWM_A0 Reset Select Configuration register
0x81101018	APWM_A0 Asynchronous Clear Configuration register
0x8110101C	APWM_A0 Leading Edge Blanking Configuration register
0x81101020	APWM_A0 Duty Cycle Configuration register
0x81101080	APWM_A0 Time Stamp Status register
0x81101084	APWM_A0 Timer Status
0x81102000	APWM_A1 Start
0x81103000	APWM_A2 Start
0x81104000	APWM_A3 Start

Note 1: APWM A1, 2 and 3 has the same registers as APWM A 0 with a different offset

53.3.4 APWM_A Status register

Table 734. 0x00 - APWM_A Status register

31	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED		E	CNE	CPE	CRE	O	AF	ASH	R	S	LEBAC	LEBSR	PS	S
NR		0	0	0	0	0	0	0	0	0	0	0	1	0
r		rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*

NOTE: The rw fields of this register can be written in order to trigger an interrupt, as per the conditions described in the APWM_A Interrupt Status register.

NOTE: The fields of this register are updated at every clock cycle, depending on the corresponding signals. These values are not latched and polling routines could easily miss signal transitions. To monitor the status of a signal it is preferable to use the Interrupt Status register.

12 Error - There has been an uncorrectable error in one of the configurations registers. For the details of the protection scheme, see APWM_A Check Bits register.

11 Counters Not Equal (NEQ) - the DNC down-counters in the LEB block have different values

10 Counters Parity Error (CPE) - there has been a parity error in the parity bits of one of the DNC down-counters in the LEB block

9 Counters Restart (CRE) - the DNC counters in the LEB block have been restarted due to an uncorrectable error

8 Out (O) - value of the signal Out in figure 126

7 ACOMP Fast (AF) - value of the input Acomp Fast in figure 126

6 ACOMP Set Hard (ASH) - value of the input Acomp Set Hard in figure 126

5 Reset (R) - value of the signal input R to the SR flip flop in figure 126

4 Set (S) - value of the signal input S to the SR flip flop in figure 126

3 LEB to ACOMP (LEBAC) - value of the LEB to ACOMP signal output of the LEB block in figure 126

2 LEB to SR (LEBSR) - If set to 1, the LEB block is not preventing asynchronous resets due to the ACOMP inputs

1 Protection Shutdown (PS) - value of signal “Prot Shdn” in figure 126

0 State (S) - value of signal “State” in figure 126

53.3.5 APWM_A Interrupt Status register

Table 735. 0x04 - APWM_A Interrupt Status register

31	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED		E	CNE	CPE	CRE	O	AF	ASH	R	S	LEBAC	LEBSR	PS	S
NR		0	0	0	0	0	0	0	0	0	0	0	1	0
r		wc	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc

A bit in this register (and an interrupt) will only be asserted if the corresponding bit in the APWM_A Interrupt Mask register is set to 1. When this condition is met, the bit will be asserted whenever the corresponding bit in the APWM_A Status register changes to match the value of the same bit in the APWM_A Interrupt Edge register.

Write 1 to clear.

53.3.6 APWM_A Interrupt Mask register

Table 736. 0x08 - APWM_A Interrupt Mask register

31	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED		E	CNE	CPE	CRE	O	AF	ASH	R	S	LEBAC	LEBSR	PS	S

Table 736. 0x08 - APWM_A Interrupt Mask register

NR	0	0	0	0	0	0	0	0	0	0	0	1	0
r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

See APWM_A Interrupt Status register.

53.3.7 APWM_A Interrupt Edge register

Table 737. 0xC - APWM_A Interrupt Edge register

31	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED		E	CNE	CPE	CRE	O	AF	ASH	R	S	LEBAC	LEBSR	PS	S
NR		0	0	0	0	0	0	0	0	0	0	0	1	0
r		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

See APWM_A Interrupt Status register.

53.3.8 APWM_A Set Select Configuration register

Table 738. 0x10 - APWM_A Set Select Configuration register

31	8	7	0
RESERVED		SET	
NR		0xFF	
r		rw	

7:0 SET - Determines the values of the 3 most significant bits of the APWM_TIMER input at which the S input of the SR flip-flop can be asserted.

Ex. SET = “00100101” -> the S input can be asserted if the 3 MSB of APWM_TIMER are equal to “000”, “011” or “101”.

53.3.9 APWM_A Reset Select Configuration register

Table 739. 0x14 - APWM_A Reset Select register

31	8	7	0
RESERVED		RESET	
NR		0xFF	
r		rw	

7:0 RESET - Determines the values of the 3 most significant bits of the APWM_TIMER input at which the R input of the SR flip-flop can be asserted.

Ex. SET = “00100101” -> the R input can be asserted if the 3 MSB of APWM_TIMER are equal to “000”, “011” or “101”.

53.3.10 APWM_A Asynchronous Clear Configuration register

Table 740. 0x18 - APWM_A Asynchronous Clear Configuration register

31	4	3	2	1	0
RESERVED		RES	EAC	SEL	
NR		0	0	0	
R		rw	rw	rw	

3 RES - Only write 0 to this field.

2 Enable Asynchronous Clear (EAC) - Set to 1 to enable clearing the State A signal using the signals selected through the field SEL of this register

1:0 Select which signal and which value of it can trigger an asynchronous clear of the State signal

“00” -> ACOMP Set Hard = 1

“01” -> ACOMP Set Hard = 0

“10” -> ACOMP Fast = 0

“11” -> ACOMP Fast = 1

53.3.11 APWM_A Leading Edge Blanking Configuration register

Table 741. 0x1C - APWM_A Leading Edge Blanking Configuration register

31 24	23 14	13 12	11 0
RESERVED		SEL	PRESET
NR		0	0
r		rw	rw

13:12 SEL - Configured the behavior of the output LEB to ACOMP.

00 - LEB to ACOMP is always set to 1

01 - LEB to ACOMP is set to 0 for the time interval from when the LEB down-counter starts counting until it reaches zero. It is set to 1 otherwise.

10- Same as for “01”

11 - LEB to ACOMP is set to 0 for the time interval from when the LEB down-counter starts counting until it reaches zero. It is then set to 1 if the State signal is set to1 and it is kept to 1 until the State signal goes to 0.

11:0 PRESET - Leading Edge Blanking Down-counter Preset value

53.3.12 APWM_A Duty Cycle configuration register

Table 742. 0x20 - APWM_A Duty Cycle configuration register

31 27	26 24	23 0
RESERVED		RESET
NR		0
r		r

23:0 RESET - Determine the value of the 24 least significant bits of the APWM_TIMER input at which the R input of the SR flip-flop can be asserted.

53.3.13 APWM_A Time Stamp Status register

Table 743. 0x80 - APWM_A Time Stamp Status register

31 27	26 24	23 0
RESERVED	VALUE_3	VALUE_24
NR	0	0
r	r	r

27:24 VALUE_3 - Stores the value of the 3 most significant bits of the input APWM_TIMER the last time the State signal had a transition high-to-low.

23:0 VALUE_24 - Stores the value of the 24 least significant bits of the input APWM_TIMER the last time the State signal had a transition high-to-low

53.3.14 APWM_A Timer Status register

Table 744. 0x84 - APWM_A Timer Status register

31 27	26 24	23 0
RESERVED	VALUE_3	VALUE_24
NR	0	0
r	r	r

27:24 VALUE_3 - current value of the 3 most significant bits of the input APWM_TIMER

23:0 VALUE_24 - current value of the 24 least significant bits of the input APWM_TIMER

53.4 APWM_CF description

APWM_CF is a general PWM block configurable for a wide variety of PWM applications. It can work as extension on APWM_A or AB, using a state input signal, see top left in figure 127. Or, it can work standalone only controlled by Timers and Duty control, which can be updated PWM cycle-by-cycle from RTA or the main LEON3FT.

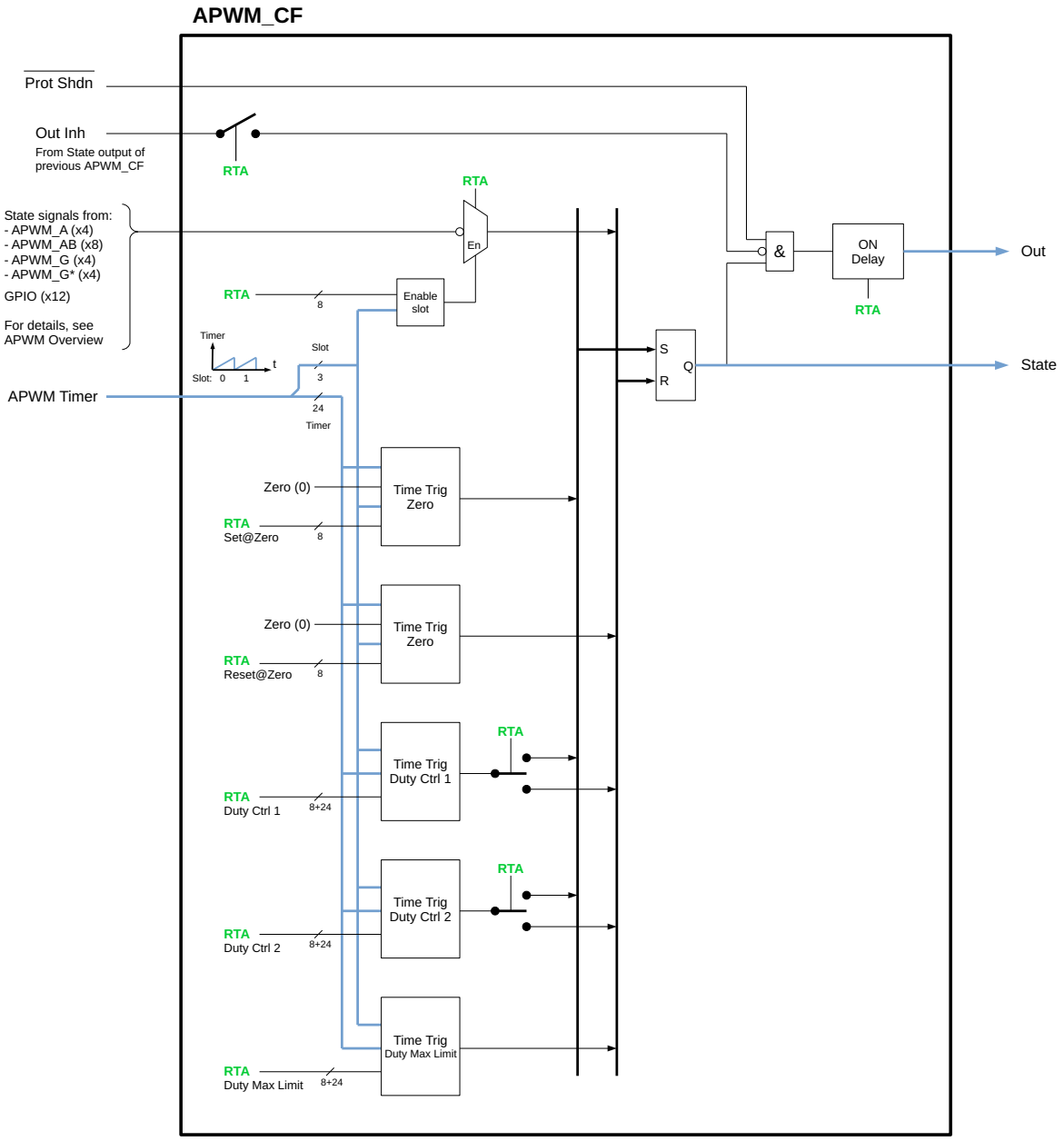


Figure 127. Functional diagram of APWM_CF

53.4.1 Operation

The registers for Timers and Duty control are implemented with double buffering, to support synchronized update of all PWM parameters for all APWM_CF blocks simultaneously. The synchronization time point when writing takes effect is configurable in each APWM_CF by event registers and synchronization enable/disable flags controlled from RTA and the main LEON3FT.

Four APWM_CF blocks use one Timer, see figure 121. If more than four APWM_CF blocks should run on identical Timer settings, another Timer for another four APWM_CF blocks is configured identically. This will ensure identical PWM period time and delay position (phase position) on system-clock cycle level, controlled by a Main Timer Tick which is the same selected (configured) for both APWM Timers.

When APWM_CF is configured as extension on APWM_A or AB, it is typically intended for DC/DC converter topologies that need other than the two PWM outputs from an APWM_AB block. For example, full-bridge topologies such as phase-shifted full-bridge converters (PSFB) need to use APWM_CF. Often APWM_G blocks can be useful also, see section 53.5, which can be configured as further extension on APWM_CF and APWM_AB. See section 53.13.4 for a PSFB controller example.

When APWM_CF is configured to work standalone, both turn-on and off time points can be controlled independently, by writing to Timers and Duty control PWM cycle-by-cycle from RTA, see "RTA" settings to the left in figure 127. This flexible PWM operational mode can be used in many different application areas, such as motor control, voltage-mode DC/DC control, etc.

Motor control can be done in several different ways. For simple motor control, direct control of GPIO outputs can be used. For current-regulation control, PWM control is needed, and this is where APWM_CF is most suitable. See section 53.13.5 for such a motor control example. For very fast settling of winding currents after commutation time points, GR716B supports peak-current control of winding currents by combining APWM_CF with APWM_AB, in a similar way as in the DC/DC controller example in section 53.13.4.

53.4.2 Registers

The core is programmed through registers mapped into APB address space.

*Table 745.*APWM_CF registers

APB address offset	Register
0x81301000	APWM CF0 0 Interrupt register
0x81301004	APWM CF0 0 Interrupt Status register
0x81301008	APWM CF0 0 Mask register
0x8130100C	APWM CF0 0 Interrupt level register
0x81301010	APWM CF0 0 Enable use of lower state
0x81301014	APWM CF0 0 Configuration of On delay counter
0x81301018	APWM CF0 0 Select disable input state for PWM
0x8130101C	APWM CF0 0 Configure internal disable state
0x81301020	APWM CF0 0 Configure time-slots multiplexer (CMPE)
0x81301024	APWM CF0 0 Enable CMPE i.e. set@zero block
0x81301028	APWM CF0 0 Configure time-slots multiplexer (CMPE)
0x8130102C	APWM CF0 0 Enable CMPE i.e. Reset@zero block
0x81301030	APWM CF0 0 Count Cnf 0
0x81301034	APWM CF0 0 Select CmpEQ 1
0x81301038	APWM CF0 0 Count Cnf 1
0x8130103c	APWM CF0 0 Select CmpEQ 2
0x81301040	APWM CF0 0 Max duty check
0x81301044	APWM CF0 0 RTA Sync setup 0 and 1
0x81301048	APWM CF0 0 RTA Sync Reg 0
0x8130104c	APWM CF0 0 RTA Sync Reg 1
0x81301080	APWM CF0 0 Counter input value
0x81301084	APWM CF0 0 Input state value
0x81301088	APWM CF0 0 RTA Duty cycle output value 0
0x8130108C	APWM CF0 0 RTA Duty cycle output value 1
0x813010E0	APWM CF0 0 Local register interface information
0x81302000 - 0x813020FF [1)]	APWM CF0 1
0x81303000 - 0x813030FF [1)]	APWM CF0 2
0x81304000 - 0x813040FF [1)]	APWM CF0 3
0x81306000 - 0x813060FF [1)]	APWM CF1 0
0x81307000 - 0x813070FF [1)]	APWM CF1 1
0x81308000 - 0x813080FF [1)]	APWM CF1 2
0x81309000 - 0x813090FF [1)]	APWM CF1 3
0x81401000 - 0x813010FF [1)]	APWM CF2 0
0x81402000 - 0x813020FF [1)]	APWM CF2 1
0x81403000 - 0x813030FF [1)]	APWM CF2 2
0x81404000 - 0x813040FF [1)]	APWM CF2 3

Note 1: APWM CF0 123, CF1 0123 and CF2 0123 has the same registers as APWM CF0 0 with a different offset

53.4.3 APWM CF Interrupt register

Table 746. 0x00 - CFINT - APWM CF Interrupt register

31 … 8	7	6	5	4	3	2	1	0
RESERVED	P	OD	MD	SO	SP	OI	PS	SO
0x0	0	0	0	0	1	0	0	0
r	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*

31: 8 RESERVED

7: Configuration parity error status (P). This parity check covers the 10 configuration registers. Becomes '1' when a configuration parity error is detected.

6: On delay out (OD)

5: Max duty (MD)

4: State CF Out (SO) - State CF out

3: State Prev in (SP)

2: Out Inh (OI): State output of previous/lower number, within the group of 4 APWM_CF blocks

1: PROT (PS) - Protection shutdown (not APWM_CFi.ProtShdn_n)

0: State CF Out (SO)

For test-purposes, bits 7:0 in this register are writable. The written data overrides the normal data source for exactly one system clock cycle. This provides a means for triggering the associated interrupt from software.

NOTE: The fields of this register are updated at every clock cycle, depending on the corresponding signals. These values are not latched and polling routines could easily miss signal transitions. To monitor the status of a signal it is preferable to use the Interrupt Status register.

53.4.4 APWM CF Interrupt Status register

Table 747. 0x04 - CFISTAT - APWM CF Interrupt Status register

31 … 8	7	6	5	4	3	2	1	0
RESERVED	P	OD	MD	SO	SP	OI	PS	SO
0x0	0	0	0	0	0	0	0	0
r	wc	wc	wc	wc	wc	wc	wc	wc

31: 8 RESERVED

7: 0 Bit *x* of this register is set to 1 whenever an interrupt has been generated due to bit *x* of CFINT. Each bit in this stays set to 1 until software clears it by writing 1.

Write 1 to clear.

53.4.5 APWM CF Interrupt Mask register

Table 748. 0x08 - CFMSK - Interrupt Mask register

31 … 8	7	6	5	4	3	2	1	0
RESERVED	P	OD	MD	SO	SP	OI	PS	SO
0x0	0	0	0	0	0	0	0	0
r	rw	rw	rw	rw	rw	rw	rw	rw

31: 8 RESERVED

7: 0 Bit *x* of this register controls whether the event corresponding to bit *x* of CFINT can generate an interrupt or not. When bit *x* is 0, that event will not generate interrupts.

53.4.6 APWM CF Interrupt Edge register

Table 749. 0xC - CFEDGE - APWM CF Interrupt Edge register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								P	OD	MD	SO	SP	OI	PS	SO
0x0																								1	1	1	1	1	1	1	1
r																								rw	rw	rw	rw	rw	rw	rw	rw

31: 8 Reserved

7: 0 Bit *x* of this registers controls the polarity of interrupt generation from bit *x* of the APWMCFINT register.
0 => falling edge: interrupt generated when bit *x* changes from 1 to 0
1 => rising edge: interrupt generated when bit *x* changes from 0 to 1

53.4.7 APWM CF Enable use of lower state register

Table 750. 0x10 - OUTINH- APWM CF Enable use of lower state register

31	1	0
RESERVED		EN
0x0		0
r		rw

31: 1 RESERVED

0: EN: Enable of "Out Inh"

53.4.8 APWM CF Configuration of On delay counter register

Table 751. 0x14 - ODCNT - APWM CF Configuration of On delay counter register

31	12	11	0
RESERVED		CNT12	
0x0		0xFFF	
r		rw	

31:12 RESERVED

11:0 CNT12: On delay counter value

53.4.9 APWM CF Select disable input state for PWM register

Table 752. 0x18 - SELDIS - APWM CF Select disable input state for PWM register

31	6	5	0
RESERVED		SEL	
0x0		0x000	
r		rw	

31:6 RESERVED

5:0 SEL: Select signal for the multiplexer selecting among the STATE_x and GPIO inputs

0 to 3: State_AB_A

4 to 7: State_AB_B

8 to 11: State_A

12 to 15: State_G

16 to 19:

State_G*(3 downto 0) for CF0,

State_G*(7 downto 4) for CF1,

State_G*(11 downto 8) for CF2

20 to 31:

GPIO(12 downto 1) for CF0,

GPIO(24 downto 13) for CF1,

GPIO(36 downto 25) for CF2,

53.4.10 APWM CF Configure internal disable state register

Table 753. 0x1C - INTDIS - APWM CF Configure internal disable state register

31	8	7	0
RESERVED		CIDS	
0x0		0xFF	
r		rw	

Table 753. 0x1C - INTDIS - APWM CF Configure internal disable state register

31:8	RESERVED
7:0	CIDS: Cnf input to enable/disable the state depending on the time slots given by the 3 bit timer

53.4.11 APWM CF Configure time-slots multiplexer (CMPE Set@Zero) register

Table 754. 0x20 - TSCMPES - APWM CF Configure time-slots multiplexer (CMPE Set@Zero) register

31 8	7 0
RESERVED	SEL
0x0	0x000
r	rw

31:8 RESERVED

7:0 VAL: CMPE Set@Zero

53.4.12 APWM CF Enable CMPE i.e. Set@Zero block register

Table 755. 0x24 - ENCMPES - APWM CF Enable CMPE i.e. Set@Zero block register

31 1	0
RESERVED	EN
0x0	0
r	rw

31: 1 RESERVED

0: EN: Enable

53.4.13 AAPWM CF Configure time-slots multiplexer (CMPE Reset@Zero) register

Table 756. 0x28 - TSCMPER - APWM CF Configure time-slots multiplexer (CMPE Reset@Zero) register

31 8	7 0
RESERVED	SEL
0x0	0x000
r	rw

31:8 RESERVED

7:0 VAL: CMPE Reset@Zero

53.4.14 APWM CF Enable CMPE i.e. Reset@Zero block register

Table 757. 0x2C - ENCMPER - APWM CF Enable CMPE i.e. Reset@Zero block register

31 1	0
RESERVED	EN
0x0	0
r	rw

31: 1 RESERVED

0: EN: Enable

53.4.15 APWM CF Count Cnf 0 register

Table 758. 0x30 - CNTCNF0 - APWM CF Count Cnf 0 register

31 24	23 0
EN	DC
0	0
rw	rw

31: 24 EN: Enable time slots

23: 0 DC: Set Duty Cycle

53.4.16 APWM CF Select CmpEQ_1 register

Table 759. 0x34 - SELEQ1 - APWM CF Select CmpEQ_1 register

31	3	2	0
RESERVED		CNF	
0		0x0	
r		rw	

31:3 RESERVED

2:0 CNF: CmpEq_Ctrl_1

53.4.17 APWM CF Count Cnf 1 register

Table 760. 0x38 - CNTCNF1 - APWM CF Count Cnf 1 register

31	24	23	0
EN		DC	
0		0	
rw		rw	

31: 24 EN: Enable time slots

23: 0 DC: Set Duty Cycle

53.4.18 APWM CF Select CmpEQ_2 register

Table 761. 0x3C - SELEQ2 - APWM CF Select CmpEQ_2 register

31	3	2	0
RESERVED		CNF	
0		0x0	
r		rw	

31:3 RESERVED

2:0 CNF: CmpEq_Ctrl_2

53.4.19 APWM CF Max duty check register

Table 762. 0x40 - MAXDUTY - APWM CF Max duty check register

31	24	23	0
EN		DC	
0		0	
rw		rw	

31: 24 EN: Enable time slots

23: 0 DC: Set Duty Cycle

53.4.20 APWM CF RTA Sync Setup 0 and 1 register

Table 763. 0x44 - SYNC - APWM CF Sync Setup 0 and 1 register

31	15	8	7	0
RESERVED	DCTRL2		DCTRL1	
0	0x00		0x00	
r	rw		rw	

Table 763. 0x44 - SYNC - APWM CF Sync Setup 0 and 1 register

15:8	Sync Setup1 y0--y6: Sync at LocalTick OR CmpXXX, if RangeChkOK x7: Sync immediately, if RangeChkOK y8--yE: Sync at LocalTick OR CmpXXX xF: Sync immediately y: Mask for CmpXXX x: Don’t care
7:0	Sync Setup0 y0--y6: Sync at LocalTick OR CmpXXX, if RangeChkOK x7: Sync immediately, if RangeChkOK y8--yE: Sync at LocalTick OR CmpXXX xF: Sync immediately y: Mask for CmpXXX x: Don’t care

53.4.21 APWM CF RTA Sync Reg 0 register

Table 764. 0x48 - RTA0SYNC - APWM CF RTA Sync Reg 0 register

31 27	26 0
R	SYNC_VAL0
0	0
r	rw

27:0	SYNC_VAL0: Synchronized RTA Value

53.4.22 APWM CF RTA Sync Reg 1 register

Table 765. 0x4C - RTA1SYNC - APWM CF RTA Sync Reg 1 register

31 27	26 0
R	SYNC_VAL1
0	0
r	rw

27:0	SYNC_VAL1: Synchronized RTA Value

53.4.23 APWM CF Counter input value register

Table 766. 0x80 - CNT_STAT - APWM CF Counter input value register

31 30 29 28 27 26 25 24 23 0

R	UPCNT_ 3	UPCNT_ 23
0	0	0
r	rw	rw

26:24 UPCNT_3 - current value of the 3-bits up-counter

23:0 UPCNT_ 23 - current value of the 29-bits up-counter

53.4.24 APWM CF Input state value register

Table 767. 0x84 - STATE_STAT - APWM CF Counter input state register

31 30 29 28 27 26 25 24 23 20	19 16	15 12	11 8	7 4	3 0
GPIO	State_G*	State_G	State_A	State_AB_B	State_AB_A
0	0	0	0	0	0
rw	rw	rw	rw	rw	rw

31:0 Input state value

16 to 19:

State_G*(3 downto 0) for CF0,

State_G*(7 downto 4) for CF1,

State_G*(11 downto 8) for CF2

20 to 31:

GPIO(12 downto 1) for CF0

GPIO(24 downto 13) for CF1

GPIO(36 downto 25) for CF2

53.4.25 APWM CF RTA Duty cycle output value 0 register

Table 768. 0x88 - TIMERVAL - APWM CF RTA Duty cycle output value 0 register

31 24	23 0
DutyCtr	DutyMax
0	0
r	r

31:24 DutyCtr

23:0 DutyMax

53.4.26 APWM CF RTA Duty cycle output value 1 register

Table 769. 0x8C - TIMERVAL - APWM CF RTA Duty cycle output value 1 register

31 24	23 0
DutyCtr	DutyMax
0	0
r	r

31:24 DutyCtr

23:0 DutyMax

53.4.27 APWM CF Local register interface information

Table 770. 0xE0 - INFO - APWM CF Local register interface information

31 24	23 16	15 8	7 0
IRQ	CFG	STAT	IRQ
0x08	0x10	0x02	0x08
r	r	r	r

31: 24 IRQ: Number of Interrupts (same as 7:0)

23: 16 CFG: Number of Configuration registers

15:8 STAT: Number of Status registers

7:0 IRQ: Number of Interrupts

53.5 APWM_G description

The main functionality of APWM_G is to provide configurable combinational logic based on the ‘state’, ACOMP and GPIO input signals, see figure 128 and figure 121. The combinational logic is flexibly configurable to support control of a wide variety of applications including motor control, complex switching DC/DC converters such as phase-shifted full bridge (PSFB) with synchronous rectification, and many other application areas.

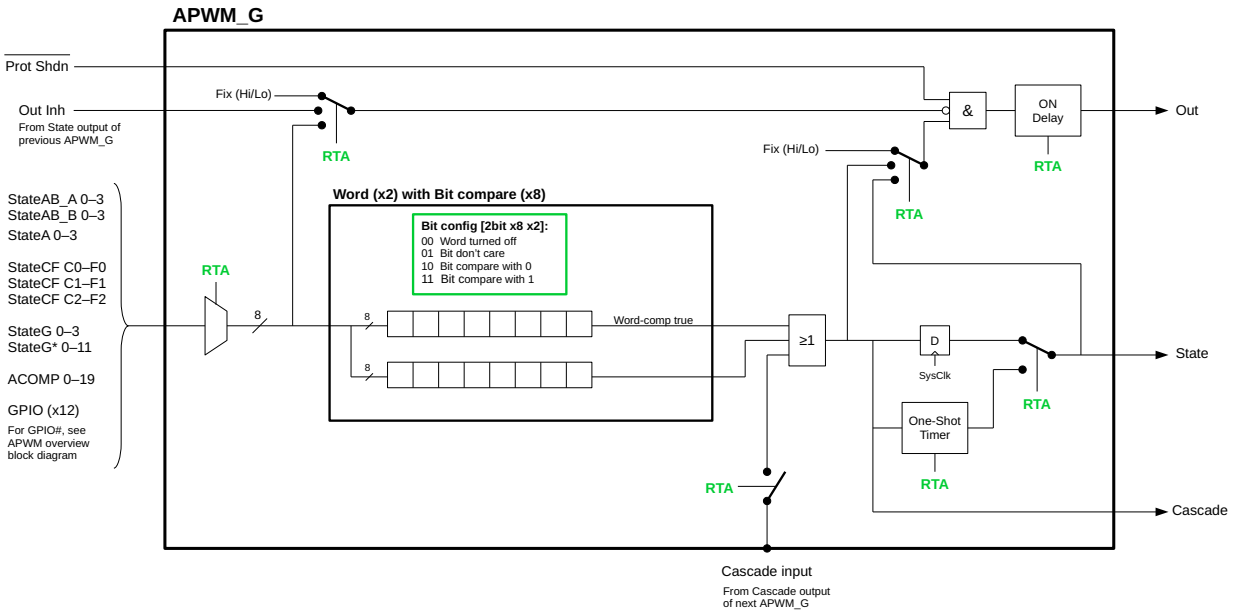


Figure 128. Functional diagram of APWM_G. Each row is an 8-bit word, and word-compare is true if all 8 bit positions compare to true.

All APWM_G blocks (x16) have an ON-Delay block at the GPIO output port ‘Out’. Four of the APWM_G blocks have a One-Shot Timer block, and the ones without One-Shot Timer are entitled ‘APWM_G*’ (x12). An APWM_G block can work standalone with ACOMP and GPIO signals directly connected (configured) as input signals, or work together with APWM_AB, APWM_A and APWM_CF with their state signals connected as inputs, or work with a combination of different inputs.

53.5.1 Operation

The combinational logic is structured to execute compare on 8 selected input signals, see to the left in figure 128. There are two rows of such 8-bit comparisons, of which one true row is enough to generate high on the APWM_G state output. Word-compare true signal is high only when all 8 bit positions are compared to true. Configuration for each bit position is 2 bits, where each position can be configured to compare with logic '0' or '1', or set to don't care. If at least one position in the word is configured to '00', this word is turned off, i.e., its Word-compare true output is forced low.

If two rows would not be enough for a certain application, then neighbor APWM_G blocks can be configured to work together, see the cascade port in figure 128. This will effectively extend the OR function in APWM_G with more rows, extended proportionally to the number of APWM_G blocks that are cascaded. Different ways how to extend or use APWM_G blocks are further described in the following examples.

An example how to extend APWM_G to more than 8 effective bits per word is presented here. The bit compare function within a word is an AND function of all 8 bit positions. Upto two inputs of the same type can be connected (configured) into one APWM_G block, e.g., two StateCF and two ACOMP/GPIO, see table 53.5.2. If an application needs more than two inputs of the same type into the word AND function, multiple APWM_G blocks can be connected in cascade. It can be done either by connecting the cascade output signal of the neighbor APWM_G block into the OR gate, or by connecting any state output into the mux input, see figure 129 and 128. A difference between these two cascade methods is that the state output has a pipeline stage (D flip-flop running on system clock, giving upto 10ns delay at 100MHz), whereas the cascade output does not have any pipeline stage. This pipeline D flip-flop constitutes a desired memory function in the second example below, but may cause an undesired delay in other applications.

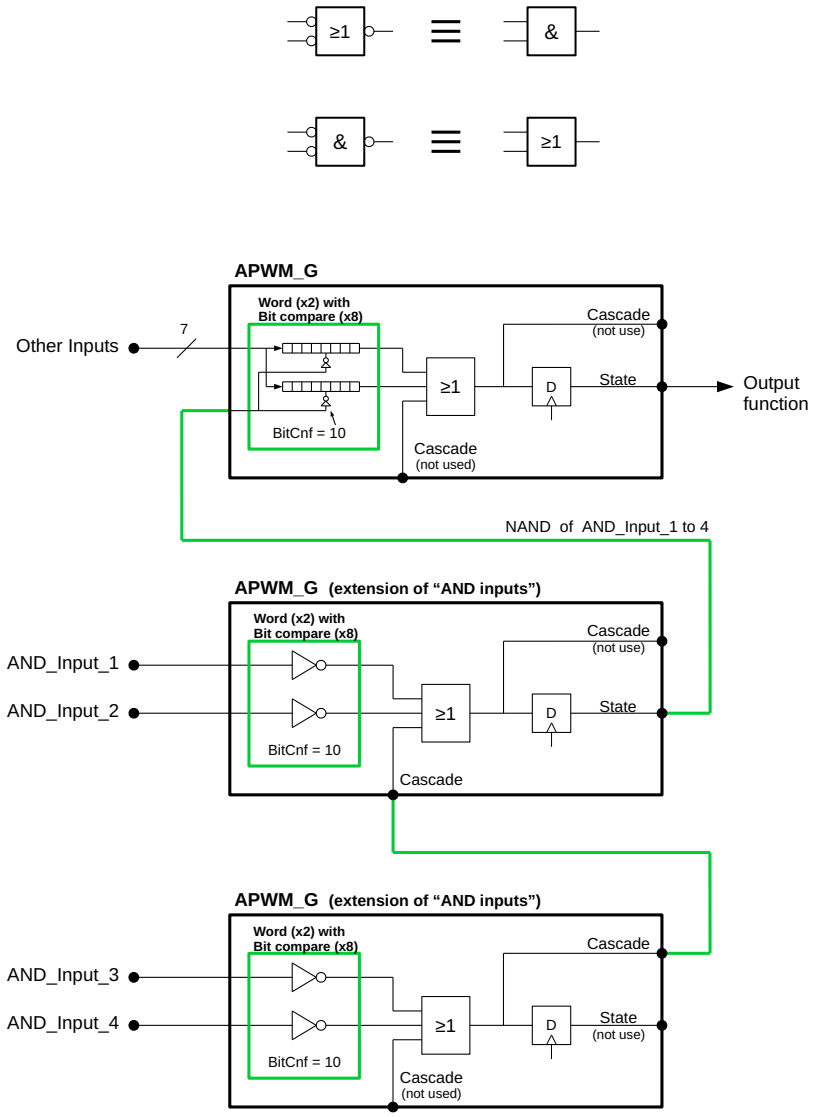


Figure 129. Multiple APWM_G blocks linked together by cascade and state signals. At the top is a reminder of logic gate equivalents.

From logic truth table it follows that an OR gate with all inputs inverted and the output inverted is equivalent with an AND function, see top of figure 129. This will be used to create an extended word AND function here. At the OR gate in APWM_G, multiple APWM_G blocks can be cascaded, which will effectively increase the number of OR inputs in proportion to the number of cascaded blocks. See the cascade signal between the two bottom APWM_G blocks in figure 129.

To convert these cascaded OR gates to an AND function, each OR-gate input signal is inverted, i.e., configured to compare with zero (BitCnf=10). This is the only function executed by this whole word function, when the purpose is to create this kind of AND function through the OR gates. Therefore, the other seven bit positions in each word are configured to don't care (BitCnf=01). Since there are two words per APWM_G block, and each of them can invert an input signal, one APWM_G block supports two input signals. Moreover, if the application would need other logic functions that happens to fit into the other seven bit positions, they are inserted into the bit configurations as desired, instead of setting these positions to don't care.

The state output of the middle APWM_G block is connected to the mux input of the top block. This mux input signal is inverted when it goes into the word AND function, i.e., configured to compare

with zero (BitCnf=10). With this inversion, the inputs, AND_Input_1 to 4, are incorporated correctly as extended AND function into the word AND function of the top block. Thereby, the word function will be an AND function of the four input signals, AND_Input_1 to 4, and the other seven bit positions in each word of the top block. And each of these two words can have their seven bits, respectively, configured independently. Hence, this configuration implementation has effectively extended the word length of the top APWM_G block with the inputs AND_Input_1 to 4, with the cost of using more APWM_G blocks.

It depends on the application if the pipeline stage (undesired here), causing 10ns delay to the state output, matters or not. Where it matters, other input signals into the top APWM_G block that need correlated delay could be configured through another APWM_G block first, which introduces one pipeline stage. Alternatively, a low-pass RC time constant (>~30ns, for 100MHz system clock) can be added on PCB where this output signal goes out on a GPIO output.

Instead of the state output there is another signal path available, which does not introduce any pipeline delay, but it occupies a GPIO pin. The OR gate output inside the middle APWM_G block can be configured through the ON-Delay block out on a GPIO pin, without any delay, see figure 128. This GPIO pin can always be read back by its schmitt input, which can be connected to the mux input of the top APWM_G block. This replaces the state signal from the middle block. If this GPIO pin is left open, or only loaded with resistive load without any noticeable capacitive load, the added delay will be in the order of 1ns. This configuration will effectively work as if the wire from the state output of the middle block in figure 129 is moved to the cascade output, but with a short additional delay of 1ns. Hence, this configuration provides the extended AND function without any pipeline delay, but with the cost of occupying another GPIO pin.

Another example is to configure an APWM_G block as a synchronously clocked set/reset flip-flop (SR-FF). The D flip-flop in figure 130 is used as the SR-FF memory cell, by connecting the state output back to the mux input of the same APWM_G block.

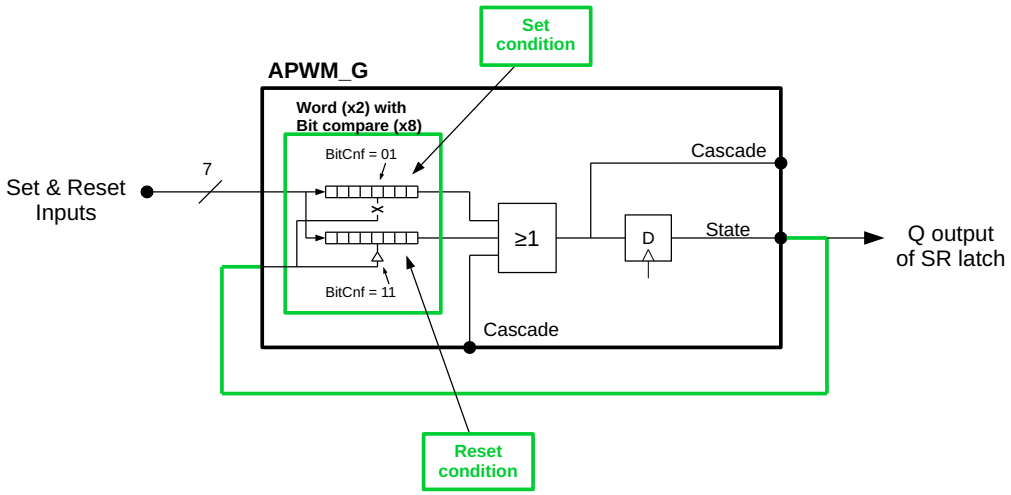


Figure 130. Set-dominant synchronously clocked SR flip-flop.

In this example, the top word is configured to be the set condition. The bottom word is configured to hold the SR-FF memory state in the D flip-flop, by connecting the state output back to the mux input, and configuring this input to compare with one (BitCnf=11). This will permanently lock high state after first clock cycle with a high state signal, assuming the other seven bits are configured to don't care (BitCnf=01). To be able to reset this locked high state, the bottom word needs to also contain a reset condition, configured in the other seven bits. An active reset condition is when at least one of these seven bits goes into the word AND function as low.

If the reset condition only is put in the bottom word, as in figure 130, this will be a set-dominant SR-FF. If the reset condition also is put in the top word, disabling the set function when reset condition is fulfilled, this will be a reset-dominant SR-FF. Instead of putting the reset condition in the top word, the don't care configuration (BitCnf=01) in figure 130 can be changed to compare with zero (BitCnf=10). This will provide an SR-FF that toggles each system clock, when both set and reset conditions are fulfilled continuously. This is traditionally called a JK flip-flop.

The above APWM_G configuration examples present rather simple straightforward logic functions. An example how to further combine functions is to extend the set and reset conditions in the SR-FF example by the word extension method in figure 129. Then, the top APWM_G block constitutes the SR-FF block.

In general, numerous variations of arbitrary logic functions can be defined in multiple APWM_G blocks, where upto 4 APWM_G blocks (with one-shot timer) resp 12 APWM_G* blocks (without one-shot timer) can be connected in cascade. Furthermore, any state signals throughout such cascade chains can be connected to mux inputs of any APWM_G blocks or connected to GPIO outputs through the ON-Delay block. So this configuration flexibility of APWM_G should be able to support a great variety of application situations.

53.5.2 Registers

The core is programmed through registers mapped into APB address space.

*Table 771.*APWM G registers

APB address offset	Register
0x81500000	APWM G 0 Interrupt register
0x81500004	APWM G 0 Interrupt status register
0x81500008	APWM G 0 Mask register
0x8150000C	APWM G 0 Interrupt edge register
0x81500010	APWM G 0 On delay timer configuration
0x81500014	APWM G 0 CmpG_word configuration
0x81500018	APWM G 0 Enable input and state
0x8150001C	APWM G 0 Select state 0 to 3
0x81500020	APWM G 0 Select state 4 to 7
0x81500024	APWM G 0 Select output state source
0x81500028	APWM G 0 Enable One Shot trigger
0x8150002C	APWM G 0 Preset value for One Shot timer
0x81500030	APWM G 0 Configure One Shot output
0x81500080	APWM G 0 Input states 31 downto 0
0x81500084	APWM G 0 Input states 63 downto 32
0x815000E0	APWM G 0 Local register interface information
0x81501000 - 0x811010FF [1)]	APWM G 1
0x81502000 - 0x811020FF [1)]	APWM G 2
0x81503000 - 0x811030FF [1)]	APWM G 3

Note 1: APWM G 1, 2 and 3 has the same registers as APWM G 0 with a different offset

53.5.3 Registers

The core is programmed through registers mapped into APB address space

*Table 772.*APWM G* registers

APB address offset	Register
0x81504000	APWM G* 0 Interrupt register
0x81504004	APWM G* 0 Interrupt Status register
0x81504008	APWM G* 0 Mask register
0x8150400C	APWM G* 0 Interrupt edge register
0x81504010	APWM G* 0 On delay timer configuration
0x81504014	APWM G* 0 CmpG_word configuration
0x81504018	APWM G* 0 Enable input and state
0x8150401C	APWM G* 0 Select state 0 to 3
0x81504020	APWM G* 0 Select state 4 to 7
0x81504024	APWM G* 0 Select output state source
0x81504080	APWM G* 0 Input states 31 downto 0
0x81504084	APWM G* 0 Input states 63 downto 32
0x815040E0	APWM G* 0 Local register interface information
0x815040F0	APWM G* 0 Check bits for configuration register 0 to 7
0x815040F4	APWM G* 0 Check bits for configuration register 8 to 15
0x815040F8	APWM G* 0 Status check bits for configuration register 0 to 7
0x815040FC	APWM G* 0 Status check bits for configuration register 8 to 15
0x81505000 - 0x811050FF [1)]	APWM G* 1
0x81506000 - 0x811060FF [1)]	APWM G* 2
0x81507000 - 0x811070FF [1)]	APWM G* 3
0x81508000 - 0x811080FF [1)]	APWM G* 4
0x81509000 - 0x811090FF [1)]	APWM G* 5
0x8150A000 - 0x8110A0FF [1)]	APWM G* 6
0x8150B000 - 0x8110B0FF [1)]	APWM G* 7
0x8150C000 - 0x8110C0FF [1)]	APWM G* 8
0x8150D000 - 0x8110D0FF [1)]	APWM G* 9
0x8150E000 - 0x8110E0FF [1)]	APWM G* 10
0x8150F000 - 0x8110F0FF [1)]	APWM G* 11

Note 1: APWM G* 1 to 11 has the same registers as APWM G* 0 with a different offset

53.5.4 APWM G Interrupt register

Table 773. 0x00 - GINT - APWM G Interrupt register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												P	CIA	CG1	CG0	TA	OS	CG	OD	SI3	SI2	SI1	SI0	SI3	SI2	SI1	SI0	CI	OI	PS	SG
0x0												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r												rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*

31: 20 RESERVED

19: Configuration parity error status (P). This parity check covers the 9 configuration registers. Becomes '1' when a configuration parity error is detected.

18: CmpInAnd (CIA) - CmpG In enable

17: CmpG8o1 (CG1) - CmpG output

16: CmpG8o0 (CG0) - CmpG output

15: Timer Alarm (TA) - OneShot Timer Alarm

14: OneShotOut (OS) - Output from One Shot timer

13: CmpG8Out (CG) - Internal state (before one shot timer)

12: On delay out (OD) - Output from ON DELAY block

11: Statei7 (SI7) - State input 7

State input as per the configuration selected in register SELSTAT4-7: ACOMP_II/GPIO_II

10: Statei6 (SI6) - State input 6

State input as per the configuration selected in register SELSTAT4-7: ACOMP_I/GPIO_I

9: Statei5 (SI5) - State input 5

State input as per the configuration selected in register SELSTAT4-7: STATE_G_II

8: Statei4 (SI4) - State input 4

State input as per the configuration selected in register SELSTAT4-7: STATE_G_I

7: Statei3 (SI3) - State input 3

State input as per the configuration selected in register SELSTAT0-3: STATE_CF_II

6: Statei2 (SI2) - State input 2

State input as per the configuration selected in register SELSTAT0-3: STATE_CF_I

5: Statei1 (SI1) - State input 1

State input as per the configuration selected in register SELSTAT0-3: STATE_A/AB_II

4: Statei0 (SI0) - State input 0

State input as per the configuration selected in register SELSTAT0-3: STATE_A/AB_I

3: CmpInp (CI): CompOut from next G block. CmpInp 'last': tied to low. CmpOut 'first': Not used.

Similarly for G*.

2: Out Inh (OI): State from previous G block, OutInh G 'first' from State G 'last'.

Similarly for G*.

1: PROT (PS) - Protection shutdown input to PWM

0: State G(SG)

For test-purposes, bits 19:0 in this register are writable. The written data overrides the normal data source for exactly one system clock cycle. This provides a means for triggering the associated interrupt from software.

NOTE: The fields of this register are updated at every clock cycle, depending on the corresponding signals. These values are not latched and polling routines could easily miss signal transitions. To monitor the status of a signal it is preferable to use the Interrupt Status register.

53.5.5 APWM G Interrupt Status register

Table 774. 0x04 - GISTAT - APWM G Interrupt Status register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												P	CIA	CG1	CG0	TA	OS	CG	OD	SI3	SI2	SI1	SI0	SI3	SI2	SI1	SI0	CI	OI	PS	SG

Table 774. 0x04 - GISTAT - APWM G Interrupt Status register

0x0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc

31: 20 RESERVED

19: 0 Bit *x* of this register is set to 1 whenever an interrupt has been generated due to bit *x* of GINT. Each bit in this stays set to 1 until software clears it by writing 1.

Write 1 to clear.

53.5.6 APWM G Interrupt Mask register

Table 775. 0x08 - GMSK - Interrupt Mask register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												P	CIA	CG1	CG0	TA	OS	CG	OD	SI3	SI2	SI1	SI0	SI3	SI2	SI1	SI0	CI	OI	PS	SG
0x0												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r												rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

31: 8 RESERVED

7: 0 Bit *x* of this register controls whether the event corresponding to bit *x* of GINT can generate an interrupt or not. When bit *x* is 0, that event will not generate interrupts.

53.5.7 APWM G Interrupt Edge register

Table 776. 0xC - GEDGE - APWM G Interrupt Edge register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												P	CIA	CG1	CG0	TA	OS	CG	OD	SI3	SI2	SI1	SI0	SI3	SI2	SI1	SI0	CI	OI	PS	SG
0x0												1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
r												rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

31: 8 Reserved

7: 0 Bit *x* of this registers controls the polarity of interrupt generation from bit *x* of the GINT register.
0 => falling edge: interrupt generated when bit *x* changes from 1 to 0
1 => rising edge: interrupt generated when bit *x* changes from 0 to 1

53.5.8 APWM G On delay timer configuration register

Table 777. 0x10 - OnDelay Cfg - APWM G On delay timer configuration register

31 12	11 0
RESERVED	CNT12
0x0	0xFFF
r	rw

31:12 RESERVED

11:0 CNT12: OnDelay Cfg

53.5.9 APWM G CmpG_word configuration register

Table 778. 0x14 - CmpGWordCnf - APWM G CmpG_word configuration register

31 16	15 0
RTA_CmpG_8_Conf1	RTA_CmpG_8_Conf0
0x0000	0x0000
rw	rw

31:16 RTA_CmpG_8_Conf1

15:0 RTA_CmpG_8_Conf0

53.5.10 APWM G Enable input and state register

Table 779. 0x18 - Enable State In - APWM G Enable input and state register

31 4	3	2 0
RESERVED	InP	InH
0x0	0	0x0
r	rw	rw

31:4 RESERVED

3: Enable previous state (CmpInp)

2:0 Enable state from previous PWM and output PWM.

Select output state (CnfOutInhSel)

000b - TieL (Output OFF)

001b - OutInh

010B - State AB_I

011b – State CF_I

100b – State G_I

101b – ACOMP_I

110b – TieHi

111B – TieHi

53.5.11 APWM G Select state 0 to 3 register

Table 780. 0x1C - SELSTAT0-3 - APWM G Select state 0 to 3 register

31 16	15 12	11 8	7 4	3 0
RESERVED	STATE_CF_II	STATE_CF_I	STATE_A/ AB_II	STATE_A/ AB_I
0x0000	0x0	0x0	0x0	0x0
r	rw	rw	rw	rw

Table 780. 0x1C - SELSTAT0-3 - APWM G Select state 0 to 3 register

31:16	RESERVED
15:12	STATE_CF_II
	0000b - 1011b: STATE_CF
	1100b - 1111b: Reserved (when selected drives low)
11:8	STATE_CF_I
	0000b - 1011b: STATE_CF
	1100b - 1111b: Reserved (when selected drives low)
7:4	STATE_A/AB_II
	0000b - 0011b: STATE_A
	0100b - 0111b: STATE_AB_A
	1000b - 1011b: STATE_AB_B
	1100b - 1111b: Reserved (when selected drives low)
3:0	STATE_A/AB_I
	0000b - 0011b: STATE_A
	0100b - 0111b: STATE_AB_A
	1000b - 1011b: STATE_AB_B
	1100b - 1111b: Reserved (when selected drives low)

53.5.12 APWM G Select state 4 to 7 register

Table 781. 0x20 - SELSTAT4-7 - APWM G Select state 4 to 7 register

31 16	17 13	12 8	7 4	3 0
RESERVED	ACOMP_II/ GPIO_II	ACOMP_I/ GPIO_I	STATE_G_II	STATE_G_I
0x0000	0x0	0x0	0x0	0x0
r	rw	rw	rw	rw

31:18 RESERVED

17:13 ACOMP_II/GPIO_II

0000b - 1011b: GPIO

1100b - 1111b: ACOMP_Synch (0 to 19)

The GPIO connections to each block are described below

APWM G 0: GPIO 1 - 12 (GPIO 1 corresponds to bit 0000b)

APWM G 1: GPIO 13 - 24

APWM G 2: GPIO 25 - 36

APWM G 3: GPIO 49 – 50, 59 - 63, 0 - 4

APWM G* 0 - 3: GPIO 1 - 12

APWM G* 4 - 7: GPIO 13 - 24

APWM G* 8 - 11: GPIO 25 - 36

12:8 ACOMP_I/GPIO_II

0000b - 1011b: GPIO

1100b - 1111b: ACOMP_Synch (0 to 19)

The GPIO connections to each block are described below

APWM G 0: GPIO 1 - 12 (GPIO 1 corresponds to bit 0000b)

APWM G 1: GPIO 13 - 24

APWM G 2: GPIO 25 - 36

APWM G 3: GPIO 49 – 50, 59 - 63, 0 - 4

APWM G* 0 - 3: GPIO 1 - 12

APWM G* 4 - 7: GPIO 13 - 24

APWM G* 8 - 11: GPIO 25 - 36

7:4 STATE_G_II

0000b - 0011b : STATE_G

0100b - 1011b : STATE_G*

1100b - 1111b : Reserved (when selected drives low)

3:0 STATE_G_I

0000b - 0011b : STATE_G

0100b - 1011b : STATE_G*

1100b - 1111b : Reserved (when selected drives low)

53.5.13 APWM G Select output state source register

Table 782. 0x24 - OUTSTAT - APWM G Select output state source register

31	2	1	0
RESERVED			SO
0x0			0
r			rw

31: 2 RESERVED

1: 0 SO: Select output source of State.

1:0 - Select output state

00b - TieL (Output OFF)

01b - State G from Cmp8

10b - State G from OneShotTimer function (for State G* block a flip-flop replaces the one shot timer)

11b - TieH (When outInh MUX is used)

53.5.14 APWM G Enable OneShot Trigger register

Table 783. 0x28 - ENABLE - APWM G Enable OneShot Trigger register

31	1	0
RESERVED		ENOS
0x0		0
r		rw

31:1 RESERVED

0 ENOS: Enable OneShot Trigger

Note: Register not available for G* block

53.5.15 APWM G Preset value for OneShot timer register

Table 784. 0x2C - PRESET - APWM G Preset value for OneShot timer register

31	24	23	1	0
RESERVED		PRESET		
0x0		0		
r		rw		

31: 24 RESERVED

23:0 Preset value for OneShot timer

Note: Register not available for G* block

53.5.16 APWM G Configure OneShot output register

Table 785. 0x30 - CNTCNF0 - APWM G Configure OneShot output register

31	2	1	0
RESERVED			CNFOS
0x0			0
r			rw

31: 2 RESERVED

1: 0 CNFOS: Configure OneShot output.

Note: Register not available for G* block

53.5.17 APWM G input state 0 register

Table 786. 0x80 - STATE0 - APWM G input state 0 register

31 28	27 16	15 12	11 8	7 4	3 0
RESERVED	STATE_CF	RESERVED	STATE_AB_B	STATE_AB_A	STATE_A
0	0	0	0	0	0
r	r	r	r	r	r

31:28 RESERVED

27:0 Input State values

53.5.18 APWM G Input state 1 register

Table 787. 0x84 - STATE1 - APWM G input state 1 register

31 16	15 4	3 0
RESERVED	State_G*	State_G
0	0	0
r	r	r

31:16 RESERVED

15:0 Input state value

16 to 19:

State_G*(3 downto 0) for CF0,

State_G*(7 downto 4) for CF1,

State_G*(11 downto 8) for CF2

20 to 31:

GPIO(12 downto 1) for CF0

GPIO(24 downto 13) for CF1

GPIO(36 downto 25) for CF2

53.5.19 APWM G input state 2 register

Table 788. 0x88 - STATE2 - APWM G input state 2 register

31 12	11 0
ACOMP_SYNC	GPIO
0	0
r	r

31:0 Input State values

GPIO:

GPIO(12 downto 1) for G 0

GPIO(24 downto 13) for G 1

GPIO(36 downto 25) for G 2

GPIO(4 downto 0), (63 downto 59) (50 downto 49) for G 3

Similarly,

GPIO(12 downto 1) for G* 0

GPIO(24 downto 13) for G* 1

GPIO(36 downto 25) for G* 2

53.5.20 APWM G Local register interface information

Table 789. 0xE0 - INFO - APWM G Local register interface information

31 24	23 16	15 8	7 0
IRQ	CFG	STAT	IRQ
0x16	0x9	0x02	0x16
r	r	r	r

31: 24 IRQ: Number of Interrupts (same as 7:0)

23: 16 CFG: Number of Configuration registers

15:8 STAT: Number of Status registers

7:0 IRQ: Number of Interrupts

53.6 Timers and Synchronization

Timers and synchronization blocks are distributed over a few central timers and multiple local timers in all ‘APWM’ blocks, to support both multiple independent operation/functions and/or synchronization operations, see overview figure 121. In the event of synchronized operations, local timers should have identical timer setting, i.e., same synchronization source should be selected and integer multiple of cycle time.

Central timers of the APWM (Timers in top left corner in figure 121) can generate synchronization to internal APWM logic, and to others GR716B devices to synchronize multiple GR716B. Thus, the central timers of the APWM (Timers in top left corner in Figure 121) can be configured to synchronize local timers to external timing.

To achieve high system reliability in critical applications, a System Clock Detector in the synchronization block continuously monitor the system clock for anomalies.

53.6.1 Operation of External Synchronization Control

External synchronization can be configured individually for the two internal Main Timers (TIMER32), see top left in figure 121. In turn, any of these two timers can synchronize all the APWM Timers, which is configured individually in each APWM Timer.

A synchronization event is defined as low-to-high transition on the selected GPIO input, which will reset the Main Timer (TIMER32) to zero. One or more GR716B devices can be configured as synchronized slaves, which receive the synchronization signal from another GR716B device configured as master, or from any other synchronization source on PCB.

Synchronization events shall come with exactly the same period time as configured in the slave Main Timer, or with a period time that is exactly an integer number of the slave period time. Hence, the slave Main Timer is allowed to count its period cycle an integer number of times between two incoming synchronization events.

More than one incoming synchronization event during one slave period will reset the slave Main Timer to zero at each event, which would make the configured slave period time meaningless. Moreover, the resulting slave period time will depend on the incoming events to maintain the desired period time, which comes with the consequence that SEE or other disturbances are propagated directly into the timers without any possibility to filter them. Therefore, it is not recommended to run any slave Main Timer on re-synchronization cycle-by-cycle like that. Instead, it is recommended to run the master Main Timer on the same period time as the slave (or integer period multiples of the slave), and apply masking of synchronization events as described below to achieve SEE/disturbance immunity.

The master synchronization generation in GR716B is based on a tick/trigger signal from any of the two Main Timers. It can be configured to toggle the synchronization output signal at an arbitrary time position in the Main Timer cycle, and will generate a square wave with 50% duty cycle and period time that is two Main Timer periods.

To configure the same phase/time position for the turnaround/reset in slave(s) as in the master Main Timer, the master should be configured according to the following (expressed in number of internal system clock cycles):

MainTimerTick = *TurnaroundMainTimer* - *AnDelay* - *Offset*

where,

MainTimerTick: tick/trigger value to be configured in the master Main Timer

TurnaroundMainTimer: turnaround/reset value configured in the master Main Timer

AnDelay: analog delay from master to slave on PCB, including any LP filtering, etc

Offset = 6 (TBC), fixed internal delay in GR716B master plus slave in total

The GR716B slave detects the synchronization events with resolution of the internal system clock, and generates tick/trigger signals into any of its two Main Timers. When the slave is configured to not have any event masking (Masking config=15 in table 790), all synchronization events are unconditionally passed through to the Main Timer, and synchronization alarm is never generated. The resulting phase/time position of the slave Main Timer, compared to the master Main Timer, is controlled by the master MainTimerTick configuration, where the expression presented above will give same phase/time position in slave and master, with uncertainty of about one internal system clock cycle.

Switching power supplies is an application area that commonly can benefit from synchronized switching control, to run different power converters at exact frequency multiples of each other, since 'beat frequencies' commonly are essential to avoid from EMC point of view in power applications. Furthermore, it may be advisable to not cause timer synchronization updates repeatedly, because the updates would cause small discrete time jumps repeatedly, which may be unfavorable from EMC point of view. Therefore, the synchronization control configuration can be set to only *detect* that valid synchronization events occur inside an expected time window, and the slave Main Timer continues to run without update. Update will only be executed if multiple synchronization events occur outside the window, which indicates that the slave Main Timer has permanently drifted outside its desired time window.

Configuration of slave-window begin and end time positions, wherein the synchronization events are expected, is done according to (expressed in number of internal system clock cycles):

MainTimerTick = *TurnaroundMainTimer* - *WinBeginMarg* - *Offset*

where,

MainTimerTick: tick/trigger value to be configured in the slave Main Timer

TurnaroundMainTimer: turnaround/reset value configured in the slave Main Timer

WinBeginMarg: User-selected window margin *before* the expected synchronization time point (always >1, since the slave and master internal system clocks cannot be regarded synchronous).

Offset = 2 (TBC), fixed internal delay

WinSize = *WinBeginMarg* + *WinEndMarg*

where,

WinSize: Window-size value to be configured in the slave

WinEndMarg: User-selected window margin *after* the expected synchronization time point (always >1).

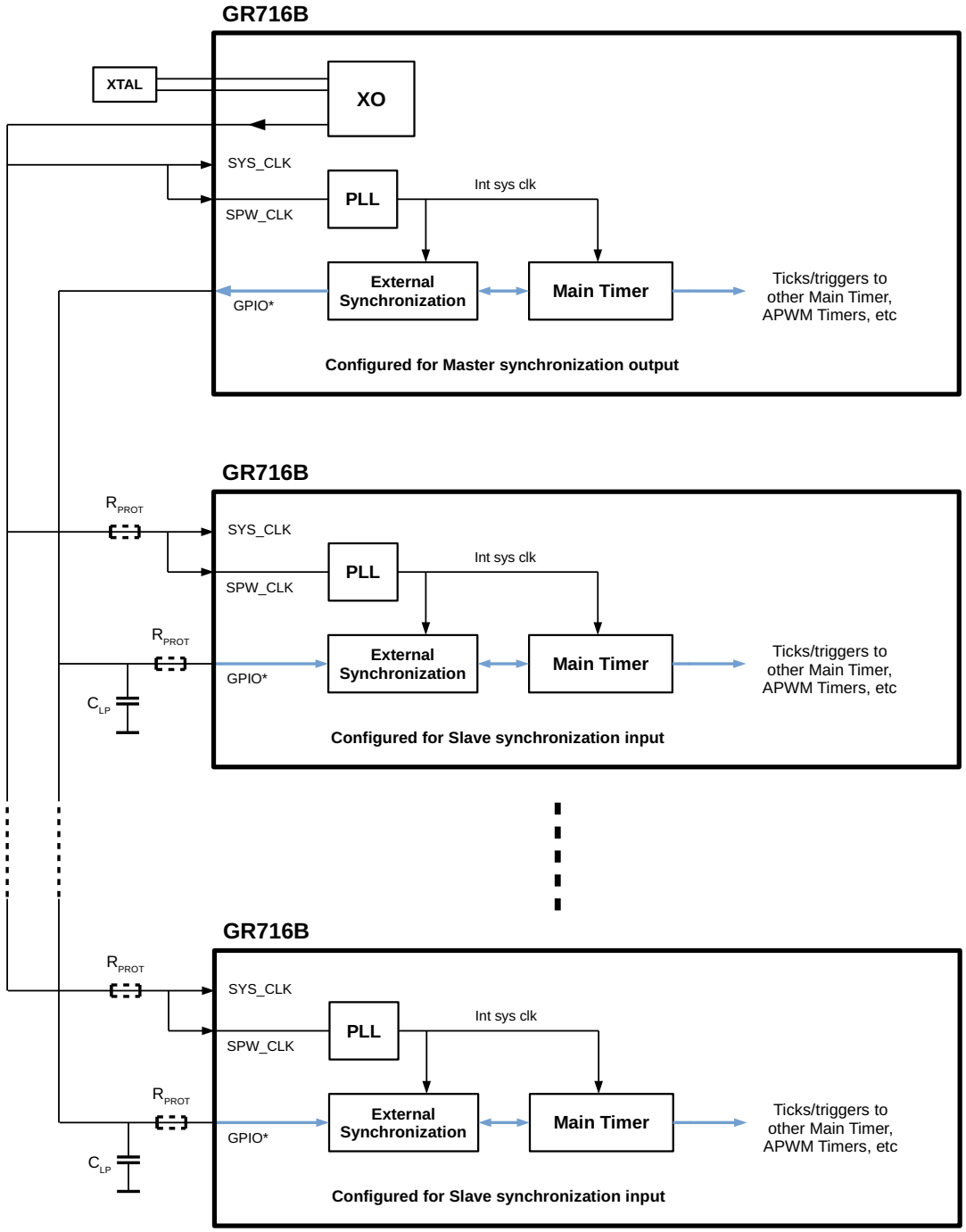
The slave synchronization control can also be configured to suppress false synchronization events from irradiation SEE or other incoming single-event disturbances such as fuse-blow events. If an event happens outside the expected time window it can generate a preventive alarm (and interrupt) before the next synchronization event comes that will execute a Main Timer update. If a false event happens inside the expected time window it can simply be ignored without timer update. Generally, configuration can be set to completely ignore the first (or more) synchronization event(s) outside the window, before generating alarm and executing Main Timer update, see table 790. This may be a commonly useful feature in any single-event disturbed environments.

Table 790. Configuration for masking of a certain number of incoming synchronization events *outside* the expected window. An event *inside* the window resets this count sequence, except for Masking config=15 which does not have any sequence.

	Masked/ignored number of events before:		
Masking config	**Alarm**	**Timer update**	**Comment**
15 (0b1111)*	-	0	Unconditional timer update. Alarm disabled.
7 (0b0111)	0	1	
3 (0b0011)	1	2	
1 (0b0001)	2	3	
0 (0b0000)	3	4	
* Configuration after Reset.			

The event masking configuration could be used to ignore up to three correct/intentional synchronization events outside the window, before generating alarm. Thus, it would be possible to run the master with shorter period time than the slave, using this masking feature to filter out up to three events, provided that the fourth event always comes inside the expected window, which will re-start this slave sequence without alarm or timer update. However, this synchronization method leaves less capability to handle single-event disturbances. Moreover, the slave turnaround/reset time position becomes ambiguous in the sense that it starts at any of four master cycles at system start up. If multiple slaves use this synchronization method, their turnaround/reset time positions relative to each other may thereby be ambiguous. Hence, this synchronization method would in general not be recommended, unless some specific application would require it. Then, special care needs to be taken for the ambiguous slave turnaround/reset time positions and the reduced single-event disturbance immunity.

A typical example of a recommended circuit connection and configuration, with synchronization event masking, is presented in figure 131. It includes analog low-pass filtering at each GPIO synchronization input pin, recommended if there is any risk of repetitive electrical disturbances on the cable/PCB wire.



*** GPIO# 0, 17, 30, 31 or 32**

Figure 131. Connection example from one GR716B master to multiple slaves. Slave configuration: Masking config=3, WinBeginMarg=WinEndMarg=4 giving WinSize=8, i.e., window of about +/-40ns, at f_intsys=100MHz. Disturbance suppression by C_LP, with recommended total capacitance sum in the order of 50 to 100 pF. Each slave with other V_DD_IO net than the master needs 470ohm input protection resistors.

In this example, the first synchronization event outside the expected window is ignored. The second consecutive event outside the window generates alarm. The third consecutive event outside the window executes a synchronization update of the slave Main Timer, to run on the same turnaround/reset time position as the master. Any synchronization event inside the window will reset this sequence without executing synchronization update. In conclusion, this application will never continue to run with the slave Main Timer outside the window longer than three synchronization cycles.

Note that, in normal operation, the slave Main Timer should never end up outside the window due to any frequency generation drifts or transient disturbances, since there is no long-term frequency drifts between the master and slaves due to a common oscillator source in this example. That would only happen if an internal Main Timer error occurs, which will activate this timer's alarm signal which, in turn, should be correctly handled in the alarm matrix and interrupt error-handling routine.

In general, to confirm that a certain configuration gives same turnaround/reset time position in slave(s) as in the master, any APWM Timer and an APWM A, AB or CF block can be configured identically in the slave(s) and master. An oscilloscope can be used to check that the APWM A/AB/CF output switchings for each slave occur at the same time as for the master, with an uncertainty of about one internal system clock cycle.

53.6.2 Operation of System Clock Detector

The internal clock detector continuously checks the internal system clock with independent internal oscillators. The detector should be configured with clock divisor and limits to be checked according to table 791. For switching power applications, in general, the first configuration is recommended, Div_intosc=0, due to the shortest detection time (t_det=1us_max), because PCB components in these applications can be subject to permanent damage very quickly if duty cycle or period time control is compromised.

*Table 791.*Configurations of the System Clock Detector.

Div_intosc config	f_instsys* (MHz)	Limit_min** config	Limit_max** config	t_det, typ (us)	t_det. max (us)
0	32-100	>=3	<=25	0.4	1
1	16-100	>=3	<=50	0.8	2
2	8-100	>=3	<=100	1.6	4
3	2-63	>=3	<=254	6	16
* Normal operation range for f_intsys. Outside this range the detector may generate false clock-fault detections or output-register values.					
** Limit check is turned off for Limit_min=0 and Limit_max=255 respectively. The presented limit values are applicable over full temperature and f_intsys range. If tighter limits would be desired, the application can measure the junction temperature and detector output register and update the limit values continuously.					

The detector should be started by transition from disabled to enabled for internal oscillators. If a single-event fault is detected in an internal oscillator, it should be restarted by first setting it to disabled followed by enabling it, which will not impact running applications. Detection of faults is done by polling overflow in the detector output register.

In case of detection of system clock fault, an alarm is generated to the alarm matrix, where interrupt and GPIO output shutdown can be configured. To clear this alarm, the clock detector alarm latch and alarm matrix latch need to be cleared in that order. But it should be noted that this alarm may imply that on-chip system clocks can be severly faulty, which might require a system restart by power shutdown from external power system control.

53.6.3 External Sync Control Registers

The core is programmed through registers mapped into APB address space.

*Table 792.*External Sync Control registers

APB address offset	Register
0x81006000	External Sync Control Interrupt register
0x81006004	External Sync Control Interrupt Status register
0x81006008	External Sync Control Interrupt Mask register
0x8100600C	External Sync Control Interrupt Edge register
0x81006010	External Sync Control Select input GPIO number (0, 17, 30, 31, 32) (GPIO with schmitt used)
0x81006014	External Sync Control Preset value for 12 bit counter
0x81006018	External Sync Control Number of external sync pulses for resynch
0x8100601C	External Sync Control Select input GPIO number (0, 17, 30, 31, 32) (GPIO with schmitt used)
0x81006020	External Sync Control Preset value for 12 bit counter
0x81006024	External Sync Control Number of external sync pulses for resynch
0x81006080	External Sync Control Shift Register status
0x81006084	External Sync Control Shift Register status
0x810060E0	Local register interface information

53.6.4 External Sync Control Interrupt register

Table 793. 0x00 - ESCINT - External Sync Control Interrupt register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								P	EA 1	EA 0	SO 1	SO 0	TO 1	TO 0	TT
0x0																								0	0	0	1	1	0	0	1
r																								rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*

31: 8 RESERVED

7: Configuration parity error status (P). This parity check covers configuration registers. Becomes '1' when a configuration parity error is detected.

6: ExtReSynchAlarm (EA1) - External Sync Control 1 ExtReSynchAlarm (connected to Alarm block)

5: ExtReSynchAlarm (EA0) - External Sync Control 0 ExtReSynchAlarm (connected to Alarm block)

4: System Tick Output (SO1) - External Sync Control 1 Tick Output (connected to external GPIO)

3: System Tick Output (SO0) - External Sync Control 0 Tick Output (connected to external GPIO)

2: Tick Output (TO1) - External Sync Control 1 Tick Output (connected to Timer32_A and B)

1: Tick Output (TO0) - External Sync Control 0 Tick Output (connected to Timer32_A and B)

0: Timer Tick (TT): Tick output from TIMER32_A tick 7, input to External Sync Control 0 and 1

For test-purposes, bits 7:0 in this register are writable. The written data overrides the normal data source for exactly one system clock cycle. This provides a means for triggering the associated interrupt from software.

NOTE: The fields of this register are updated at every clock cycle, depending on the corresponding signals. These values are not latched and polling routines could easily miss signal transitions. To monitor the status of a signal it is preferable to use the Interrupt Status register.

53.6.5 External Sync Control Interrupt Status register

Table 794. 0x04 - ESCSTAT - External Sync Control Interrupt Status register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								P	EA 1	EA 0	SO 1	SO 0	TO 1	TO 0	TT
0x0																								0	0	0	0	0	0	0	0
r																								wc	wc	wc	wc	wc	wc	wc	wc

31: 7 RESERVED

6: 0 Bit *x* of this register is set to 1 whenever an interrupt has been generated due to bit *x* of ESCINT. Each bit in this stays set to 1 until software clears it by writing 1.

Write 1 to clear.

53.6.6 External Sync Control Interrupt Mask register

Table 795. 0x08 - ESCMSK - Interrupt Mask register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								P	EA 1	EA 0	SO 1	SO 0	TO 1	TO 0	TT
0x0																								0	0	0	0	0	0	0	0
r																								rw	rw	rw	rw	rw	rw	rw	rw

31: 7 RESERVED

6: 0 Bit *x* of this register controls whether the event corresponding to bit *x* of ESCINT can generate an interrupt or not. When bit *x* is 0, that event will not generate interrupts.

53.6.7 External Sync Control Interrupt Edge register

Table 796. 0xC - ESCEDGE - External Sync Control Interrupt Edge register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								P	EA 1	EA 0	SO 1	SO 0	TO 1	TO 0	TT
0x0																								0	0	0	0	0	0	0	0
r																								rw	rw	rw	rw	rw	rw	rw	rw

31: 7 Reserved

6: 0 Bit *x* of this registers controls the polarity of interrupt generation from bit *x* of the ESCINT register.
0 => falling edge: interrupt generated when bit *x* changes from 1 to 0
1 => rising edge: interrupt generated when bit *x* changes from 0 to 1

53.6.8 External Sync Control x Select GPIO input source register

Table 797. 0x10 - SELGPIO - External Sync Control x Select GPIO input source register

31	3	2	0
RESERVED		SEL	
0x0		0x0	
r		rw	

31: 4 RESERVED

3:0 SEL: Select GPIO Input Source

0: GPIO 0

1: GPIO 17

2: GPIO 30

3: GPIO 31

4: GPIO 32

5: 7: reserved

Note: Register description applies for External Sync Control 1as well available at address offset 0x1C.

53.6.9 External Sync Control x Preset value for 12 bit counter register

Table 798. 0x14 - PRESET- External Sync Control Preset value for 12 bit counter register

31	12	11	0
RESERVED		PRESET	
0		0x0	
rw		rw	

31:12 Reserved

11:0 PRESET: Preset value for 12 bit counter

Note: Register description applies for External Sync Control 1as well available at address offset 0x20.

53.6.10 External Sync Control x Number of external sync pulses for resynch register

Table 799. 0x18 - NBRRESYNC - External Sync Control x Number of external sync pulses for resynch register

31	4	3	0
RESERVED		NBR	
0x0		0x0	
r		rw	

31:4 Reserved

3:0 NBR: Number of external sync pulses for resynch

Note: Register description applies for External Sync Control 1as well available at address offset 0x24.

53.6.11 External Sync Control x shift register status

Table 800. 0x80 - ShiftRegSts - External Sync Control x shift register status

31 4	3 0
RESERVED	NBR
0x0	0x0
r	r

31:4 Reserved

3:0 NBR: Number of external sync pulses for resynch

Note: Register description applies for External Sync Control 1as well available at address offset 0x84.

53.6.12 External Sync Control Local register interface information

Table 801. 0xE0 - INFO - External Sync Control Local register interface information

31 24	23 16	15 8	7 0
IRQ	CFG	STAT	IRQ
0x20	0x06	0x02	0x20
r	r	r	r

31: 24 IRQ: Number of Interrupts (same as 7:0)

23: 16 CFG: Number of Configuration registers

15:8 STAT: Number of Status registers

7:0 IRQ: Number of Interrupts

53.6.13 Clock Error Detect Registers

The core is programmed through registers mapped into APB address space.

*Table 802.*Clock Error detect registers

APB address offset	Register
0x8100D000	Interrupt register
0x8100D004	Interrupt status register
0x8100D008	Mask register
0x8100D00C	Interrupt edge register
0x8100D010	Mux Configuration for ClkDiv
0x8100D014	Min value configuration for SysClkDet_A
0x8100D018	Max value configuration for SysClkDet_A
0x8100D01C	Min value configuration for SysClkDet_B
0x8100D020	Max value configuration for SysClkDet_B
0x8100D024	Reserved
0x8100D028	Reserved
0x8100D02C	Input data and mux enable for input data from CPU
0x8100D030	Oscillator Enable
0x8100D080	Counter value from SysClkDet_A
0x8100D084	Counter value from SysClkDet_B
0x8100D088	Valid signal from SysClkDet_A
0x8100D08C	Valid signal from SysClkDet_B
0x8100D090	Error signal from SysClkDet_A
0x8100D094	Error signal from SysClkDet_B
0x8100D098	Clock detect latch SysClk_Ok
0x8100D0E0	Local register interface information
0x8110B000 - 0x8110B0FF [1)]	Clock error detection 1

53.6.14 Clock Error Detect Interrupt register

Table 803. 0x00 - CLKDETINT - Clock Error Detect Interrupt register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																											P	BOK	AOK	OK	R
0x0																											0	0	0	0	0
r																											rw*	rw*	rw*	rw*	r

31: 5 RESERVED

4: Configuration parity error status (P). This parity check covers configuration registers. Becomes '1' when a configuration parity error is detected.

3: SysClk_Ok_B

2: SysClk_Ok_A

1: SysClk_Ok (OK) - System clock functioning properly

0: Reserved

For test-purposes, bits 4:0 in this register are writable. The written data overrides the normal data source for exactly one system clock cycle. This provides a means for triggering the associated interrupt from software.

NOTE: The fields of this register are updated at every clock cycle, depending on the corresponding signals. These values are not latched and polling routines could easily miss signal transitions. To monitor the status of a signal it is preferable to use the Interrupt Status register.

53.6.15 Clock Error Detect Interrupt Status register

Table 804. 0x04 - CLKDETISTAT - APWM G Interrupt Status register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																											P	BOK	AOK	OK	R
0x0																											0	0	0	0	0
r																											wc	wc	wc	wc	r

31: 5 RESERVED

4: 1 Bit *x* of this register is set to 1 whenever an interrupt has been generated due to bit *x* of GINT. Each bit in this stays set to 1 until software clears it by writing 1.

0: Reserved

Write 1 to clear.

53.6.16 Clock Error Detect Interrupt Mask register

Table 805. 0x08 - CLKDETMSK - Interrupt Mask register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																											P	BOK	AOK	OK	R
0x0																											0	0	0	0	0
r																											rw	rw	rw	rw	r

31: 5 RESERVED

4: 1 Bit *x* of this register controls whether the event corresponding to bit *x* of GINT can generate an interrupt or not. When bit *x* is 0, that event will not generate interrupts.

0: Reserved

53.6.17 Clock Error Detect Interrupt Edge register

Table 806. 0xC - CLKDETEDGE - Clock Error Detect Interrupt Edge register

31 … 5	4	3	2	1	0
RESERVED	P	BOK	AOK	OK	R
0x0	1	1	1	1	0
r	rw	rw	rw	rw	r

31: 5 Reserved

4: 1 Bit *x* of this registers controls the polarity of interrupt generation from bit *x* of the GINT register.
0 => falling edge: interrupt generated when bit *x* changes from 1 to 0
1 => rising edge: interrupt generated when bit *x* changes from 0 to 1

0: Reserved

53.6.18 Clock Error Detect Mux Configuration for ClkDiv register

Table 807. 0x10 - MUXCLKDIV - Clock Error Detect Mux Configuration for ClkDiv register

31 … 2	1 … 0
RESERVED	CD
0x0	0
r	rw

31:12 RESERVED

11:0 ClkDiv (CD) configuration. Ring Oscillator Clock divided by the configured value

0: No division

1: Divide by 2

2: Divide by 4

3: Divide by 16

53.6.19 Clock Error Detect Min value configuration for SysClkDet_A register

Table 808. 0x14 - MIN_SYSClkDet_A - Clock Error Detect Min value configuration for SysClkDet_A register

31 … 8	7 … 0
RESERVED	MINA
0x0	0
r	rw

31:8 RESERVED

7:0 MINA: Min value configuration for SysClkDet_A

53.6.20 Clock Error Detect Max value configuration for SysClkDet_A register

Table 809. 0x18 - MAX_SYSClkDet_A - Clock Error Detect Max value configuration for SysClkDet_A register

31 8	7 0
RESERVED	MAXA
0x0	0
r	rw

31:8 RESERVED

7:0 MAXA: Max value configuration for SysClkDet_A

53.6.21 Clock Error Detect Min value configuration for SysClkDet_B register

Table 810. 0x1C - MIN_SYSClkDet_B - Clock Error Detect Min value configuration for SysClkDet_B register

31 8	7 0
RESERVED	MINB
0x0	0
r	rw

31:8 RESERVED

7:0 MINB: Min value configuration for SysClkDet_B

53.6.22 Clock Error Detect Max value configuration for SysClkDet_B register

Table 811. 0x20- MAX_SYSClkDet_B - Clock Error Detect Max value configuration for SysClkDet_B register

31 8	7 0
RESERVED	MAXB
0x0	0
r	rw

31:8 RESERVED

7:0 MAXB: Min value configuration for SysClkDet_B

53.6.23 Clock Error Detect Input data and mux enable for input data from CPU register

Table 812. 0x2C - INDATA_MUX - Clock Error Detect Input data and mux enable for input data from CPU register

31 24	23 2	1	0
RESERVED		MX	IB
0x0		0	0
r		rw	rw

31: 24 RESERVED

1: MX: Mux enable for input data

0: ID: Input data

53.6.24 Clock Error Detect Oscillator Enable register

Table 813. 0x30 - ENOS- Clock Error Detect Oscillator Enable register

31 1	0
RESERVED	EN
0x0	0
r	rw

31: 1 RESERVED

0: EN: Oscillator Enable.

53.6.25 Clock Error Detect Counter value from SysClkDet_A register

Table 814. 0x80 - STATE0 - Clock Error Detect Counter value from SysClkDet_A register

31 9	8 0
RESERVED	COUNT_A
0	0
r	r

31:9 RESERVED

8:0 Counter value from SysClkDet_A

53.6.26 Clock Error Detect Counter value from SysClkDet_B register

Table 815. 0x84 - STATE1 - Clock Error Detect Counter value from SysClkDet_B register

31 9	8 0
RESERVED	COUNT_B
0	0
r	r

31:9 RESERVED

8:0 Counter value from SysClkDet_B

53.6.27 Clock Error Detect valid signal from SysClkDet_A register

Table 816. 0x88 - STATE2 - Clock Error Detect valid signal from SysClkDet_A register

31	1	0
RESERVED		VA
0		0
r		r

31:1 RESERVED

0: VA: Valid signal from SysClkDet_A

53.6.28 Clock Error Detect valid signal from SysClkDet_B register

Table 817. 0x8C - STATE3 - Clock Error Detect valid signal from SysClkDet_B register

31	1	0
RESERVED		VB
0		0
r		r

31:1 RESERVED

0: VB: Valid signal from SysClkDet_B

53.6.29 Clock Error Detect error signal from SysClkDet_A register

Table 818. 0x90 - STATE4 - Clock Error Detect error signal from SysClkDet_A register

31	1	0
RESERVED		EA
0		0
r		r

31:1 RESERVED

0: EA: Valid signal from SysClkDet_A

53.6.30 Clock Error Detect error signal from SysClkDet_B register

Table 819. 0x94 - STATE5 - Clock Error Detect error signal from SysClkDet_B register

31	1	0
RESERVED		EB
0		0
r		r

31:1 RESERVED

0: EB: Valid signal from SysClkDet_B

53.6.31 Clock Error Detect latch SysClk_Ok register

Table 820. 0x98 - STATE6 - Clock Error Detect latch SysClk_Ok register

31	2	1	0
RESERVED	OKB	OKA	OK
0	0	0	0
r	r	r	r

31:1 RESERVED

2: SysClk_Ok_B

Table 820. 0x98 - STATE6 - Clock Error Detect latch SysClk_Ok register

1: SysClk_Ok_A

0: SysClk_Ok

53.6.32 Clock Error Detect Local register interface information

Table 821. 0xE0 - INFO - Clock Error Detect Local register interface information

31 24	23 16	15 8	7 0
IRQ	CFG	STAT	IRQ
0x4	0x9	0x07	0x4
r	r	r	r

31: 24 IRQ: Number of Interrupts (same as 7:0)

23: 16 CFG: Number of Configuration registers

15:8 STAT: Number of Status registers

7:0 IRQ: Number of Interrupts

53.7 Alarm Matrix and Shutdown

The APWM unit provides protection alarm signals, going to the central protection configuration matrix on the chip. In this matrix, the user configures which protection shutdown signals that shall be activated, based on all incoming alarm signals. Alarm input signals are from ACOMP, FIR limit trig, GPIO, and internal health-check monitors. Shutdown signals out from the matrix are, e.g., the protection shutdown signals that go to each APWM block and do protection shutdown of the PWM output signal, e.g., connected to power switches in DC/DC applications. Also, a number of shutdown signals are configurable to GPIO outputs.

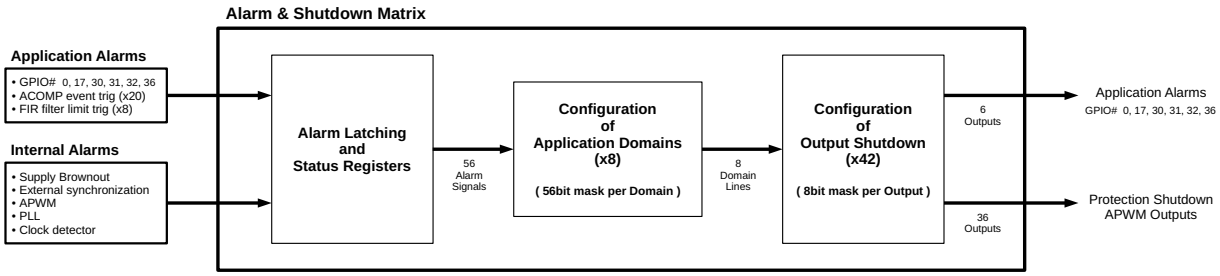


Figure 132. Block diagram of the Alarm and Shutdown Matrix.

*Table 822.*Alarm Protection lines

Incoming alarm signal	Source
Alarm input (0 to 5)	GPIO 0, 17, 30, 31, 32 and 36
Alarm input (6 to 25)	ACOMP Event Trig x20, each input is also ORed with a common ACOMP configuration parity error signal.
Alarm input (26 to 33)	FIR Filter Limit Trig x8, each input is also ORed with FIR configuration parity error signal from its respective configuration registers.
Alarm input (34)	External synchronisation 0, also ORed with configuration parity error signal from Ext Sync 0 configuration registers.
Alarm input (35)	External synchronisation 1, also ORed with configuration parity error signal from Ext Sync 1 configuration registers.
Alarm input (36)	Timer32_A, also ORed with configuration parity error signal from Timer32_A configuration registers.
Alarm input (37)	Timer32_B,also ORed with configuration parity error signal from Timer32_A configuration registers.
Alarm input (38)	Timer27_0, also ORed with configuration parity error signal from Timer27_0 configuration registers.
Alarm input (39)	Timer27_1, also ORed with configuration parity error signal from Timer27_1 configuration registers.
Alarm input (40)	APWM_TIMER (APWM_A), also ORed with configuration parity error signal from APWM_A 0, 1, 2 and 3 configuration registers.
Alarm input (41)	APWM_TIMER0 (APWM_AB0), also ORed with configuration parity error signal from APWM_AB 0 configuration registers.
Alarm input (42)	APWM_TIMER1 (APWM_AB1), also ORed with configuration parity error signal from APWM_AB 1 configuration registers.
Alarm input (43)	APWM_TIMER2 (APWM_AB2), also ORed with configuration parity error signal from APWM_AB 2 configuration registers.
Alarm input (44)	APWM_TIMER3 (APWM_AB3), also ORed with configuration parity error signal from APWM_AB 3 configuration registers.
Alarm input (45)	APWM_TIMER0 (APWM_CF0), also ORed with configuration parity error signals from APWM_CF0 0, 1, 2 and 3 configuration registers.
Alarm input (46)	APWM_TIMER1 (APWM_CF1), also ORed with configuration parity error signals from APWM_CF1 0, 1, 2 and 3 configuration registers.
Alarm input (47)	APWM_TIMER2 (APWM_CF2), also ORed with configuration parity error signals from APWM_CF2 0, 1, 2 and 3 configuration registers.
Alarm input (48)	One shot Timer Alarm (APWM_G 0), also ORed with configuration parity error signal from APWM_G 0 configuration registers.
Alarm input (49)	One shot Timer Alarm (APWM_G 1), also ORed with configuration parity error signal from APWM_G 1 configuration registers.
Alarm input (50)	One shot Timer Alarm (APWM_G 2), also ORed with configuration parity error signal from APWM_G 2 configuration registers.
Alarm input (51)	One shot Timer Alarm (APWM_G 3), also ORed with configuration parity error signal from APWM_G 3 configuration registers.
Alarm input (52)	Configuration registers parity error signals from APWM_G* 0 to 11 ORed together.
Alarm input (53)	Configuration parity error signal from PLL configuration registers.
Alarm input (54)	Clock detector Alarm, also ORed with configuration parity error signals from Clock detector 0, Clock detector 1. This alarm is provided with asynchronous bypass of SR_Alarm.
Alarm input (55)	Brownout Alarm, also ORed with configuration parity error signal from Brownout configuration registers.
Alarm input (56)	Configuration registers parity error signal from IO Matrix configuration registers.
Alarm input (57)	Configuration registers parity error signal from Alarm Protection registers (Alarm Matrix) configuration registers.
Note: Bit-masking plus OR:ing for Application Alarm Lines (x8) are configured through 53.7.1 Alarm Protection APP Registers	

Table 823. Alarm Outputs and Protection Shutdown Outputs

Configuration Number	Alarm Outputs and Protection Shutdown Outputs
0 to 5	GPIO_Alarm_Output 0, 17, 30, 31, 32 and 36
6 to 9	ProtShdn_APWM_A 0, 1, 2 and 3
10 to 13	ProtShdn_APWM_AB 0, 1, 2 and 3
14 to 17	ProtShdn_APWM_CF0 0, 1, 2 and 3
18 to 21	ProtShdn_APWM_CF1 0, 1, 2 and 3
22 to 25	ProtShdn_APWM_CF2 0, 1, 2 and 3
26 to 29	ProtShdn_APWM_G 0, 1, 2 and 3
30 to 41	ProtShdn_APWM_G* 0 to 11
Note: Bit-masking plus OR:ing for Protection Shutdown Outputs (x42) are configured through 53.7.10 Alarm Protection Shutdown Registers	

53.7.1 Alarm Protection APP Registers

The core is programmed through registers mapped into APB address space.

Table 824. Alarm protection A registers

APB address offset	Register
0x81003000	Alarm protection APP Interrupt register
0x81003004	Alarm protection APP Interrupt status register
0x81003008	Alarm protection APP Interrupt Mask register
0x8100300C	Alarm protection APP Interrupt edge register
	Config for Bit-masking plus OR:ing for Application Alarm Lines (32 * 8) + (24 *8)
0x81003010	Alarm protection APP_ 0_0_REG
0x81003014	Alarm protection APP_ 0_1_REG
0x81003018	Alarm protection APP_ 1_0_REG
0x8100301C	Alarm protection APP_ 1_1_REG
0x81003020	Alarm protection APP_ 2_0_REG
0x81003024	Alarm protection APP_ 2_1_REG
0x81003028	Alarm protection APP_ 3_0_REG
0x8100302C	Alarm protection APP_ 3_1_REG
0x81003030	Alarm protection APP_ 4_0_REG
0x81003034	Alarm protection APP_ 4_1_REG
0x81003038	Alarm protection APP_ 5_0_REG
0x8100303C	Alarm protection APP_ 5_1_REG
0x81003040	Alarm protection APP_ 6_0_REG
0x81003044	Alarm protection APP_ 6_1_REG
0x81003048	Alarm protection APP_ 7_0_REG
0x8100304C	Alarm protection APP_ 7_1_REG
0x81003080	Alarm protection APP status
0x810030E0	Alarm protection APP Local register interface information

53.7.2 Alarm protection APP Interrupt register

Table 825. 0x00 - APROTAINT - Alarm protection APP Interrupt register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P	AO																														
0	0																														
rw*	rw*																														

31 Configuration parity error status (P). This parity check covers configuration registers. Becomes '1' when a configuration parity error is detected.

30: 0 AO: Alarm Output(30 downto 0)

For test-purposes, bits 31:0 in this register are writable. The written data overrides the normal data source for exactly one system clock cycle. This provides a means for triggering the associated interrupt from software.

NOTE: The fields of this register are updated at every clock cycle, depending on the corresponding signals. These values are not latched and polling routines could easily miss signal transitions. To monitor the status of a signal it is preferable to use the Interrupt Status register.

53.7.3 Alarm protection APP Interrupt Status register

Table 826. 0x04 - APROTAISTAT - Alarm protection APP Interrupt Status register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P	AO																														
0	0																														
wc	wc																														

31: 0 Bit *x* of this register is set to 1 whenever an interrupt has been generated due to bit *x* of APROTAINT. Each bit in this stays set to 1 until software clears it by writing 1.

Write 1 to clear.

53.7.4 Alarm protection APP Interrupt Mask register

Table 827. 0x08 - APROTAMSK - Alarm protection APP Interrupt Mask register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P	AO																														
0	0																														
rw	rw																														

31: 0 Bit *x* of this register controls whether the event corresponding to bit *x* of APROTAINT can generate an interrupt or not. When bit *x* is 0, that event will not generate interrupts.

53.7.5 Alarm protection APP Interrupt Edge register

Table 828. 0xC - APROTAEDGE - Alarm protection APP Interrupt Edge register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P	AO																														
1	0x7FFFFFFF																														
rw	rw																														

31: 0 Bit *x* of this registers controls the polarity of interrupt generation from bit *x* of the APROTAINT register.
0 => falling edge: interrupt generated when bit *x* changes from 1 to 0
1 => rising edge: interrupt generated when bit *x* changes from 0 to 1

53.7.6 Alarm protection APP_ 0_0 register

Table 829. 0x10 - APP 00 - Alarm protection APP_ 0_0 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CFG_APP_0_0																															
0																															
rw																															

31:0 CFG_APP_0_0: Config for Bit-masking plus OR:ing for Application Alarm Line 0 (31 downto 0)

Note: Similar set of bit mask available for the application alarm lines 1 to 7, corresponding registers are

Alarm protection APP_ 1_0

Alarm protection APP_ 2_0

Alarm protection APP_ 3_0

Alarm protection APP_ 4_0

Alarm protection APP_ 5_0

Alarm protection APP_ 6_0

Alarm protection APP_ 7_0

53.7.7 Alarm protection APP_ 0_1 register

Table 830. 0x14 - APP 01 - Alarm protection APP_ 0_1 register

31	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED		CFG_APP_0_1																									
0x0		0																									
r		rw																									

25:0 CFG_APP_0_1: Config for Bit-masking plus OR:ing for Application Alarm Lines 0 (57 downto 32)

Note: Similar set of bit mask available for the application alarm lines 1 to 7, corresponding registers are

Alarm protection APP_ 1_1

Alarm protection APP_ 2_1

Alarm protection APP_ 3_1

Alarm protection APP_ 4_1

Alarm protection APP_ 5_1

Alarm protection APP_ 6_1

Alarm protection APP_ 7_1

53.7.8 Alarm protection APP status register

Table 831. 0x80 - STATE0 - Alarm protection APP status register

31	0
AO	
0	
r	

31: 0 AO: Alarm Output(31 downto 0)

53.7.9 Alarm protection APP local register interface information

Table 832. 0xE0 - INFO - Alarm protection APP local register interface information

31 24	23 16	15 8	7 0
IRQ	CFG	STAT	IRQ
0x20	0x10	0x1	0x20
r	r	r	r

31: 24 IRQ: Number of Interrupts (same as 7:0)

23: 16 CFG: Number of Configuration registers

15:8 STAT: Number of Status registers

7:0 IRQ: Number of Interrupts

53.7.10 Alarm Protection Shutdown Registers

The core is programmed through registers mapped into APB address space.

*Table 833.*Alarm protection B registers

APB address offset	Register
0x81004000	Alarm protection shutdown Interrupt register
0x81004004	Alarm protection shutdown Interrupt Status register
0x81004008	Alarm protection shutdown Mask register
0x8100400C	Alarm protection shutdown Interrupt level register
	Config for Bit-masking plus OR:ing for GPIO/AB/A/CF/G protection shut-down (42*8)
0x81004010	Alarm protection shutdown 0_REG
0x81004014	Alarm protection shutdown 1_REG
0x81004018	Alarm protection shutdown 2_REG
0x8100401C	Alarm protection shutdown 3_REG
0x81004020	Alarm protection shutdown 4_REG
0x81004024	Alarm protection shutdown 5_REG
0x81004028	Alarm protection shutdown 6_REG
0x8100402C	Alarm protection shutdown 7_REG
0x81004030	Alarm protection shutdown 8_REG
0x81004034	Alarm protection shutdown 9_REG
0x81004038	Alarm protection shutdown 10_REG
0x81004080	Alarm protection shutdown status
0x810040E0	Alarm protection shutdown local register interface information

53.7.11 Alarm protection shutdown interrupt register

Table 834. 0x00 - APROTSINT - Alarm protection shutdown interrupt register

31	30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
P	RESERVED	AO
0	0	0
rw*	r	rw*

31 Configuration parity error status (P). This parity check covers configuration registers. Becomes '1' when a configuration parity error is detected.

30:26 Reserved

25: 0 AO: Alarm Output(57 downto 32)

For test-purposes, bits 25:0 and 31 in this register are writable. The written data overrides the normal data source for exactly one system clock cycle. This provides a means for triggering the associated interrupt from software.

NOTE: The fields of this register are updated at every clock cycle, depending on the corresponding signals. These values are not latched and polling routines could easily miss signal transitions. To monitor the status of a signal it is preferable to use the Interrupt Status register.

53.7.12 Alarm protection shutdown interrupt status register

Table 835. 0x04 - APROTSISTAT - Alarm protection shutdown interrupt status register

31	30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
P	RESERVED	AO
0	0	0
wc	r	wc

31: 0 Bit *x* of this register is set to 1 whenever an interrupt has been generated due to bit *x* of APROTSINT. Each bit in this stays set to 1 until software clears it by writing 1.

Write 1 to clear.

53.7.13 Alarm protection shutdown interrupt mask register

Table 836. 0x08 - APROTSMSK - Alarm protection shutdown Interrupt Mask register

31	30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
P	RESERVED	AO
0	0	0
rw	r	rw

31: 0 Bit *x* of this register controls whether the event corresponding to bit *x* of APROTSINT can generate an interrupt or not. When bit *x* is 0, that event will not generate interrupts.

53.7.14 Alarm protection shutdown interrupt edge register

Table 837. 0xC - APROTSEDGE - Alarm protection shutdown interrupt edge register

31	30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
P	RESERVED	AO
1	0	0x3FFFFFF
rw	r	rw

Table 837. 0xC - APROTSEDGE - Alarm protection shutdown interrupt edge register

31: 0 Bit *x* of this registers controls the polarity of interrupt generation from bit *x* of the APROTAINT register.
0 => falling edge: interrupt generated when bit *x* changes from 1 to 0
1 => rising edge: interrupt generated when bit *x* changes from 0 to 1

53.7.15 Alarm protection shutdown 0 register

Table 838. 0x10 - APS0 - Alarm protection shutdown 0 register

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
GPIO_31	GPIO_30	GPIO_17	GPIO_0
0x00	0x00	0x00	0x00
rw	rw	rw	rw

31:24 GPIO_31: Config for Bit-masking plus OR:ing for Protection Shutdown Output(3)

23:16 GPIO_30: Config for Bit-masking plus OR:ing for Protection Shutdown Output(2)

15:8 GPIO_17: Config for Bit-masking plus OR:ing for Protection Shutdown Output(1)

7:0 GPIO_0: Config for Bit-masking plus OR:ing for Protection Shutdown Output(0)

Note: Similar set of bit mask available for the Protection Shutdown Output 4 to 41,

Corresponding registers are

Reg name	Offset	Register split
Alarm protection shutdown 1	0x14	ProtShdn_APWM_A 1, 0, GPIO_36, GPIO_32
Alarm protection shutdown 2	0x18	ProtShdn_APWM_AB 1, 0, ProtShdn_APWM_A 3, 2
Alarm protection shutdown 3	0x1C	ProtShdn_APWM_CF0 1, 0, ProtShdn_APWM_AB 3, 2
Alarm protection shutdown 4	0x20	ProtShdn_APWM_CF1 1, 0, ProtShdn_APWM_CF0 3, 2
Alarm protection shutdown 5	0x24	ProtShdn_APWM_CF2 1, 0, ProtShdn_APWM_CF1 3, 2
Alarm protection shutdown 6	0x28	ProtShdn_APWM_G 1, 0, ProtShdn_APWM_CF2 3, 2
Alarm protection shutdown 7	0x2C	ProtShdn_APWM_G* 1, 0, ProtShdn_APWM_G 3, 2
Alarm protection shutdown 8	0x30	ProtShdn_APWM_G* 5, 4, 3, 2
Alarm protection shutdown 9	0x34	ProtShdn_APWM_G* 9, 8, 7, 6
Alarm protection shutdown 10	0x38	ProtShdn_APWM_G* 11, 10

53.7.16 Alarm protection shutdown status register

Table 839. 0x80 - STATE0 - Alarm protection shutdown status register

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
RESERVED	AO
0	0
r	r

31:26 Reserved

25: 0 AO: Alarm Output(57 downto 32)

53.7.17 Alarm protection shutdown local register interface information

Table 840. 0xE0 - INFO - Alarm protection shutdown local register interface information

31 24	23 16	15 8	7 0
IRQ	CFG	STAT	IRQ
0x20	0x10	0x1	0x20
r	r	r	r

31: 24 IRQ: Number of Interrupts (same as 7:0)

23: 16 CFG: Number of Configuration registers

15:8 STAT: Number of Status registers

7:0 IRQ: Number of Interrupts

53.7.18 Alarm protection set clear registers

The core is programmed through registers mapped into APB address space.

Table 841. Alarm protection set clear registers

APB address offset	Register
0x81005000	Alarm protection set clear Status register
0x81005004	Alarm protection set clear Interrupt register
0x81005008	Alarm protection set clear Mask register
0x8100500C	Alarm protection set clear Interrupt level register
0x81005010	Alarm protection set alarm 0
0x81005014	Alarm protection set alarm 1
0x81005018	Alarm protection clear alarm 0
0x8100501C	Alarm protection clear alarm 1
0x81005020	Alarm protection SR write arm
0x81005080	Alarm protection set clear status
0x810050E0	Alarm protection set clear local register interface information

53.7.19 Alarm protection set clear interrupt register

Table 842. 0x00 - APROTSCINT - Alarm protection set clear interrupt register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P	RESERVED																														AO
0	0																														0
rw*	r																														rw*

31 Configuration parity error status (P). This parity check covers configuration registers. Becomes '1' when a configuration parity error is detected.

30:1 Reserved

0 AO: Alarm Output(31)

For test-purposes, bits 0 and 31 in this register are writable. The written data overrides the normal data source for exactly one system clock cycle. This provides a means for triggering the associated interrupt from software.

NOTE: The fields of this register are updated at every clock cycle, depending on the corresponding signals. These values are not latched and polling routines could easily miss signal transitions. To monitor the status of a signal it is preferable to use the Interrupt Status register.

53.7.20 Alarm protection set clear interrupt status register

Table 843. 0x04 - APROTSCISTAT - Alarm protection set clear interrupt status register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P	RESERVED																														AO
0	0																														0
wc	r																														wc

Bit *31 or 0* of this register is set to 1 whenever an interrupt has been generated due to bit *31 or 0* of APROTSCINT. Each bit in this stays set to 1 until software clears it by writing 1.

Write 1 to clear.

53.7.21 Alarm protection set clear interrupt mask register

Table 844. 0x08 - APROTSCMSK - Alarm protection set clear Interrupt Mask register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P	RESERVED																														AO
0	0																														0
rw	r																														rw

Bit *31 or 0* of this register controls whether the event corresponding to bit *31 or 0* of APROTSCINT can generate an interrupt or not. When bit *31 or 0* is 0, that event will not generate interrupts.

53.7.22 Alarm protection set clear interrupt edge register

Table 845. 0xC - APROTSCEDGE - Alarm protection set clear interrupt edge register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P	RESERVED																														AO
1	0																														1
rw	r																														rw

Bit *31 or 0* of this registers controls the polarity of interrupt generation from bit *31 or 0* of the APROTAINT register.
0 => falling edge: interrupt generated when bit *31 or 0* changes from 1 to 0
1 => rising edge: interrupt generated when bit *31 or 0* changes from 0 to 1

53.7.23 Alarm protection set alarm 0 register

Table 846. 0x10 - SETALR0 - Alarm protection set alarm 0 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SET_0																															
0																															
rw																															

31:0 SET_0: Set Alarm(31 downto 0)

53.7.24 Alarm protection set alarm 1 register

Table 847. 0x14 - SETALR1 - Alarm protection set alarm 1 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED						SET_1																									
0x0						0																									
r						rw																									

25:0 SET_1: Set Alarm(57 downto 32)

53.7.25 Alarm protection clear alarm 0 register

Table 848. 0x18 - CLRALR0 - Alarm protection clear alarm 0 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLR_0																															
0																															
rw																															

31:0 CLR_0: Clear Alarm(31 downto 0)

53.7.26 Alarm protection clear alarm 1 register

Table 849. 0x1C - CLRALR1 - Alarm protection clear alarm 1 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED						CLR_1																									
0x0						0																									
r						rw																									

25:0 CLR_1: Clear Alarm(57 downto 32)

53.7.27 Alarm protection SR write arm register

Table 850. 0x20 - SRARM - Alarm protection SR write arm register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																															ARM
0x0																															0
r																															rw

Table 850. 0x20 - SRARM - Alarm protection SR write arm register

0	ARM: SR write arm

53.7.28 Alarm protection set clear status register

Table 851. 0x80 - STATE0 - Alarm protection set clear status register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																															AO
0																															0
r																															r

31:1	Reserved
0	AO: Alarm Output(31)

53.7.29 Alarm protection set clear local register interface information

Table 852. 0xE0 - INFO - Alarm protection set clear local register interface information

31	24	23	16	15	8	7	0
IRQ		CFG		STAT		IRQ	
0x20		0x5		0x1		0x20	
r		r		r		r	

31: 24	IRQ: Number of Interrupts (same as 7:0)
23: 16	CFG: Number of Configuration registers
15:8	STAT: Number of Status registers
7:0	IRQ: Number of Interrupts

53.8 Registers

The core is programmed through registers mapped into APB address space.

*Table 853.*APWM registers

Function	APB address offset	Register
TIMER32	0x81000000 - 0x810000FF	APWM system timer and synchronization 0
TIMER32	0x81001000 - 0x810010FF	APWM system timer and synchronization 1
SYNCREG	0x81002000 - 0x810020FF	APWM Register synchronization
PROTSHDN	0x81003000 - 0x810030FF	Protection shutdown 0
PROTSHDN	0x81004000 - 0x810040FF	Protection shutdown 1
PROTSHDN	0x81005000 - 0x810050FF	Protection shutdown 2
SYNC	0x81006000 - 0x810060FF	External synchronization
CLKDET	0x8100D000 - 0x8100D0FF	Clock error detection 0
TIMER27	0x8100E000 - 0x8100E0FF	APWM function timer and synchronization 0
TIMER27	0x8100F000 - 0x8100F0FF	APWM function timer and synchronization 1
APWM_TIMER	0x81100000 - 0x811000FF	APWM A Timer and synchronization
APWM_A	0x81101000 - 0x811010FF	APWM A 0
APWM_A	0x81102000 - 0x811020FF	APWM A 1
APWM_A	0x81103000 - 0x811030FF	APWM A 2
APWM_A	0x81104000 - 0x811040FF	APWM A 3
CLKDET	0x8110B000 - 0x8110B0FF	Clock error detection 1
TIMESTAMP	0x8110C000 - 0x8110C0FF	APWM Timestamp
APWM_TIMER	0x81200000 - 0x812000FF	APWM AB0 Timer and synchronization
APWM_AB	0x81201000 - 0x812010FF	APWM AB0
APWM_TIMER	0x81202000 - 0x812020FF	APWM AB1 Timer and synchronization
APWM_AB	0x81203000 - 0x812030FF	APWM AB1
APWM_TIMER	0x81204000 - 0x812040FF	APWM AB2 Timer and synchronization
APWM_AB	0x81205000 - 0x812050FF	APWM AB2
APWM_TIMER	0x81206000 - 0x812060FF	APWM AB3 Timer and synchronization
APWM_AB	0x81207000 - 0x812070FF	APWM AB3
REGSYNC_CF	**0x81300000 - 0x813000FF**	**APWM CF0 Timer and synchronization**
APWM_CF	0x81301000 - 0x813010FF	APWM CF0 0
APWM_CF	0x81302000 - 0x813020FF	APWM CF0 1
APWM_CF	0x81303000 - 0x813030FF	APWM CF0 2
APWM_CF	0x81304000 - 0x813040FF	APWM CF0 3
REGSYNC_CF	0x81305000 - 0x813050FF	APWM CF1 Timer and synchronization
APWM_CF	0x81306000 - 0x813060FF	APWM CF1 0
APWM_CF	0x81307000 - 0x813070FF	APWM CF1 1
APWM_CF	0x81308000 - 0x813080FF	APWM CF1 2

Table 853. APWM registers

Function	APB address offset	Register
APWM_CF	0x81309000 - 0x813090FF	APWM CF1 3
REGSYNC_CF	0x81400000 - 0x814000FF	APWM CF2 Timer and synchronization
APWM_CF	0x81401000 - 0x814010FF	APWM CF2 0
APWM_CF	0x81402000 - 0x814020FF	APWM CF2 1
APWM_CF	0x81403000 - 0x814030FF	APWM CF2 2
APWM_CF	0x81404000 - 0x814040FF	APWM CF2 3
APWM_G	0x81500000 - 0x815000FF	APWMG 0
APWM_G	0x81501000 - 0x815010FF	APWMG 1
APWM_G	0x81502000 - 0x815020FF	APWMG 2
APWM_G	0x81503000 - 0x815030FF	APWMG 3
APWM_G	0x81504000 - 0x815040FF	APWMG 4
APWM_G	0x81505000 - 0x815050FF	APWMG 5
APWM_G	0x81506000 - 0x815060FF	APWMG 6
APWM_G	0x81507000 - 0x815070FF	APWMG 7
APWM_G	0x81508000 - 0x815080FF	APWMG 8
APWM_G	0x81509000 - 0x815090FF	APWMG 9
APWM_G	0x8150A000 - 0x8150A0FF	APWMG 10
APWM_G	0x8150B000 - 0x8150B0FF	APWMG 11
APWM_G	0x8150C000 - 0x8150C0FF	APWMG 12
APWM_G	0x8150D000 - 0x8150D0FF	APWMG 13
APWM_G	0x8150E000 - 0x8150E0FF	APWMG 14
APWM_G	0x8150F000 - 0x8150F0FF	APWMG 15

53.9 System Timers

53.9.1 Registers

The core is programmed through registers mapped into APB address space.

*Table 854.*System Timers A and B registers

APB address offset	Register
0x81000000	TIMER 32 A Status register
0x81000004	TIMER 32 A Interrupt register
0x81000008	TIMER 32 A Mask register
0x8100000C	TIMER 32 A Interrupt level register
0x81000010	TIMER 32 A Select synchronization source
0x81000014	TIMER 32 A System period
0x81000018	TIMER 32 A Timer Tick 0
0x8100001C	TIMER 32 A Timer Tick 1
0x81000020	TIMER 32 A Timer Tick 2
0x81000024	TIMER 32 A Timer Tick 3
0x81000028	TIMER 32 A Timer Tick 4
0x8100002c	TIMER 32 A Timer Tick 5
0x81000030	TIMER 32 A Timer Tick 6
0x81000034	TIMER 32 A Timer Tick 7
0x81000080	TIMER 32 A Timer counter value
0x810000E0	TIMER 32 A Local register interface information
0x81001000 - 0x810010FF [1)]	TIMER 32 B

Note 1: TIMER 32 B has the same register as TIMER 32 A with a different offset

53.9.2 TIMER32 Interrupt register

Table 855. 0x00 - T32INT - TIMER32 Interrupt register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																					P	TT 7	TT 6	TT 5	TT 4	TT 3	TT 2	TT 1	TT 0	PE	TA
0x0																					0	1	1	1	1	1	1	1	1	0	0
r																					rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*

31: 11 RESERVED

10: Configuration parity error status (P). This parity check covers the 10 configuration registers. Becomes '1' when a configuration parity error is detected.

9: 2 Timer Tick (TT): Status of TIMER32 tick 7 downto 0.

1: Timer Parity Error (PE) - An correctable error has occurred in the timer

0: Timer Alarm (TA) - An uncorrectable error has occurred in the timer

For test-purposes, bits 10:0 in this register are writable. The written data overrides the normal data source for exactly one system clock cycle. This provides a means for triggering the associated interrupt from software.

NOTE: The fields of this register are updated at every clock cycle, depending on the corresponding signals. These values are not latched and polling routines could easily miss signal transitions. To monitor the status of a signal it is preferable to use the Interrupt Status register.

53.9.3 TIMER32 Interrupt Status register

Table 856. 0x04 - T32ISTAT - TIMER32 Interrupt Status register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																					P	TT 7	TT 6	TT 5	TT 4	TT 3	TT 2	TT 1	TT 0	PE	TA
0x0																					0	0	0	0	0	0	0	0	0	0	0
r																					wc	wc	wc	wc	wc	wc	wc	wc	wc	wc	wc

31: 11 RESERVED

10: 0 Bit *x* of this register is set to 1 whenever an interrupt has been generated due to bit *x* of T32INT. Each bit in this stays set to 1 until software clears it by writing 1.

Write 1 to clear.

53.9.4 TIMER32 Interrupt Mask register

Table 857. 0x08 - T32MSK - Interrupt Mask register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																					P	TT 7	TT 6	TT 5	TT 4	TT 3	TT 2	TT 1	TT 0	PE	TA
0x0																					0	0	0	0	0	0	0	0	0	0	0
r																					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

31: 11 RESERVED

10: 0 Bit *x* of this register controls whether the event corresponding to bit *x* of T32INT can generate an interrupt or not. When bit *x* is 0, that event will not generate interrupts.

53.9.5 TIMER32 Interrupt Edge register

Table 858. 0xC - T32EDGE - TIMER32 Interrupt Edge register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																					P	TT7	TT6	TT5	TT4	TT3	TT2	TT1	TT0	PE	TA
0x0																					1	1	1	1	1	1	1	1	1	1	1
r																					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

31: 26 Reserved

25: 0 Bit *x* of this registers controls the polarity of interrupt generation from bit *x* of the T32INT register.
0 => falling edge: interrupt generated when bit *x* changes from 1 to 0
1 => rising edge: interrupt generated when bit *x* changes from 0 to 1

53.9.6 TIMER32 Select Synchronization Source register

Table 859. 0x10 - SYNCSEL- TIMER32 Select Synchronization Source register

31	4	3	2	0
RESERVED	EN	SEL		
0x0	0	0x0		
r	rw	rw		

31: 5 RESERVED

4: EN: Enable the selected synchronization source

3:0 SEL: Select synchronization source

0: Tick Output from External sync control 0

1: Timer32_B tick output 1

2 to 7: reserved

8: Tick Output from External sync control 1

9: Timer32_A tick output 1

10: 15: reserved

53.9.7 TIMER32 Set System Period register

Table 860. 0x14 - SYSPER- TIMER32 Set System Period register

31	28	0
upcnt 3	upcnt 29	
0	0	
rw	rw	

31:27 Config upcnt 3-bits up-counter

28:0 Config upcnt 29-bits up-counter

Note: Used to Set System Period time in system clock cycles

53.9.8 TIMER32 Timer Tick x register

Table 861. 0x18- 0x34 - TIMERTICKx - TIMER32 Timer Tick x register

31	28	0
VALUE_3	VALUE_29	
0	0	
rw	rw	

31:0 Timer System Tick x period in system clock cycles

Note: Register description applies for TIMER32 Timer Tick 0-7 registers.

0x18: TIMERTICK0
0x1C: TIMERTICK1
0x20: TIMERTICK2
0x24: TIMERTICK3
0x28: TIMERTICK4
0x2C: TIMERTICK5
0x30: TIMERTICK6
0x34: TIMERTICK7

53.9.9 TIMER32 Timer Value register

Table 862. 0x80 - TIMERVAL - Timer Counter Value register

31 28	0
VALUE_3	VALUE_ 29
0	0
r	r

31:29 VALUE_3 - current value of the 3-bits up-counter

28:0 VALUE_29 - current value of the 29-bits up-counter

53.9.10 TIMER32 Local register interface information

Table 863. 0xE0 - INFO - TIMER32 Local register interface information

31 24	23 16	15 8	7 0
IRQ	CFG	STAT	IRQ
0x0b	0x0a	0x01	0x0b
r	r	r	r

31: 24 IRQ: Number of Interrupts (same as 7:0)

23: 16 CFG: Number of Configuration registers

15:8 STAT: Number of Status registers

7:0 IRQ: Number of Interrupts

53.10 APWM_TIMER Description

Each of the APWM_AB blocks has a dedicated 27-bits timer. The four APWM_A blocks also share a dedicated 27-bits timer. These timers (hereafter called APWM_TIMER) provide the 24-bit input APWM_timer to the APWM_AB blocks, as per figure 122, and to the APWM_A blocks, as per figure figure 126.

Each 27-bit timer is actually composed by two smaller up-counters, one of 24 bits and one of 3 bits, with reset values configurable through the fields RES_3 and RES_24 of the APWM_TIMER Reset Value Configuration register. The 24 bits up-counter is always counting up. Whenever the 24-bits value becomes equal or higher than the 24-bits reset value (RES_24), the 24-bits up-counter is reset to 0 and the 3-bits up-counter is incremented by 1. When the 3-bits up-counter value becomes equal or higher than the 3-bits reset value (RES_3) and the 24-bits value becomes equal or higher than the 24-bits reset value (RES_24), the 3-bits up-counter is also reset to 0.

The RES_3 and RES_24, configured in the APWM_TIMER Reset Value register, are only transmitted to the timer if the SyncEn signal input to the APWM_TIMER is asserted and one of the conditions configured by the APWM_TIMER Reset Value Synchronization register is met. The reset value currently used by the timer is readable through the APWM_TIMER Current Reset Value register.

The whole 27-bit counter can also be reset to 0 whenever the local tick line selected via the APWM_TIMER Tick Select register is asserted. This is only possible when the reset via tick line is enabled through the APWM_TIMER Tick Select register.

There are 7 local tick lines, generated through a subsystem of the timer. The local tick lines are generated in the following way:

Local Tick 0: Asserted when the value of the 27-bits timer is equal to zero.

Local Tick 1: The signal is asserted when two conditions are met:

- The lower 24 bits (23:0) of the timer value match the value set in the LOW field of the APWM_TIMER Local Tick 1 Configuration register.
- The upper 3 bits (26:24) of the timer value, when interpreted as a decimal number, correspond to a bit in the HIGH field of the APWM_TIMER Local Tick 1 Configuration register that is asserted.

Local Tick 2:

If the field SEL1 of the APWM_TIMER Local Tick Synchronization Configuration Register is equal to 0, the tick signal is asserted using the same comparison logic as for Local Tick 1, but using APWM_TIMER Local Tick **2** Configuration register and with an additional tweak: the values of the register are only updated for the comparison logic if the field WREN1 of the APWM_TIMER Local Tick Synchronization Configuration Register is set to 1.

If the field SEL1 of the APWM_TIMER Local Tick Synchronization Configuration Register is equal to 1, the tick signal is asserted using the same comparison logic as for Local Tick 1, still using APWM_TIMER Local Tick **1** Configuration register but with additional synchronization logic. The value of the register is updated for the comparison logic if the SyncEn signal is asserted and if another condition, configured depending on the SYNCSET1 field of the APWM_TIMER Local Tick Synchronization Configuration Register, is satisfied.

- SYNCSET1 = 0-6: Update when LocalTick1-6 is asserted, if range check is OK
- SYNCSET1 = 7: Update immediately, if range check is OK
- SYNCSET1 = 8-E: Update when LocalTick1-6 is asserted, no range check
- SYNCSET1 = F: Update immediately, no range check.

The range check is a digital circuit that validates the value of the APWM_TIMER Local Tick 1 Configuration register by performing two checks:

- Upper Bits Check (bits 24 to 31): The value of the upper 8 bits is evaluated based on the RES_3 field in the APWM_TIMER Current Reset Value register. For this check to pass, the RES_3 value must be greater than or equal to the position of the first bit set to 1 in these 8 bits.
- Lower Bits Check (bits 0 to 23): The lower 24 bits are compared to the RES_24 field in the APWM_TIMER Current Reset Value register. For this part of the check to pass, the value of the lower 24 bits must be less than or equal to the value of RES_24.

Both parts must pass for the overall range check to be considered OK.

Local tick 3: Same as for Local tick 1 but using the APWM_TIMER Local Tick 3 Configuration register.

Local tick 4: Same as for Local tick 2 but using the APWM_TIMER Local Tick 3 and 4 Configuration registers, and the fields WREN2, SEL2 and SYNCSET2 fields of the APWM_TIMER Local Tick Synchronization Configuration Register.

Local tick 5: Same as for Local tick 1 but using the APWM_TIMER Local Tick 5 Configuration register.

Local tick 6: Same as for Local tick 2 but using the APWM_TIMER Local Tick 5 and 6 Configuration registers, and the fields WREN3, SEL3 and SYNCSET3 fields of the APWM_TIMER Local Tick Synchronization Configuration Register.

53.10.1 Registers

The core is controlled through registers mapped into APB address space.

*Table 864.*APWM_TIMER registers

APB address offset	Register
0x81100000	APWM_TIMER (APWM_A) Status register
0x81100004	APWM_TIMER (APWM_A) Interrupt Status register
0x81100008	APWM_TIMER (APWM_A) Interrupt Mask register
0x8110000C	APWM_TIMER (APWM_A) Interrupt Edge register
0x81100010	APWM_TIMER (APWM_A) Reset Value Configuration register
0x81100014	APWM_TIMER (APWM_A) Tick Select register
0x81100018	APWM_TIMER (APWM_A) Reset Value Synchronization register
0x8110001C	APWM_TIMER (APWM_A) Local Tick 1 Configuration register
0x81100020	APWM_TIMER (APWM_A) Local Tick 2 Configuration register
0x81100024	APWM_TIMER (APWM_A) Local Tick 3 Configuration register
0x81100028	APWM_TIMER (APWM_A) Local Tick 4 Configuration register
0x8110002C	APWM_TIMER (APWM_A) Local Tick 5 Configuration register
0x81100030	APWM_TIMER (APWM_A) Local Tick 6 Configuration register
0x81100034	APWM_TIMER (APWM_A) Local Tick Synchronization Configuration register
0x81100080	APWM_TIMER (APWM_A) Current Reset Value register
0x81100084	APWM_TIMER (APWM_A) Timer Value register
0x81100088	APWM_TIMER (APWM_A) Redundant Timer A Value register
0x8110008C	APWM_TIMER (APWM_A) Redundant Timer B Value register
0x81200000	APWM_TIMER (APWM_AB0) Start
0x81202000	APWM_TIMER (APWM_AB1) Start
0x81204000	APWM_TIMER (APWM_AB2) Start
0x81206000	APWM_TIMER (APWM_AB3) Start

53.10.2 APWM_TIMER Status register

Table 865. 0x00 - APWM_TIMER Status register

31 … 14	13	12 … 9	8	7	6 … 0
RESERVED	E	SYNC_EN	R	R	TICK
NR	0	0	0	0	0
r	rw	rw	rw	rw	rw

NOTE: The rw fields of this register can be written in order to trigger an interrupt, as per the conditions described in the APWM_TIMER Interrupt Status register.

NOTE: The fields of this register are updated at every clock cycle, depending on the corresponding signals. These values are not latched and polling routines could easily miss signal transitions. To monitor the status of a signal it is preferable to use the Interrupt Status register.

13	Error - There has been an uncorrectable error in the Timer configurations registers. For the details of the protection scheme, see APWM_TIMER Check Bits registers.
12:9	Synchronization Enable (SYNC_EN) - each bit shows the status of the input SYNC_EN signals 0-3
8	RESERVED
7	RESERVED
6:0	Localtick inputs (TICK) - each bit shows the status of the input tickline 0-6

53.10.3 APWM_TIMER Interrupt Status register

Table 866. 0x04 - APWM_TIMER Interrupt Status register

31 … 14	13	12 … 9	8	7	6 … 0
RESERVED	E	SYNC_EN	PE	TA	TICK
NR	0	0		0	0
r	rw	rw		rw	rw

A bit in this register (and an interrupt) will only be asserted if the corresponding bit in the APWM_-TIMER Interrupt Mask register is set to 1. When this condition is met, the bit will be asserted whenever the corresponding bit in the APWM_TIMER Status register changes to match the value of the same bit in the APWM_TIMER Interrupt Edge register.

Write 1 to clear.

53.10.4 APWM_TIMER Interrupt Mask register

Table 867. 0x08 - APWM_TIMER Interrupt Mask register

31 … 14	13	12 … 9	8	7	6 … 0
RESERVED	E	SYNC_EN	PE	TA	TICK
NR	0	0		0	0
r	rw	rw		rw	rw

Enable interrupts by setting the bits of this register to 1.

53.10.5 APWM_TIMER Interrupt Edge register

Table 868. 0xC - APWM_TIMER Interrupt Edge register

31 … 14	13	12 … 9	8	7	6 … 0
RESERVED	E	SYNC_EN	PE	TA	TICK
0	0	0		0	0
r	rw	rw		rw	rw

See APWM_TIMER Interrupt Status register.

53.10.6 APWM_TIMER Reset Value Configuration register

Table 869. 0x10 - APWM_TIMER Reset Value Configuration register

31 27	26 24	23 0
RESERVED	RES_3	RES_24
NR	0	0
r	rw	rw

27:24 RES_3 - reset value for the 3-bits up-counter

23:0 RES_24- reset value for the 24-bits up-counter

53.10.7 APWM_TIMER Tick Select register

Table 870. 0x14 - APWM_TIMER Tick Select register

31 5	4	3 0
RESERVED	EN	TICKSEL
NR	0	0
r	rw	rw

4 EN - Enable the reset via tick line

3:0 TICKSEL- Select which tick line that can reset the timer to 0

53.10.8 APWM_TIMER Reset Value Synchronization register

Table 871. 0x18 - APWM_TIMER Reset Value Synchronization register

31 3	2 0
RESERVED	SEL
NR	“111”
r	rw

2:0 SEL - Select which event allows for the RES_24 and RES_3 values from the Reset Value Configuration register Current Reset Value register.

-> When “111”: always update,

-> When others: select which local tick line should be asserted for the update to happen.

NOTE: The updates only happen if the conditions above are met AND the SyncEn input is asserted

53.10.9 APWM_TIMER Local Tick 1-6 Configuration registers

Table 872. 0x1C - 0x30 - APWM_TIMER Local Tick 1-6 Configuration register

31 24	23 0
HIGH	LOW
0	0
rw	rw

The fields of this register are used for the local tick lines generation. See description in section 53.10.

53.10.10APWM_TIMER Local Tick Synchronization Configuration registers

Table 873. 0x34 - APWM_TIMER Local Tick Sync Configuration register

31 8	6	5	4	3	2	1	0
RESERVED	SYNCSET3	SYNCSET2	SYNCSET1	SEL3	WREN3	SEL2	WREN2	SEL1	WREN1
NR	0	0	0	0	0	0	0	0	0
r	rw	rw	rw	rw	rw	rw	rw	rw	rw

Table 873. 0x34 - APWM_TIMER Local Tick Sync Configuration register

The usage of the fields of this register are described in section 53.10.

53.10.11APWM_TIMER Current Reset Value register

Table 874. 0x80 - APWM_TIMER Reset Value register

31 27	26 24	23 0
RESERVED	RES_3	RES_24
NR	0	0
r	r	r

27:24 RES_3 - reset value currently used by the timer for the 3-bits up-counter

23:0 RES_24 - reset value currently used by the timer for the 24-bits up-counter

53.10.12APWM_TIMER Timer Value register

Table 875. 0x84 - APWM_TIMER Timer Value register

31 27	26 24	23 0
RESERVED	VALUE_3	VALUE_24
NR	0	0
r	r	r

27:24 VALUE_3 - current value of the 3-bits up-counter

23:0 VALUE_24 - current value of the 24-bits up-counter

53.11 APWM_TIMER Check BitssAPWM Synchronization Register

53.11.1 Registers

The core is programmed through registers mapped into APB address space.

*Table 876.*SYNC Enable registers

APB address offset	Register
0x81002000	SyncEnReg Status register
0x81002004	SyncEnReg Interrupt register
0x81002008	SyncEnReg Mask register
0x8100200C	SyncEnReg Interrupt level register
0x81002010	SyncEn28 RTA0 set mask for A and AB
0x81002014	SyncEn28 RTA0 Reset mask for A and AB
0x81002018	SyncEn28 RTA1 set mask for A and AB
0x8100201C	SyncEn28 RTA1 Reset mask for A and AB
0x81002020	SyncEn28 CPU0 set mask for A and AB
0x81002024	SyncEn28 CPU0 reset mask for A and AB
0x81002028	SyncEn24 RTA0 set mask for CF
0x8100202C	SyncEn24 RTA0 Reset mask for CF
0x81002030	SyncEn24 RTA1 set mask for CF
0x81002034	SyncEn24 RTA1 Reset mask for CF
0x81002038	SyncEn24 CPU0 set mask for CF
0x8100203C	SyncEn24 CPU0 reset mask for CF
0x81002080	SyncEn28 Status of register synchronization for A and AB
0x81002084	SyncEn24 Status of register synchronization for CF
0x810020E0	SyncEnReg Local register interface information

53.11.2 SyncEnReg Interrupt register

Table 877. 0x00 - SYNCENINT - SyncEnReg Interrupt register

31 30 29	28	27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
RESERVED	P	SyncEn28
0x0	0	0x182865
r	rw*	rw*

31: 29 RESERVED

28: Configuration parity error status (P). This parity check covers the 10 configuration registers. Becomes '1' when a configuration parity error is detected.

27:0 SyncEn28: Status of synchronization of A and AB PWMs

Refer section 53.11.6 for bit definition

For test-purposes, bits 28:0 in this register are writable. The written data overrides the normal data source for exactly one system clock cycle. This provides a means for triggering the associated interrupt from software.

NOTE: The fields of this register are updated at every clock cycle, depending on the corresponding signals. These values are not latched and polling routines could easily miss signal transitions. To monitor the status of a signal it is preferable to use the Interrupt Status register.

53.11.3 SyncEnReg Interrupt Status register

Table 878. 0x04 - SYNCENSTAT - SyncEnReg Interrupt Status register

31 30 29	28	27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
RESERVED	P	SyncEn28
0x0	0	0x0000000
r	wc	wc

31: 29 RESERVED

28: 0 Bit *x* of this register is set to 1 whenever an interrupt has been generated due to bit *x* of SYNCENINT. Each bit in this stays set to 1 until software clears it by writing 1.

Write 1 to clear.

53.11.4 SyncEnReg Interrupt Mask register

Table 879. 0x08 - SYNCENMSK - Interrupt Mask register

31 30 29	28	27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
RESERVED	P	SyncEn28
0x0	0	0x0000000
r	rw	rw

31: 29 RESERVED

28: 0 Bit *x* of this register controls whether the event corresponding to bit *x* of SYNCENINT can generate an interrupt or not. When bit *x* is 0, that event will not generate interrupts.

53.11.5 SyncEnReg Interrupt Edge register

Table 880. 0xC - SYNCENEDGE - SyncEnReg Interrupt Edge register

31 30 29	28	27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
RESERVED	P	SyncEn28
0x0	1	0xFFFFFFF
r	rw	rw

31: 29 Reserved

28: 0 Bit *x* of this registers controls the polarity of interrupt generation from bit *x* of the SYNCENINT register.
0 => falling edge: interrupt generated when bit *x* changes from 1 to 0
1 => rising edge: interrupt generated when bit *x* changes from 0 to 1

53.11.6 SyncEn28 RTA0 set mask for A, AB and TIMER27 register

Table 881. 0x10 - RTA0SET - SyncEn28 RTA0 set mask for A, AB and TIMER27 register

31 30 29 28	27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
RESERVED	RTA0SET
0x0	0x0
r	rw

31: 28 RESERVED

27:0 RTA0SETA: RTA0 set mask for A, AB and TIMER27

APWM_AB 0 Timer27_RegSync SyncEn
TimerRegSync_27 = RTA0SET(0)
APWM_DualTickReg = RTA0SET(4)
APWM_DualTickReg = RTA0SET(8)
APWM_DualTickReg = RTA0SET(12)

APWM_AB 1 Timer27_RegSync SyncEn
TimerRegSync_27 = RTA0SET(1)
APWM_DualTickReg = RTA0SET(5)
APWM_DualTickReg = RTA0SET(9)
APWM_DualTickReg = RTA0SET(13)

APWM_AB 2 Timer27_RegSync SyncEn
TimerRegSync_27 = RTA0SET(2)
APWM_DualTickReg = RTA0SET(6)
APWM_DualTickReg = RTA0SET(10)
APWM_DualTickReg = RTA0SET(14)

APWM_AB 3 Timer27_RegSync SyncEn
TimerRegSync_27 = RTA0SET(3)
APWM_DualTickReg = RTA0SET(7)
APWM_DualTickReg = RTA0SET(11)
APWM_DualTickReg = RTA0SET(15)

APWM_A Timer27_RegSync SyncEn
TimerRegSync_27 = RTA0SET(16)
APWM_DualTickReg = RTA0SET(17)
APWM_DualTickReg = RTA0SET(18)
APWM_DualTickReg = RTA0SET(19)

APWM Timer27_RegSync 0 SyncEn
TimerRegSync_27 = RTA0SET(20)
APWM_DualTickReg = RTA0SET(21)
APWM_DualTickReg = RTA0SET(22)
APWM_DualTickReg = RTA0SET(23)

APWM Timer27_RegSync 1 SyncEn
TimerRegSync_27 = RTA0SET(24)
APWM_DualTickReg = RTA0SET(25)
APWM_DualTickReg = RTA0SET(26)
APWM_DualTickReg = RTA0SET(27)

53.11.7 SyncEn28 RTA0 reset mask for A, AB and TIMER27 register

Table 882. 0x14 - RTA0RST - SyncEn28 RTA0 reset mask for A, AB and TIMER27 register

31 30 29 28	27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
RESERVED	RTA0RST
0x0	0x0
r	rw

31: 28 RESERVED

27:0 RTA0RST: RTA0 reset mask for A, AB and TIMER27

Refer section 53.11.6 for bit definition

53.11.8 SyncEn28 RTA1 set mask for A, AB and TIMER27 register

Table 883. 0x18 - RTA1SET - SyncEn28 RTA1 set mask for A, AB and TIMER27 register

31 30 29 28	27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
RESERVED	RTA1SET
0x0	0x0
r	rw

31: 28 RESERVED

27:0 RTA1SET: RTA1 set mask for A, AB and TIMER27

Refer section 53.11.6 for bit definition

53.11.9 SyncEn28 RTA1 reset mask for A, AB and TIMER27 register

Table 884. 0x1C - RTA1RST - SyncEn28 RTA1 reset mask for A, AB and TIMER27 register

31 30 29 28	27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
RESERVED	RTA1RST
0x0	0x0
r	rw

31: 28 RESERVED

27:0 RTA1RST: RTA1 reset mask for A, AB and TIMER27

Refer section 53.11.6 for bit definition

53.11.10SyncEn28 CPU0 set mask for A, AB and TIMER27 register

Table 885. 0x20 - CPU0SET - SyncEn28 CPU0 set mask for A, AB and TIMER27 register

31 30 29 28	27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
RESERVED	CPU0SET
0x0	0x0
r	rw

31: 28 RESERVED

27:0 CPU0SET: CPU0 set mask for A, AB and TIMER27

Refer section 53.11.6 for bit definition

53.11.11SyncEn28 CPU0 reset mask for A, AB and TIMER27 register

Table 886. 0x24 - CPU0RST - SyncEn28 CPU0 reset mask for A, AB and TIMER27 register

31 30 29 28	27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
RESERVED	CPU0RST
0x0	0x0

Table 886. 0x24 - CPU0RST - SyncEn28 CPU0 reset mask for A, AB and TIMER27 register

r	rw

31: 28	RESERVED
27:0	CPU0RST: CPU0 reset mask for A, AB and TIMER27 Refer section 53.11.6 for bit definition

53.11.12SyncEn24 RTA0 set mask for CF register

Table 887. 0x28 - RTA0SETCF - SyncEn24 RTA0 set mask for CF register

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
RESERVED	RTA0SETCF
0x0	0x0
r	rw

31: 24 RESERVED

23:0 RTA0SETCF: RTA0 set mask for CF

APWM_CF 0 Timer27_RegSync SyncEn
TimerRegSync_27 = RTA0SETCF(0)
APWM_DualTickReg = RTA0SETCF(1)
APWM_DualTickReg = RTA0SETCF(2)
APWM_DualTickReg = RTA0SETCF(3)
APWM_CF0 0 RegSync = RTA0SETCF(4)
APWM_CF0 1 RegSync = RTA0SETCF(5)
APWM_CF0 2 RegSync = RTA0SETCF(6)
APWM_CF0 3 RegSync = RTA0SETCF(7)

APWM_CF 1 Timer27_RegSync SyncEn
TimerRegSync_27 = RTA0SETCF(8)
APWM_DualTickReg = RTA0SETCF(9)
APWM_DualTickReg = RTA0SETCF(10)
APWM_DualTickReg = RTA0SETCF(11)
APWM_CF1 0 RegSync = RTA0SETCF(12)
APWM_CF1 1 RegSync = RTA0SETCF(13)
APWM_CF1 2 RegSync = RTA0SETCF(14)
APWM_CF1 3 RegSync = RTA0SETCF(15)

TimerRegSync_27 = RTA0SETCF(16)
APWM_DualTickReg = RTA0SETCF(17)
APWM_DualTickReg = RTA0SETCF(18)
APWM_DualTickReg = RTA0SETCF(19)
APWM_CF2 0 RegSync = RTA0SETCF(20
APWM_CF2 1 RegSync = RTA0SETCF(21)
APWM_CF2 2 RegSync = RTA0SETCF(22)
APWM_CF2 3 RegSync = RTA0SETCF(23)

53.11.13SyncEn24 RTA0 reset mask for CF register

Table 888. 0x2C - RTA0RSTCF - SyncEn24 RTA0 reset mask for CF register

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
RESERVED	RTA0RSTCF
0x0	0x0
r	rw

Table 888. 0x2C - RTA0RSTCF - SyncEn24 RTA0 reset mask for CF register

31: 28	RESERVED
27:0	RTA0RSTCF: RTA0 reset mask for CF
	Refer section 53.11.12 for bit definition

53.11.14SyncEn24 RTA1 set mask for CF register

Table 889. 0x30 - RTA1SETCF - SyncEn24 RTA1 set mask for CF register

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
RESERVED			RTA1SETCF
0x0			0x0
r			rw

31: 28	RESERVED
27:0	RTA1SETCF: RTA1 set mask for CF
	Refer section 53.11.12 for bit definition

53.11.15SyncEn24 RTA1 reset mask for CF register

Table 890. 0x34 - RTA1RSTCF - SyncEn24 RTA1 reset mask for CF register

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
RESERVED			RTA1RSTCF
0x0			0x0
r			rw

31: 28	RESERVED
27:0	RTA1RSTCF: RTA1 reset mask for CF
	Refer section 53.11.12 for bit definition

53.11.16SyncEn24 CPU0 set mask for CF register

Table 891. 0x38 - CPU0SETCF - SyncEn24 CPU0 set mask for CF register

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
RESERVED			CPU0SETCF
0x0			0x0
r			rw

31: 24	RESERVED
23:0	CPU0SETCF: CPU0 set mask for CF
	Refer section 53.11.12 for bit definition

53.11.17SyncEn24 CPU0 reset mask for CF register

Table 892. 0x3C - CPU0RSTCF - SyncEn24 CPU0 reset mask for CF register

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
RESERVED			CPU0RSTCF
0x0			0x0
r			rw

31: 24	RESERVED
23:0	CPU0RSTCF: CPU0 reset mask for CF
	Refer section 53.11.12 for bit definition

53.11.18SyncEn28 Status of register synchronization for A, AB and TIMER27

Table 893. 0x80 - STATUS - Status of register synchronization for A, AB and TIMER27

31 30 29 28	27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
RESERVED	STATUS
0x0	0x0
r	r

31: 28 RESERVED

27:0 STATUS: Status of register synchronization for APWM A, AB and TIMER27

Refer section 53.11.6 for bit definition

53.11.19SyncEn24 Status of register synchronization for CF

Table 894. 0x84 - Status of register synchronization for CF

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
RESERVED	STATUSCF
0x0	0x0
r	r

31: 24 RESERVED

23:0 STATUS: Status of register synchronization for APWM CF

Refer section 53.11.12 for bit definition

53.11.20SyncEnReg Local register interface information

Table 895. 0xE0 - INFO - SyncEnReg Local register interface information

31 24	23 16	15 8	7 0
IRQ	CFG	STAT	IRQ
0x20	0x0C	0x02	0x20
r	r	r	r

31: 24 IRQ: Number of Interrupts (same as 7:0)

23: 16 CFG: Number of Configuration registers

15:8 STAT: Number of Status registers

7:0 IRQ: Number of Interrupts

53.12 APWM Timestamp Register

53.12.1 Registers

The core is programmed through registers mapped into APB address space.

Table 896. Timestamp registers

APB address offset	Register
0x8110C000	Timestamp Status register
0x8110C004	Timestamp Interrupt register
0x8110C008	Timestamp Mask register
0x8110C00C	Timestamp Interrupt level register
0x8110C010	Timestamp 0 Config Register
0x8110C014	Timestamp 1 Config Register
0x8110C018	Timestamp 2 Config Register
0x8110C01C	Timestamp 3 Config Register
0x8110C020	Timestamp 4 Config Register
0x8110C024	Timestamp 5 Config Register
0x8110C028	Timestamp 6 Config Register
0x8110C02C	Timestamp 7 Config Register
0x8110C030	Timestamp 8 Config Register
0x8110C034	Timestamp 9 Config Register
0x8110C038	Timestamp 10 Config Register
0x8110C03C	Timestamp 11 Config Register
0x8110C040	Timestamp 12 Config Register
0x8110C044	Timestamp 13 Config Register
0x8110C048	Timestamp 14 Config Register
0x8110C04C	Timestamp 15 Config Register
0x8110C080	Timestamp 0 Status Register
0x8110C084	Timestamp 1 Status Register
0x8110C088	Timestamp 2 Status Register
0x8110C08C	Timestamp 3 Status Register
0x8110C090	Timestamp 4 Status Register
0x8110C094	Timestamp 5 Status Register
0x8110C098	Timestamp 6 Status Register
0x8110C09C	Timestamp 7 Status Register
0x8110C0A0	Timestamp 8 Status Register
0x8110C0A4	Timestamp 9 Status Register
0x8110C0A8	Timestamp 10 Status Register
0x8110C0AC	Timestamp 11 Status Register
0x8110C0B0	Timestamp 12 Status Register
0x8110C0B4	Timestamp 13 Status Register
0x8110C0B8	Timestamp 14 Status Register
0x8110C0BC	Timestamp 15 Status Register
0x8110C0E0	Timestamp Local register interface information

53.12.2 Timestamp Interrupt Register

Table 897. 0x00 - TSINT - Timestamp Interrupt Register

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17	16	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
RESERVED	P	VALID
0x0	0	0x0000
r	rw*	rw*

31: 17 RESERVED

16: Configuration parity error status (P). This parity check covers the 10 configuration registers. Becomes '1' when a configuration parity error is detected.

15:0 VALID: Timestamp event occurred

For test-purposes, bits 16:0 in this register are writable. The written data overrides the normal data source for exactly one system clock cycle. This provides a means for triggering the associated interrupt from software.

NOTE: The fields of this register are updated at every clock cycle, depending on the corresponding signals. These values are not latched and polling routines could easily miss signal transitions. To monitor the status of a signal it is preferable to use the Interrupt Status register.

53.12.3 Timestamp Interrupt Status Register

Table 898. 0x04 - TSSTAT - Timestamp Interrupt Status Register

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17	16	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
RESERVED	P	VALID
0x0	0	0x0000
r	wc	wc

31: 17 RESERVED

16: 0 Bit *x* of this register is set to 1 whenever an interrupt has been generated due to bit *x* of TSINT. Each bit in this stays set to 1 until software clears it by writing 1.

Write 1 to clear.

53.12.4 Timestamp Interrupt Mask register

Table 899. 0x08 - TSMSK - Interrupt Mask register

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17	16	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
RESERVED	P	VALID
0x0	0	0x0000
r	rw	rw

31: 17 RESERVED

16: 0 Bit *x* of this register controls whether the event corresponding to bit *x* of TS INT can generate an interrupt or not. When bit *x* is 0, that event will not generate interrupts.

53.12.5 Timestamp Interrupt Edge register

Table 900. 0xC - TSEDGE - Timestamp Interrupt Edge register

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
RESERVED	P	VALID
0x0	1	0xFFFF
r	rw	rw

31: 17 Reserved

16: 0 Bit *x* of this registers controls the polarity of interrupt generation from bit *x* of the TS INT register.
0 => falling edge: interrupt generated when bit *x* changes from 1 to 0
1 => rising edge: interrupt generated when bit *x* changes from 0 to 1

53.12.6 Timestamp 0 config register

Table 901. 0x10 - TS0CNF - Timestamp 0 config register

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6	5	4 3 2 1 0
RESERVED	E	SEL
0x0	0	0
r	rw	rw

31: 6 RESERVED

5: E: Select positive or negative edge for the time stamp.
'0': Negative Edge
'1': Positive Edge

4:0 SEL: Select timestamp source

Note: Similar set of bit fields are available for the Timestamp 1-15 config registers.

The SEL value for the inputs are describe below,

Reg name	Address Offset	SEL input
Timestamp 0 config registers	0x10	SEL value 0 to 2: State_G*(0 to 2) SEL value 3: State_G(0) SEL value 4 to 11: GPIO(1 to 8) SEL value 12 to 31: ACOMP_SYNC(0 to 19)
Timestamp 1 config registers	0x14	
Timestamp 2 config registers	0x18	
Timestamp 3 config registers	0x1C	
Timestamp 4 config registers	0x20	SEL value 0 to 2: State_G*(3 to 5) SEL value 3: State_G(1) SEL value 4 to 11: GPIO(9 to 16) SEL value 12 to 31: ACOMP_SYNC(0 to 19)
Timestamp 5 config registers	0x24	
Timestamp 6 config registers	0x28	
Timestamp 7 config registers	0x2C	
Timestamp 8 config registers	0x30	SEL value 0 to 2: State_G*(6 to 8) SEL value 3: State_G(2) SEL value 4 to 11: GPIO(17 to 24) SEL value 12 to 31: ACOMP_SYNC(0 to 19)
Timestamp 9 config registers	0x34	
Timestamp 10 config registers	0x38	
Timestamp 11 config registers	0x3C	
Timestamp 12 config registers	0x40	SEL value 0 to 2: State_G*(9 to 11) SEL value 3: State_G(3) SEL value 4 to 11: GPIO(25 to 32) SEL value 12 to 31: ACOMP_SYNC(0 to 19)
Timestamp 13 config registers	0x44	
Timestamp 14 config registers	0x48	
Timestamp 15 config registers	0x4C	

53.12.7 Timestamp 0 Status register

Table 902. 0x80 - STATUS - Timestamp 0 Status register

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
TimeStamp
0x0
r

31:0 TimeStamp: Time counter value when the timestamp event occurred.

Note: Similar bit fields are available for the Timestamp 1-15 Status registers with offset 0x84-0xBC.

53.12.8 Timestamp local register interface information

Table 903. 0xE0 - INFO - Timestamp local register interface information

31 24	23 16	15 8	7 0
IRQ	CFG	STAT	IRQ
0x11	0x10	0x10	0x11
r	r	r	r

31: 24	IRQ: Number of Interrupts (same as 7:0)
23: 16	CFG: Number of Configuration registers
15:8	STAT: Number of Status registers
7:0	IRQ: Number of Interrupts

53.13 Application Notes for APWM Controllers

53.13.1 APWM_AB application: DC/DC converter connection examples

Examples of GR716B connection diagrams for commonly used DC/DC topologies in the industry are presented in figure 133. Frequency controlled topologies such as resonant LLC converters are also supported, since the PWM frequency from APWM Timer is cycle-by-cycle reprogrammable from RTA. An example how to control a more complex DC/DC topology such as a phase-shifted full-bridge converter (PSFB) is presented in section 53.4. That DC/DC converter uses APWM_CF and APWM_G, in addition to APWM_AB.

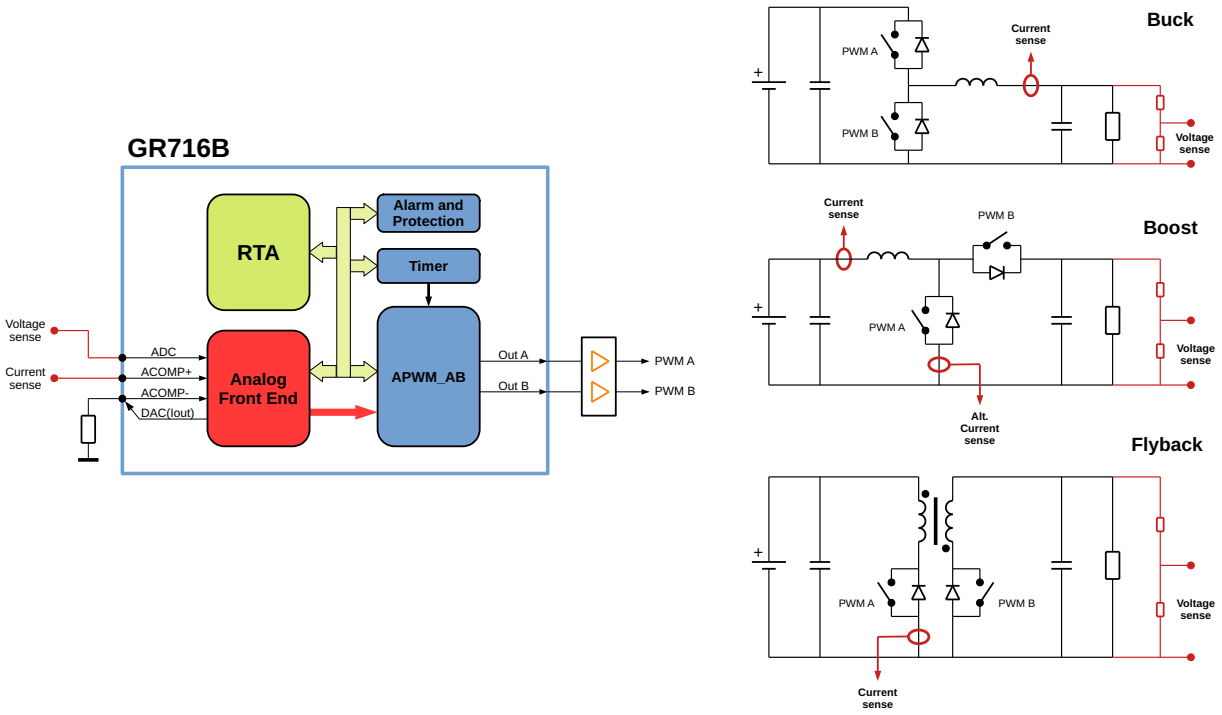


Figure 133. Simplified application schematics of DC/DC converters controlled by APWM_AB: step-down (buck), step-up (boost), and flyback (buck-boost).

The step-down (buck) converter only can convert to lower output voltage than its input voltage, therefrom the name step-down converter. The step-up (boost) converter only can convert to higher output voltage, which means that special care must be taken during start-up from zero output voltage until it is higher than the input voltage. The flyback (buck-boost) can convert seamlessly to lower or higher output voltage than its input voltage.

The Out A signal controls the duty-cycle switch, i.e., the PWM A switch, which always is required in these switching power topologies. The Out B signal controls the synchronous-rectification switch, i.e., the PWM B switch. This switch is optional, and can be replaced by a diode when power efficiency is of less importance. In the order of 10A and higher output currents, this switch is normally beneficial to implement, and for the order of 1A and lower it is often not space and cost efficient compared to the benefits.

The class of point-of-load (POL) converters is commonly implemented with step-down (buck) converters. A main reason is that the output voltage from a POL converter is usually rather low, commonly in the range 0.5-5 V, and needs to be accurately regulated close to the physical point of load to not get too large voltage drop in the supply distribution net. The maximum output current to be designed for a POL converter can vary in a large span, usually somewhere in the range of 1A to 50A. When designing for the upper end of this range, large load-current changes can occur, which also can have fast rise and fall times in many applications, e.g., in supplies for digital ICs such as FPGAs.

A special duty-cycle control case for POL converters is, therefore, to handle extraordinarily large and fast load-current changes. Then, the duty cycle needs to change quickly and to a large extent, preferably with a step of the whole allowed duty-cycle change within one or a couple of power-switch cycles. This may be difficult to accommodate fast enough in a conventional digital control-law algorithm. First, the load change must be detected by the ADC voltage measurement, including sample sanity check to not jump to conclusions based on one faulty or disturbed ADC sample. Second, the RTA needs to detect and conclude that a large load-step has occurred, and apply a special control-law algorithm for a limited period in time.

An alternative solution for such load change cases is provided by GR716B. An analog comparator, ACOMP, can be configured to detect a fixed converter output-voltage level, and trig the one-shot timer in APWM_G, see figure 134. The one-shot timer output is combined with the APWM_AB outputs in two other APWM_G such that when no ACOMP has triggered the APWM_AB controls the power switches as in normal APWM_AB operation. When ACOMP has triggered, the power switches are controlled by the one-shot timer such that a duty-switch turn-on is generated if the output voltage is too low, or a rectification-switch turn-on is generated if the output voltage is too high. The pulse length from the one-shot timer is configurable from RTA.

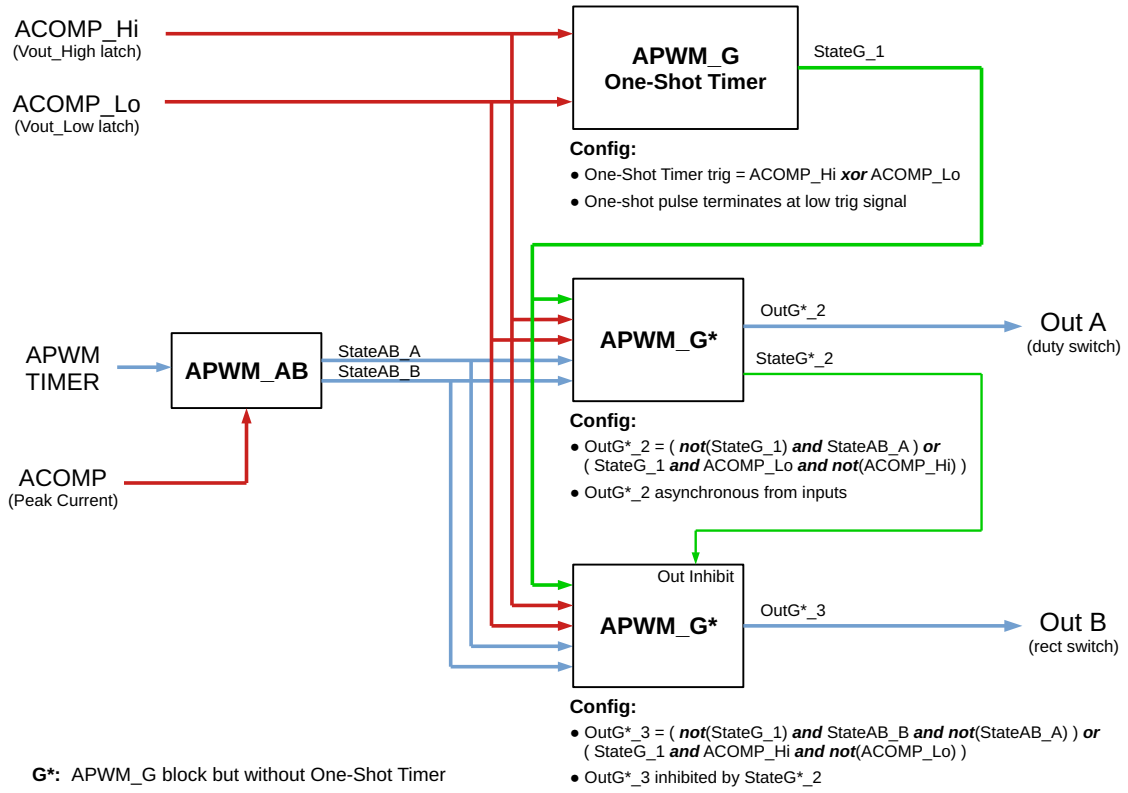


Figure 134. GR716B configuration diagram for ACOMP level-triggered duty control, to handle large and fast load-current changes in step-down (buck) converters in general.

This solution gives the converter similar reaction time to load changes as in hysteretic-controlled converters, which are known to perform well in POL applications. An essential benefit with ACOMP level-triggered duty control is the fast ACOMP reaction time of about 100ns, when configured to SET-hard mode. Moreover, the risk of uncertain delay due to ADC sample sanity-check error is eliminated. Hence, the extraordinary load change cases for POL converters are more robustly and safely handled by this GR716B solution than ADC-based solutions, especially in space environment where ADC samples can be subject to heavy-ion corruption randomly.

53.13.2 APWM_A application: A low-power DC/DC converter

The low-power DC/DC converter in figure 135 will be used to present the detailed operation of APWM_A. The controller is based on ACOMP in peak-voltage mode, without use of RTA, ADC or DAC. A controller for multiple low-power DC/DC converters, based on one ACOMP and APWM_A combined with multiple APWM_CF blocks, is presented thereafter.

General note: Component types and values in schematics should be interpreted as recommendations of right order of magnitude, presented to illustrate the functionality of the circuit topologies. They do not guarantee performance in worst-case corners, which needs to be guaranteed by detailed parts stress and worst case analyses based on your exact component types and values including tolerances and all other electrical properties in your designs.

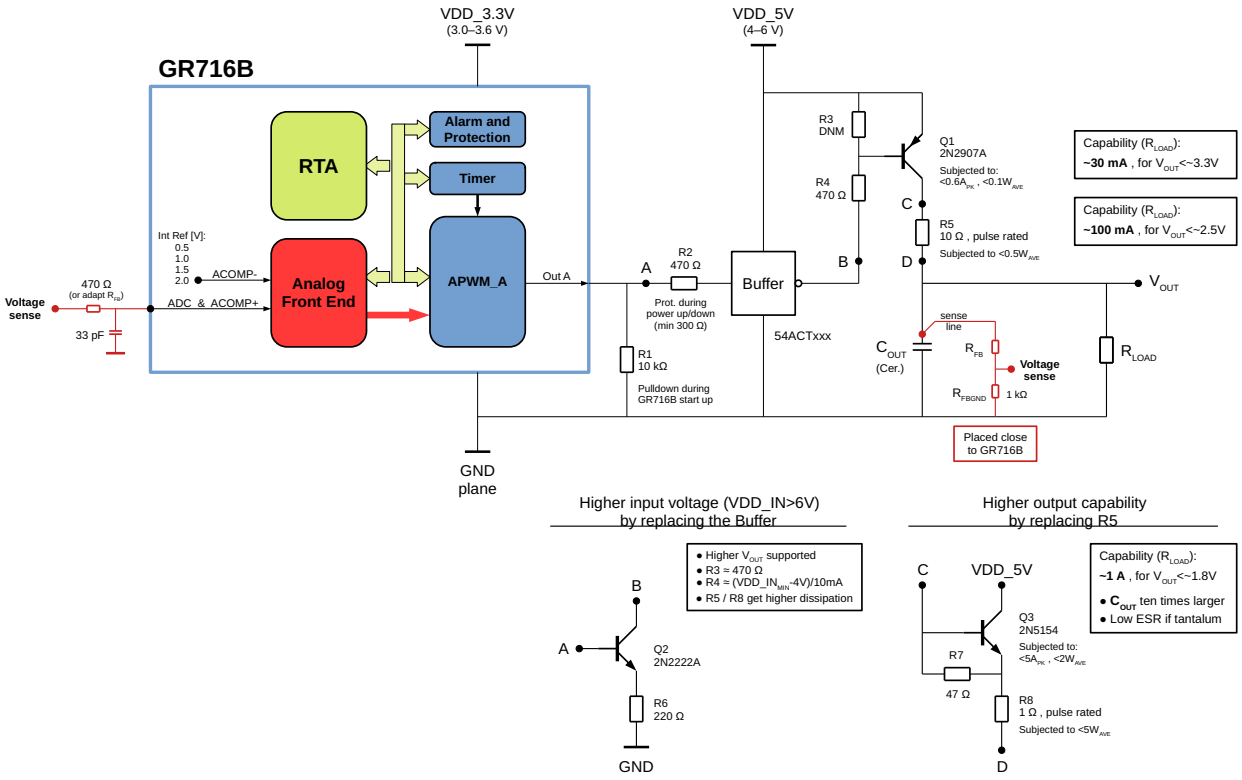


Figure 135. Simplified application schematic of a low-power DC/DC converter controlled by APWM_A. Component types and values only illustrate topology functionality, and parts stress and worst-case corner performance are to be guaranteed by your detailed circuit analyses.

Time slot configuration in the APWM Timer is set to one slot, i.e., only slot 0 exists, since there is only one output voltage to be regulated by this APWM_A block. Alternatively, if other APWM_A blocks need this APWM Timer to be configured for more than one time slot, the APWM_A in this example will simply be configured to have all its time slots enabled for operation.

The APWM_A Out signal turns the power switch, Q1, on at APWM Timer equal to zero. This switching can cause disturbance transients on the output voltage and the feedback signal, 'Voltage sense', which could cause ACOMP to falsely turn off the duty cycle immediately. Therefore, it is advisable to configure adequate leading-edge blanking (LEB) time, and maybe implement a low-pass RC link on the ACOMP input pin, compatible with the ACOMP LEB ground switch. A source resistance in the order of 1kΩ and a capacitance of 33pF would be suitable together with LEB mode 1, and a realistic LEB time may be 100-300 ns. At least 50ns is advisable, to ensure that ACOMP configured to fast mode has enough time to go low during the LEB time.

After Q1 has turned on, current is fed into the large output capacitor, C_{OUT}, and the output voltage, V_{OUT}, starts to increase. When the output voltage reaches the trig level of ACOMP, the duty cycle ends and the power switch turns off. There is no inductance with stored energy in this converter, so the output voltage stops to increase instantly at the end of the duty cycle. Thus, the positive-peak output voltage, $V_{POS,PK}$, never goes higher than the ACOMP trig level, V_{TRIG}, except for the switch turn-off delay in the PCB implementation.

The negative-peak output voltage, $V_{NEG,PK}$, is designed by the value of the output capacitor, C_{OUT}, together with the load current, I_{LOAD}, and the PWM period time, T_{PWM}. The worst (lowest) negative-peak output voltage will be:

$$V_{NEG,PK,min} = V_{POS,PK,min} - V_{DROP,max} = V_{TRIG,min} - I_{LOAD,max} * T_{PWM,max} / C_{OUT,min}$$

For this equation to be applicable, the current through the switch must be higher than the maximum allowed load current. Otherwise, the output voltage would continue to decrease even when the switch is turned on with 100% duty cycle. The current through the switch is designed by the value of the series resistor, R5. Here, also the maximum saturation voltage across Q1 needs to be taken into account. This current should be designed high enough with adequate margin to handle specified load current transients, spread of input supply voltage, VDD_IN (VDD_5V), and all component tolerances. This current calculation gives the maximum allowed limit for R5.

The minimum allowed limit for R5, which will set the maximum instant current through R5 and Q1, will be limited by the maximum rating of pulse current for the resistor, R5, and the switch, Q1. The maximum instant current occurs during start up of this converter from zero output voltage. From the very first switching cycle at start up, the whole input supply voltage, VDD_IN (VDD_5V), can be across R5, so the pulse current can be maximum: I_{max} = $VDD\_IN_{max}/R5_{min}$. Both R5 and Q1 must withstand this pulse current, and the resulting pulse power dissipation, throughout the start-up ramp on the output voltage, V_{OUT}. Therefore, pulse rating must be specified in the datasheet for the selected resistor type, and the absolute maximum limit for current should not be exceeded for the transistor.

The maximum voltage drop, $V_{DROP,max}$ = $I_{LOAD,max}$ * $T_{PWM,max}$ / $C_{OUT,min}$, from the equation above, will be the output peak-to-peak triangular voltage ripple, and sets the required minimum output capacitance, $C_{OUT,min}$. For example, maximum allowed ripple of $30mV_{PK-PK}$, maximum load current of 100mA, and PWM period of 10us give minimum output capacitance of 33uF. Hence, for example, $47uF_{nom}$ could be suitable to provide adequate margin for tolerances and load transients, but it depends on your application situation.

To get higher output current capability than R5=10Ω and Q1 (2N2907A, 0.6A rating) provide, R5 can be replaced by R7=47ohm, R8=1Ω and Q3 (2N5154, 5A rating) in the C and D connection points. This solution may provide about 1A output current, but it depends on the required output voltage level compared to the available input voltage level. Another way to get higher output current capability is to put more than one block of R3, R4, R5 and Q1 in parallel, in the connection points B and D. Each block needs its own R4=470Ω and R5=10Ω, to ensure equal current sharing. If more than three such blocks would be put in parallel, more than one 54ACT Buffer output is needed.

Alternatively, R5 can be changed to 1Ω and Q1 (2N2907A) changed to Q1=2N5153 (5A rating), with base drive from Q2 (2N2222A) and R6=39Ω, R3=100Ω and R4=~(VDD_IN_{min}-4V)/50mA (note that R4=0 for VDD_5V). This design may provide about 1A output current, and the dissipation and current properties stated for Q3 and R8 in figure 135 become applicable for the new Q1 and R5. For higher input voltage (see below), the choice of R4 can significantly affect the power dissipation in Q2, where higher R4 value will decrease Q2 dissipation. Anyhow, if the parts stress analysis shows that Q2=2N5154 is required, the Q3=2N5154 solution to get higher output current capability would be advisable instead of this whole Q1=2N5153 solution.

To handle higher input voltage, VDD_IN, than a VDD_5V supply bus, the Buffer can be replaced by R6 and Q2 (2N2222A) in the A and B connection points. The maximum input voltage will be limited by the collector voltage rating, V_{CEO}, for Q1 and Q2. A benefit with higher input voltage is that the output voltage can be designed equally much higher as the increase of the minimum limit of input voltage. A drawback with higher input voltage is that the power dissipation in the converter itself can increase, especially in R5, R4 and Q2, due to increased voltage drop across them.

When significantly lower output current capability is required than provided by Q1=2N2907A, then Q1 can be by-passed. Q1, R3 and R4 are removed, and a (schottky) diode is inserted from point B to C. The converter output current is driven directly from the 54ACTxxx buffer output, and the buffer needs to have opposite signal polarity. To not overstress the buffer, R5 needs to have roughly an order of magnitude higher value, which will limit the output current capability to an order of magnitude lower than with Q1=2N2907A.

Support for PWM pulse skipping may be required in some application cases. Such a case is when the minimum duty cycle, due to necessary LEB time, is too long for the minimum load current in the application. This will cause a runaway overvoltage on the converter output, unless PWM pulses are

skipped, i.e., not turned on at all when the ACOMP indicates high already at the start of the PWM cycle. A straightforward way to implement pulse skipping is to generate the PWM control signal in an APWM_G block instead of using the APWM_A Out signal, and the APWM_G Out signal becomes the new PWM control signal. The APWM_A State signal and ACOMP signal should be combined, where a simple logic definition for APWM_G Out is: StateA *and not*(ACOMP).

A more advanced configuration, which ensures that the APWM_G Out signal cannot toggle multiple times during the LEB time, is to use the set/reset flip-flop (SR-FF) configuration in APWM_G, see the SR-FF example in section 53.5. Here, the set-condition should be: StateA *and not*(ACOMP). The reset-condition should be: StateA. These conditions ensure the SR-FF to be reset-dominant on StateA low level.

It is essential, whenever pulse skipping operation is based on ACOMP, that the ACOMP input pin represents the output voltage, $V_{OUT,}$ at all times when StateA is high (actually from about 50ns before StateA goes high). Therefore, during this whole time interval, until the pulse-skipping decision is made, the ACOMP pin must not be disturbed by any PCB disturbance or on-chip activity such as ADC track or LEB activity. An LEB setting of about 100-300 ns, mode 0, can work well.

Regarding ground connections for the load and GR716B, the orders of current in this type of converter will not cause any problem with one and the same PCB ground plane used to both the load and GR716B. Note that the ACOMP internal reference in figure 135 (0.5, 1.0, 1.5, 2.0 V) has ground reference from GND (pin 75 is closest) for GPIO 51-58, and from V_{SSA_REF} (pin 33) for GPIO 37-44. This reference grounding works well when the same PCB ground plane is used to both the load and GR716B, for the moderate orders of current here. However, in application cases where the total PCB ground current to and around the load is in the order of 10A or higher, the ground voltage drop between the load and GND/V_{SSA_REF} can become too large, making the internal reference not valid to use as ACOMP trig level. In these cases, either an external difference amplifier can replace R_{FB} and R_{FBGND} and generate the 'Voltage sense' signal to the ACOMP input, or an external reference based on the local ground potential at the load can provide the trig level to the ACOMP minus input pin.

53.13.3 APWM_A application: A controller for multiple DC/DC converters

A controller for multiple low-power DC/DC converters is presented here, see figure 136. It is based on one APWM_A and four APWM_CF blocks to regulate four independent output voltages. This type of controller can be useful in application cases where the number of ACOMPs in GR716B is a limited resource, at the same time as lower load-current capacity than in section 53.13.2 can fulfill the requirements.

General note: Component types and values in schematics should be interpreted as recommendations of right order of magnitude, presented to illustrate the functionality of the circuit topologies. They do not guarantee performance in worst-case corners, which needs to be guaranteed by detailed parts stress and worst case analyses based on your exact component types and values including tolerances and all other electrical properties in your designs.

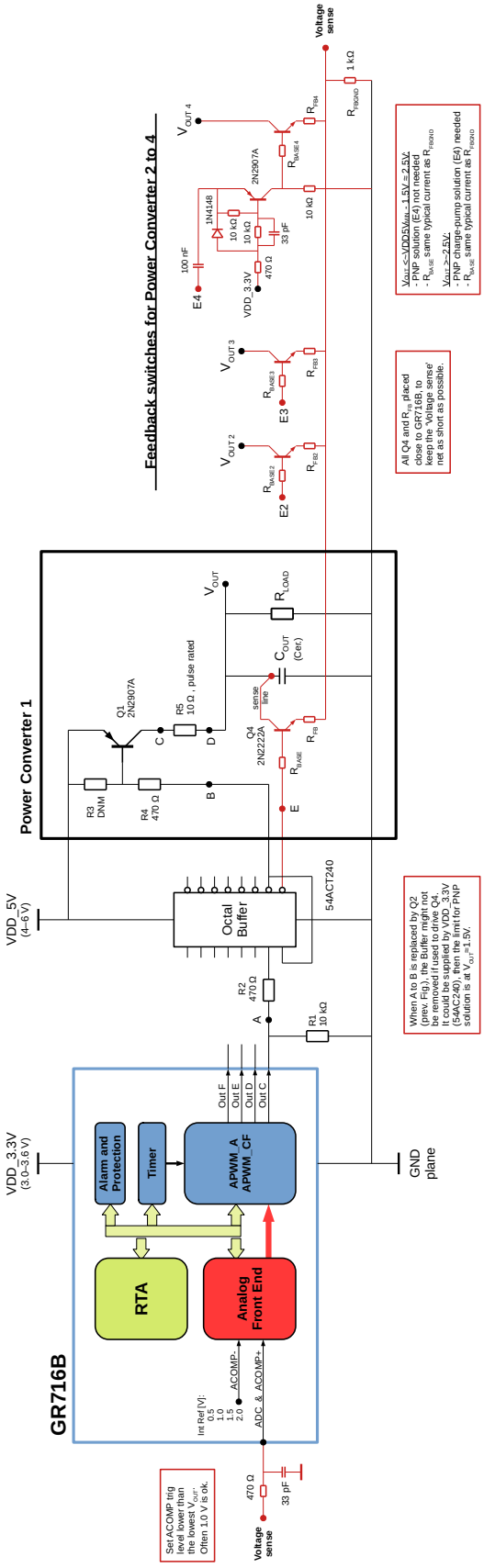


Figure 136. Simplified application schematic of four low-power DC/DC converters controlled by one APWM_A and four APWM_CF blocks. Parts stress and worst-case corner performance are to be guaranteed by your detailed circuit analyses.

Time slot configuration is set to 8 slots in the APWM_A Timer and the APWM_CF Timer, i.e., slot 0 to 7 exist. It is critical that both these Timers are configured identically, including their input Tick control from the same Main Timer, see figure 121. In this example, one output voltage uses every second slot, one uses two slots, and the last two outputs use one slot each, see figure 137. For configuration of APWM_CF, see section 53.4.

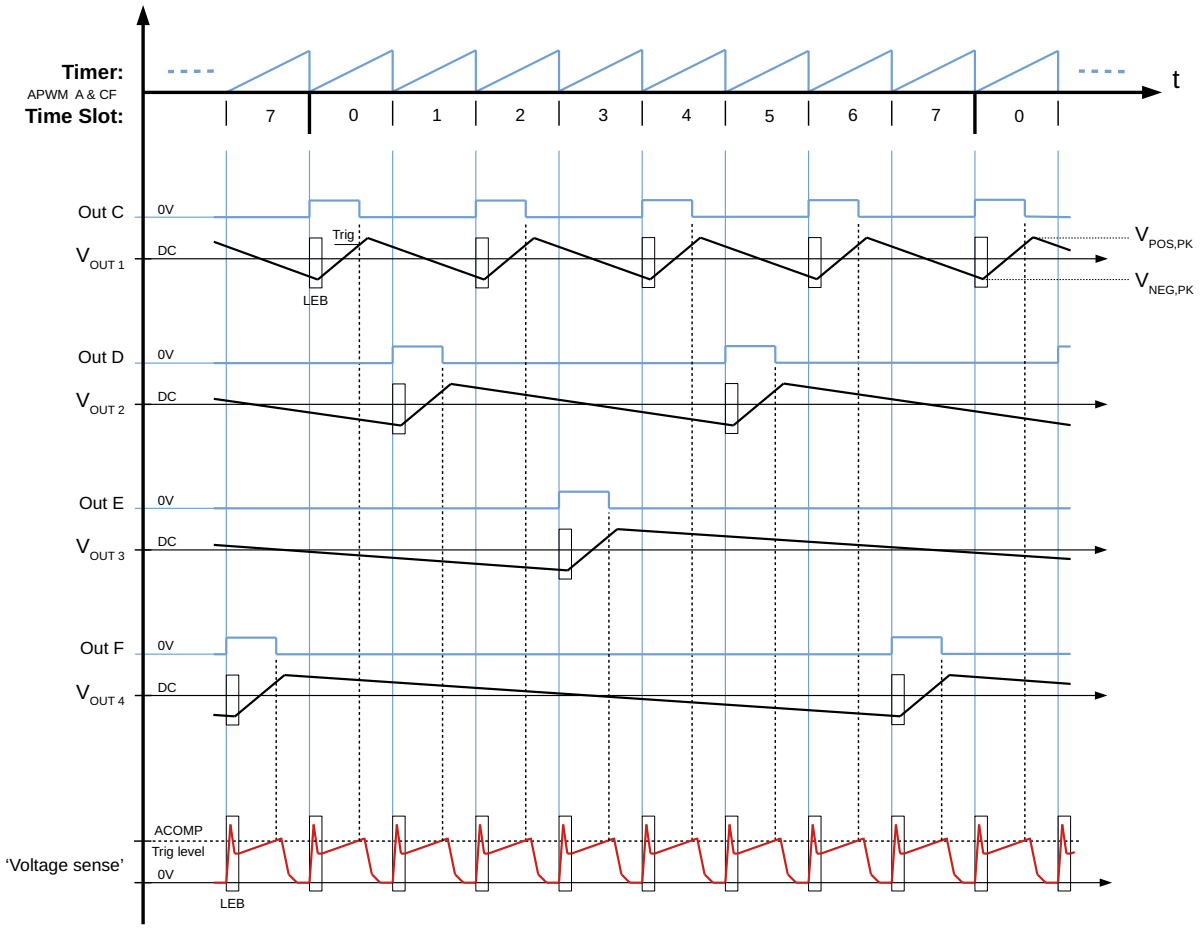


Figure 137. Timing diagram of a multi-converter controller for four output voltages.

At the start of each time slot, at APWM Timer equal to zero, the APWM_A State signal goes high. Precisely one of the APWM_CF blocks is configured to be active in each time slot, and the State signal for this APWM_CF goes high simultaneously with the APWM_A State signal. Thus, the power switch, Q1, for this APWM_CF block turns on at this time point, since the ON Delay block should not give any delay in this kind of application.

In the first time slot in figure 137, Out C turns on and connects its feedback signal to the ACOMP input, see the red signal paths in figure 136. The output voltage, V_{OUT}, for this converter starts to increase. When it reaches the ACOMP trigger level, the APWM_A State signal goes low, which is configured to end the duty cycle in APWM_CF. Thus, Out C goes low and turns off its power switch. After this slot ends, Out D turns on and connects its feedback signal to the ACOMP input. This output voltage starts to increase, and when it reaches the ACOMP trigger level the APWM_A ends the duty cycle so Out D goes low and turns off its power switch. In each of the 8 time slots, the ACOMP controls the duty cycle in the active APWM_CF block in the same way. When the last time slot (7) has been executed, the sequence restarts immediately in the first time slot (0) again.

Regarding LEB settings, the LEB time should cover the turn-on delay of the feedback switches including settling of ACOMP, and the power-switch turn-on transients. They may require an effective LEB time of about 100-300 ns, e.g., LEB time configuration of 100ns, mode 1, and up to 200ns for

accurate RC settling. However, it depends strongly on the detailed design of the switches. Any turn-off delay in the feedback switch in the previous time slot should be finished and settled in off state before that time slot ends, or at the latest it could continue partly into the LEB time of the next slot. Therefore, the duty cycle in each slot must be unconditionally terminated at a latest time point, preferably ensured by 'Duty Max Limit' configuration in each APWM_CF block. Otherwise, the feedback signal in the next slot would become corrupted.

The output current capability for each output voltage will be according to the single-converter case above, but linearly decreased to be proportional to the number of time slots used for each output. So if all 8 slots would be used for one output, the capability is equal to the single-converter case, but the multiple APWM_CF concept would obviously be pointless. If half of the slots are used for one output, like for V_{OUT1} in figure 137, the current capability becomes 1/2. If 1/8 of the slots is used, like for V_{OUT3} and V_{OUT4}, the current capability becomes 1/8. Regarding the converter internal power dissipations, the average dissipations from cooling point of view will be reduced by the same slot factors. However, the pulse/peak stresses in the components will stay the same as in the single-output case.

To get higher current capability for a converter output, the same solutions as presented for the single-converter case can be used (i.e., Q3=2N5154, Q1=2N5153 and power-switch blocks put in parallel). Here too, the average dissipations are reduced by the same slot factors, but the component pulse/peak stresses are not reduced.

To handle higher input voltage, the same solution as in figure 135 can be used, which is based on Q2 and R6 instead of the Buffer. However, also the feedback-switch control signal (connection point E), needs to be generated with equally higher voltage swing, which should be between ground and the new higher input voltage, VDD_IN>6V. This control signal can be generated by implementing a copy of the power-switch block, point B to C, i.e., R4, R3 and Q1. And a pulldown resistor to ground needs to be added in point C. Point B of the new block is connected to point B in the existing circuit. Thus, the same Q2 collector drives point B of both the existing and new block. The new point C will drive the feedback-switch control signal, i.e., point E.

Let these new components be called Q1', R3', R4' and R_{PD}. The values will primarily affect the switching time, which may require a certain LEB time to be configured in APWM_A, which in turn will be a limiting factor for minimum PWM period time. In general, keep in mind that a longer PWM period time will require larger output capacitor, C_{OUT}, in a converter. A reasonable target could be to keep LEB within about 300 ns (including settling of ACOMP pin), where the following values may be the right order of magnitude: Q1'=2N2907A, R3'=1kΩ and R4'=~$(VDD\_IN_{min}-4V)/1.5mA$, R_{PD}=~$VDD\_IN_{typ}/3mA$.

PWM pulse skipping is not as easy to support as in the single-output APWM_A case, because the ACOMP is not connected to one and the same output voltage all the time. It can be viewed as it is 'disturbed' when connected to the other output voltages during all the other time slots, and can only be trusted in the dedicated time slot. Based on this time-slot limitation, one straightforward way to introduce pulse skipping can be to let the feedback-switch turn on as fast as possible when the APWM_CF Out signal goes high, whereas an intentional delay is introduced on PCB to the power switch. After the feedback switch and ACOMP have settled accurately, but before the power switch turns on, the LEB in APWM_A should end. A realistic LEB configuration may be in the range 100-300 ns, mode 0, but it depends on the feedback-switch design.

The intentional power-switch delay needs to be longer than the LEB time, to ensure that the ACOMP can end the duty cycle before the power-switch transistor turns on, in the cases the PWM pulse should be skipped. If the power-switch delay in worst-case corner would become slightly to short, turning on slightly before LEB ends, the consequence is that the pulse skipping function is not ideal, i.e., a minimum load current will still be required, but can be significantly lower than without any intentional power-switch delay at all.

A drawback with this simple delay solution is that there is no effective LEB function during the turn-on transients from the power switch, since this switch is intentionally delayed till after the LEB time

has ended. Therefore, it is especially important to handle the whole sense signal path, all the way from C_{OUT} pin (long layout wire), through the feedback switch (placed close to GR716B), and through the low-pass RC link, with great care in the PCB layout design. To get slower and lower turn-on transients, it might also be beneficial to introduce a low-pass capacitor across base-emitter of the power switch, Q1. This capacitor could also constitute the intentional delay of the power switch, provided that the delay accuracy will be good enough.

In case proper LEB function is desired throughout the power-switch transients, the power-switch control can be implemented by a set/reset flip-flop (SR-FF) configured in an APWM_G block, see the SR-FF example in section 53.5. The reset-condition should be the State signal from ‘duty-APWM_CF’, defined here to be the APWM_CF block controlled by the APWM_A according to above. The SR-FF should be configured to be reset-dominant, i.e., this reset-condition should be configured into both APWM_G words. The set-condition should be ‘set-APWM_CF’ *and not*(ACOMP), where this new APWM_CF block shall turn on after accurate ACOMP settling (after 100-300 ns), and stay high a couple of system clock cycles. In case more than one of the converters need pulse skipping, the State signal from ‘set-APWM_CF’ can be re-used to all those APWM_G blocks, and should generate such a pulse in all time slots.

If ACOMP is high during the ‘set-APWM_CF’ system clock cycles, the output voltage is above the regulation level already at the start of the PWM cycle, and this PWM cycle will be skipped (the SR-FF will not be set). If ACOMP is low during these cycles, the PWM duty cycle is started and the power switch turns on. It stays on until ACOMP reaches its trig level, or until the duty cycle reaches the configured ‘Duty Max Limit’ in ‘duty-APWM_CF’. After that, the ‘duty-APWM_CF’ stays low (the feedback switch stays off), and thereby the SR-FF stays low (the power switch stays off), until the next active time slot.

It is essential that the low-pass RC link and ACOMP has settled accurately, for at least 4 RC time constants after the feedback switch is turned on, before ‘set-APWM_CF’ turns on (100-300 ns). Otherwise, ACOMP can make the wrong decision for pulse skipping. The LEB setting in APWM_A should be long enough to cover the feedback-switch settling (maybe 100-300 ns) plus the following power-switch settling (maybe another 100-300 ns), which in this example gives an LEB configuration in the range 200-600 ns, mode 0. However, this configuration will be strongly dependent on all the switch designs.

The benefits with this SR-FF solution are accurate delay control, and LEB function provided throughout the whole feedback-switch and power-switch turn-on time, which maintains a robust controller functionality also in strongly disturbed circuit board environments. The drawback is the additional APWM_G block and GPIO pin per converter.

Regarding ground connections for these loads in relation to the GR716B ground connection, the same comments are valid as for the single-converter case above. One and the same PCB ground plane can be used to the loads and GR716B, for the orders of current in these converters. But if the total PCB ground currents are in the order of 10A or higher, special care needs to be taken for design of the 'Voltage sense' signal. When this design is based on external difference amplifier(s), it can be one op-amp per output voltage, generating each signal going to the Q4 collector; or, it can be one common op-amp, generating the 'Voltage sense' signal. A drawback with one common op-amp is that it can only sense one common ground point for all the converters. Moreover, it needs to be fast (slewrate plus exponential settling within a couple of hundred nanoseconds) to provide accurate settling before the LEB time ends. It might actually be likely that the design with one op-amp per output voltage, in addition to making the design work more straightforward with easy individual ground points and no need for critical settling-time verifications, also will reduce op-amp cost, since these can be low-performance general op-amps.

53.13.4 APWM_CF application: DC/DC converter example (PSFB)

A phase-shifted full-bridge DC/DC converter (PSFB) is an application where APWM_CF blocks are suitable to use, see figure 138. This converter also uses APWM_AB and APWM_G, see section 53.5.

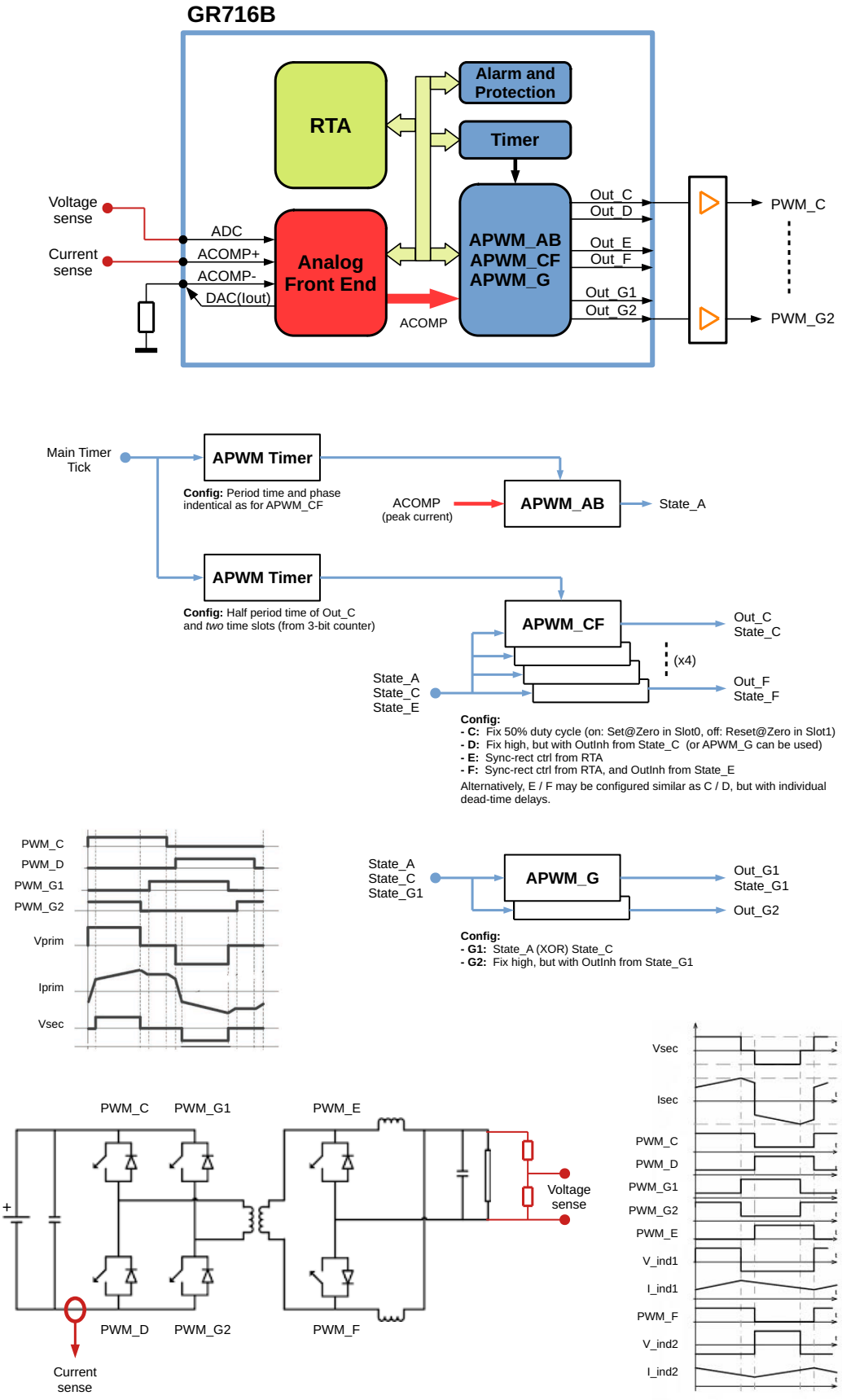


Figure 138. Configuration example for a phase-shifted full-bridge DC/DC converter (PSFB). The left time diagram shows primary-side signals with dead-time delays. The right time diagram shows secondary-side signals with synchronous rectification.

Peak-current control is performed by APWM_AB, where the current can be measured by one sense resistor in the primary ground line. The current-sense voltage goes into the on-chip analog comparator that provides the ACOMP signal to APWM_AB. The current-sense voltage can also be A/D converted and made available to RTA for monitoring purposes.

Each switch pair in the full-bridge is a half-bridge, where it is critical to apply well-controlled dead time to avoid destructive switching transients. This is first described for APWM_AB, see section 53.2. In general, the view points in that section, how to switch voltages and currents in a half-bridge, are applicable also for PSFB half-bridges. But specifically for PSFB converters, it can be beneficial to apply zero-voltage switching by dynamic control of dead times. This is straightforward to implement in RTA software since it has direct access to the ON Delay settings.

In a PSFB converter, the switching in each primary-side half-bridge is always 50% duty cycle in both half-bridges. A converter duty cycle starts when PWM_C turns on, and when PWM_C turns off; that is, there are two cycles of peak-current control per PWM_C/D period. Control of the converter duty cycle is done by phase shifting (delaying) the PWM_G1/G2 switch time points in relation to the PWM_C/D switch time points. This is controlled by APWM_AB peak-current control twice per PWM_C/D period.

When this delay is zero (PWM_G1/G2 is in-phase with PWM_C/D), the converter duty cycle is zero, and no power is transferred to the converter output. When this delay is half a PWM_C/D period (PWM_G1/G2 is 180° delayed to PWM_C/D), the converter duty cycle is maximum, and maximum power is transferred to the converter output.

The desired converter duty cycle, to maintain the desired converter output voltage, is controlled by RTA software together with the APWM_AB peak-current control. Each APWM_AB cycle, the RTA reads the output voltage, executes the control law, and writes a new reference level to the DAC for peak-current control in APWM_AB. This control loop will regulate the output voltage to a predefined DC level set in the RTA software. Further details how to control a PSFB converter with synchronous rectification are referred to textbook literature.

53.13.5 APWM_CF application: Motor control example

A standard 3-phase brushless DC motor (BLDC) can be PWM controlled by 3 APWM_CF blocks combined with 3 APWM_G blocks, see figure 139. There are several schemes for control and commutation of the motor windings such as trapezoidal (often for BLDC), sinusoidal (often for PMSM), and field-oriented control (FOC). Any such scheme is supported by APWM_CF running in standalone mode, since Timer and Duty control can be updated PWM cycle-by-cycle from RTA.

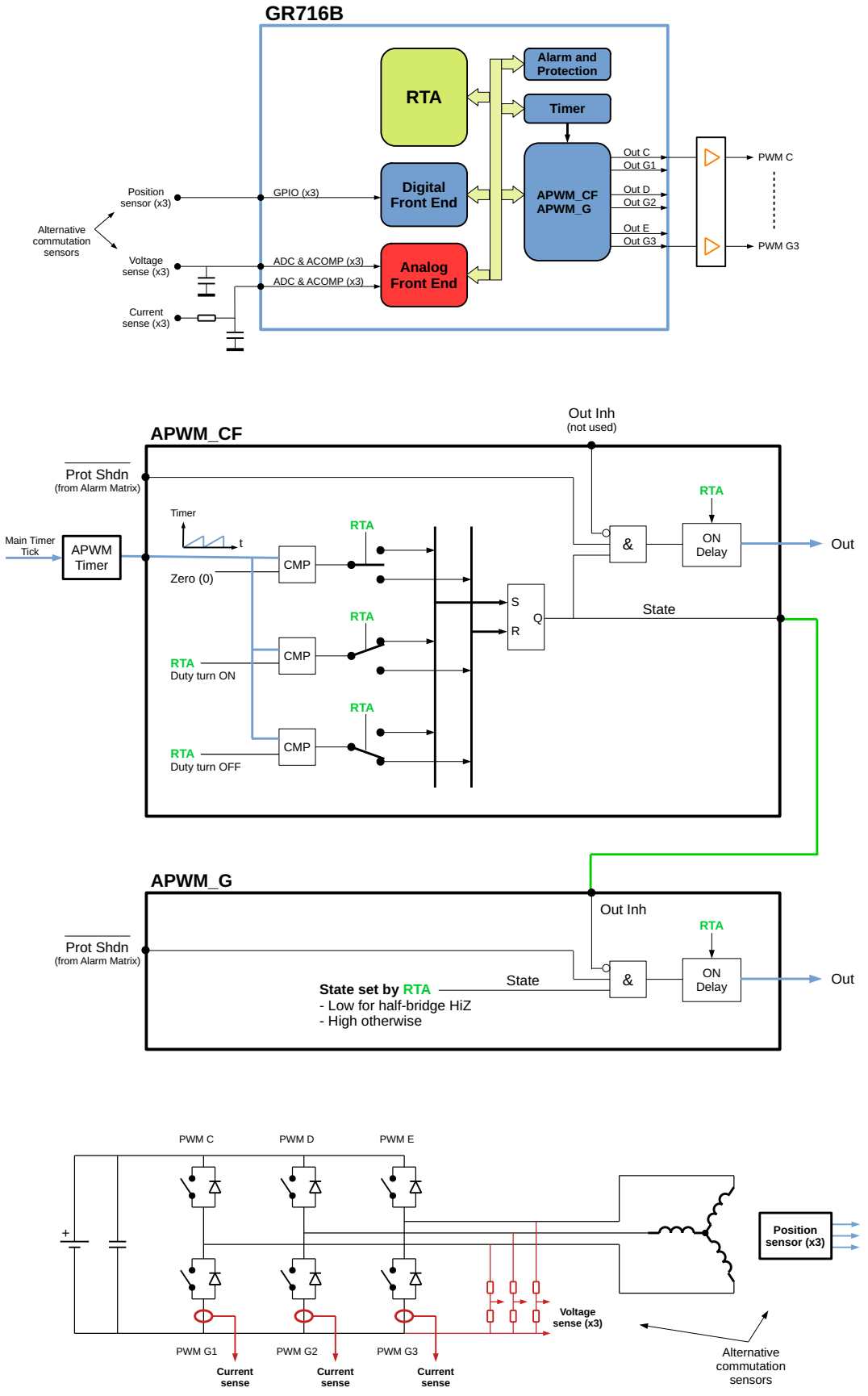


Figure 139. 3-phase BLDC motor application.

The top switch in each half-bridge is connected to an APWM_CF block, controlling the PWM function and the commutation timing, see switches PWM C/D/E in figure 139. Each bottom switch is controlled by an APWM_G block (or another APWM_CF), and provides interlock and dead-time against its top switch. If desired, these interlock and (fixed) dead-time functions are straightforward to implement in hardware on PCB instead, making these APWM_G blocks free for other usage.

Trapezoidal commutation is a basic and commonly used commutation scheme for 3-phase BLDC motors, see the example in figure 140. The motor windings are energized in a 6-step pattern every 60° electrical degree so that one phase is on (high), another phase is off (low), and the last phase remains unconnected (high impedance), which produces a 120° trapezoidal-shaped current waveform. The duty cycle of the PWM signal sets the motor winding current, and is controlled by the RTA based on winding currents measured by ADC. This 6-step pattern makes the motor rotate in one direction, and exchanging the patterns for two of the windings will reverse the rotation direction.

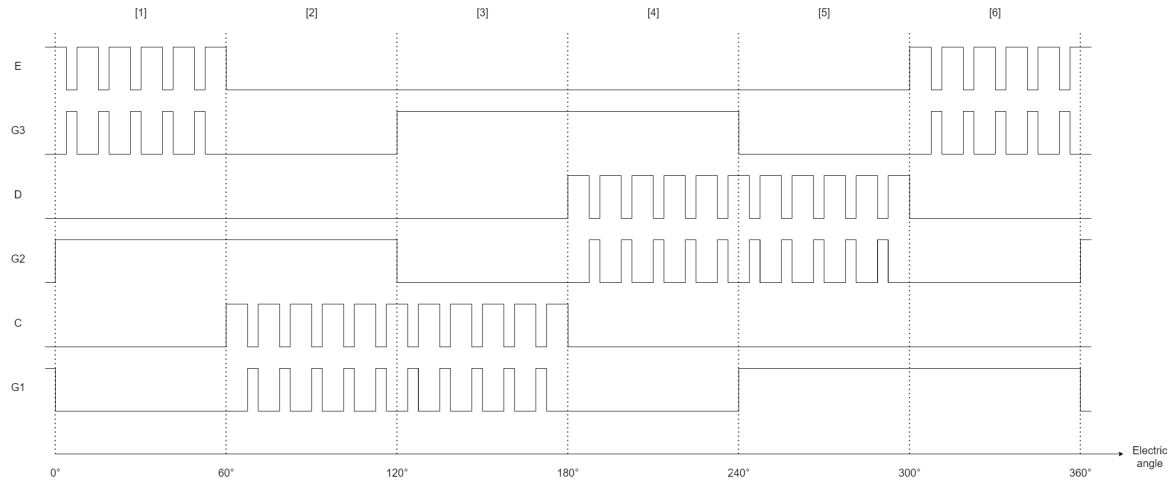


Figure 140. Bridge drive scheme for PWM controlled 3-phase BLDC motor with trapezoidal commutation.

Commutation time points depend on the motor position and can be detected by sensors, e.g., halleffect sensors, encoders, resolvers, or by voltage detection on motor windings by ACOMP level trigging or ADC measurement. A wide variety of different sensor and encoder signals can be received by GPIOs, including accurate timestamp information if needed.

Winding current measurements can be done across a sense resistor to ground during the low time in each half-bridge PWM cycle. A prerequisite is that the duty cycle does not go all the way up to 100% (the low-side switch must be on a couple of percent each PWM cycle). The accurate ADC sampling control in GR716B, provided for DC/DC converter current measurements, hence, is useful also here for motor current measurements. Furthermore, if desired, overcurrent detection can be configured as an ACOMP voltage trig level on the same sense resistor.

54 Digital Modulator for Analog Precision DAC Outputs (APWMDAC)

54.1 Overview

The GR716B microcontroller contains four Digital Modulator for Analog Precision DAC Output units (APWMDAC). An APWMDAC unit provides a digital modulated output signal which can be low-pass (LP) filtered on PCB to create a precision analog voltage with 24-bit resolution. Clocking is configurable from GPIO input or self-generated by dividing the system clock. The modulator input value can be updated continuously from the main processor, RTAs and DMA.

The analog precision after the LP filter has higher resolution than the 12-bit on-chip DAC in section 15. But the analog bandwidth is typically lower, e.g., in the range 0.1 to 100 kHz, depending on how the PCB LP-filter is optimized for analog bandwidth, ripple, filter complexity, etc.

Each APWMDAC unit controls its own external pins as described in section 2.6, and has a unique AMBA address described in section 2.10. They are controlled through register interfaces located on an APB bus in the address range from 0x81009000 to 0x8100C0FF. Figure 141 shows a partial bus diagram of the APWMDAC units within the GR716B including supporting peripherals.

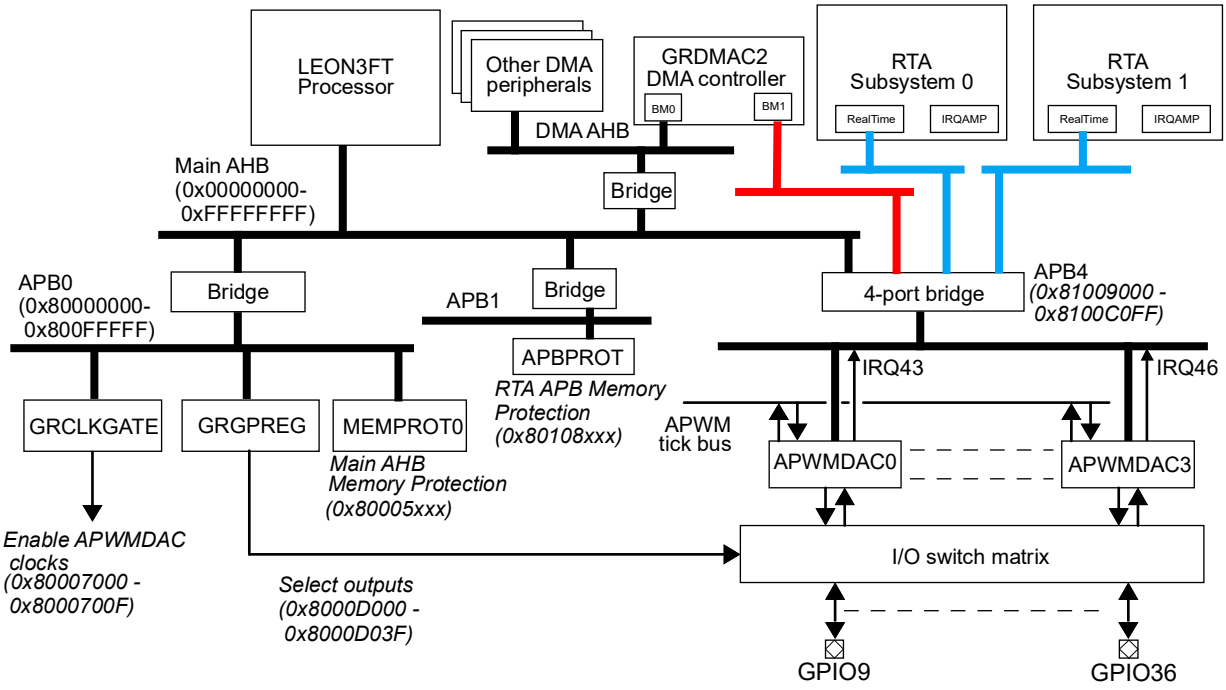


Figure 141. GR716B APWMDAC bus and pin connections.

The I/O switch matrix described in sections 2.5 and 7.1 is used to select GPIO pins for use as APWM-DAC outputs. However, note that the optional APWMDAC clock inputs from GPIO bypass the I/O switch matrix, see section 54.2.2.

The secondary clock gating unit **GRCLKGATE** described in section 27 is used to enable/disable the APWMDAC units. The unit **GRCLKGATE** can also be used to perform reset of the APWMDAC units. Software must enable clock and release reset before configuration and operation can start. Additionally, software must enable the clock and release reset for the APWM subsystem in the primary clock gating unit before the APWMDAC registers can be accessed.

The APWMDAC configuration and status registers are described in section 54.3. The system can be configured to protect and restrict access to APWMDAC units in the **MEMPROT** and **APBPROT** units. See section 47 for more information.

54.2 Detailed description

APWMDAC provides a digital modulated output signal, to be low-pass (LP) filtered on PCB. It can work as a single second-order modulator or as two independent first-order delta sigma modulators. It can run either from an external clock input on a GPIO pin or from an internal clock generated by division of the system clock. The modulator input value can be updated continuously from RTAs or the main LEON3FT. A block diagram of the APWMDAC is shown in Figure 142.

The input digital value to the modulators are 24-bit unsigned integers. The output of the first-order modulator is binary and its average is linear to the digital input value with a DAC full scale range of 0 to 1. The second-order modulator has three binary outputs (A, B, and C) and the average of the sum of the outputs (A+B+C)/3 is linear to the digital input with a DAC full scale range from 1/3 to 2/3. When the modulator outputs are low-pass filtered on PCB, the filter output will be close to the ideal average value. The filter output will differ from the ideal by a ripple that depends on the filter bandwidth relative to the modulator sample clock frequency. See section 54.2.3 for simulated output spectra of the modulators before low-pass filtering and section 54.2.4 for application examples of low-pass filters of varying complexity and performance. See section 54.2.1 for I/O switch matrix settings.

When the modulator generates its own clock, the generated clock (PDAC_CLK) is available on GPIO output if desired. The frequency is configured by an integer divisor on the system clock, with a minimum divisor such that the modulator frequency is maximum 25 MHz. The modulator outputs are updated on negative output-clock edge, so if flip-flops are implemented on PCB, they should be clocked on the positive edge. The generated clock can be synchronized to TIMER32 ticks, see section 54.2.2.

When the modulator runs on external clock from PCB, the positive edge of this input clock will act as modulator clock and update the modulator outputs. The positive edge should also be used to clock flip-flops on PCB, to provide maximum setup time to these flip-flops. A typical propagation delay in GR716B, from positive input-clock edge on GPIO pin to modulator outputs on GPIO pin, is 20 ns+1.6/f_{intsys}, which is 36 ns at 100 MHz system clock (add up to 5 ns for PCB load on GPIO output of up to 25 pF). The received clock can be used to generate an APWM tick and a system interrupt. See section 54.2.2. Note that the clock input bypasses the I/O switch matrix, see section 54.2.1.

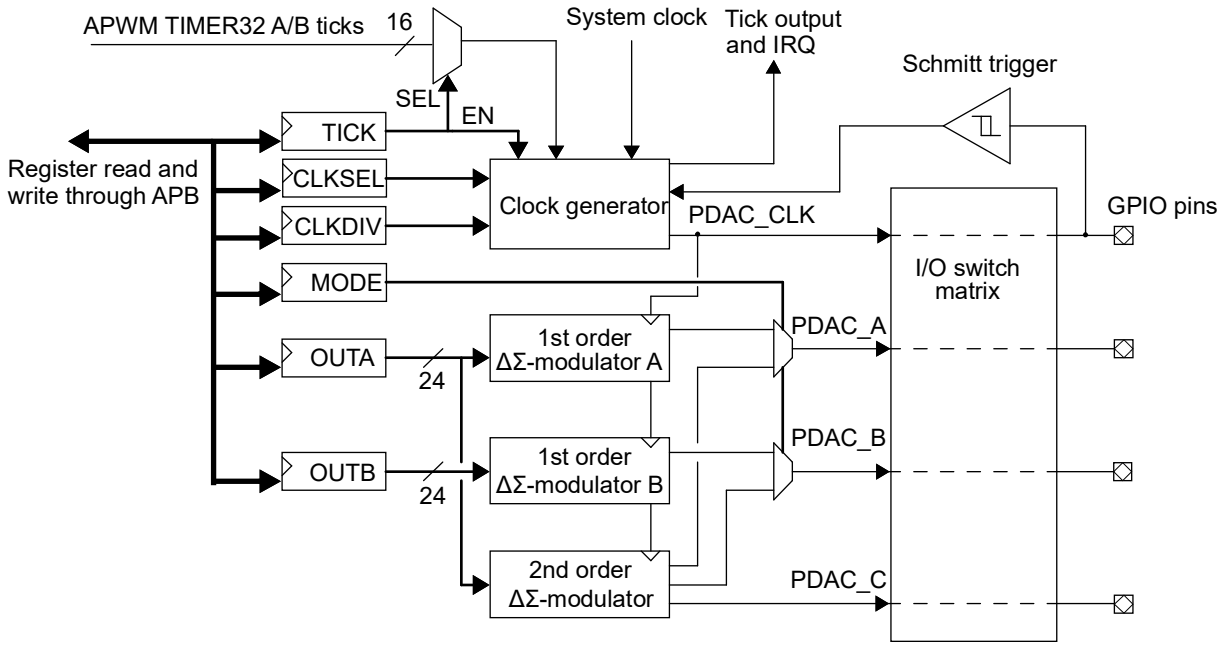


Figure 142. Block diagram of an APWMDAC unit.

54.2.1 GPIO pins and IOMUX

An APWMDAC unit can be operated with as few as one and as many as four GPIO pins depending on operating mode, clock source and LP-filter topology. The minimal configuration is a single output, e.g. in case of a first-order modulator clocked from the internal system clock without external DFFs.

PDAC_A, PDAC_B and PDAC_C are always outputs. When configured as first-order modulators (APWMDAC.MODE=1), PDAC_C is unused and the corresponding GPIO pin can be muxed for another function. PDAC_A and PDAC_B are then connected to independent first-order modulators whose inputs are the AWPMDAC.OUTA and APWMDAC.OUTB registers, respectively. If only a single output is needed by the application, then the PDAC_B (or A) output can be muxed for another function with only the PDAC_A (or B) GPIO pin muxed for APWMDAC use.

When configured as a second order modulator (APWDAC.MODE=0), PDAC_A, PDAC_B, and PDAC_C must all be muxed for APWMDAC use and summed on PCB as in the filter examples in the right column of Figure 144.

PDAC_CLK is an input if APWMDAC.CLKSEL=1 and an output if APWMDAC.CLKSEL=0. This pin is intended to be used with filter topologies that include an external DFF. See the bottom two rows of Figure 144 for filter examples of this type.

When PDAC_CLK is used as an output, the corresponding GPIO pin should be configured as PDAC_CLK in the I/O switch matrix. The PDAC_A, PDAC_B, and PDAC_C outputs are then updated on the falling edge of PDAC_CLK. Therefore, if external DFF are present on the PCB, they should sample the modulator outputs on the rising edge of PDAC_CLK.

When PDAC_CLK is used as an input, the I/O switch matrix (sections 2.5 and 7.1) must be configured such that the GPIO pin is not an output. Counterintuitively, this means that the PDAC_CLK option (mux mode 0x9) must not be used since that option would configure the pin to be an output. It is recommended to use mux mode 0x0 (GPIO) and set the pin as an input in the GRGPIO controller (section 30). The APWMDAC clock input is always taken from the Schmitt-trigger receiver of the GPIO, regardless of the setting of the SYS.CFG.SCHMITT0-1 registers. Note that although APWMDAC0 can use both GPIO9 (mux mode 0x9) and GPIO33 (mux mode 0xA) as PDAC_CLK output, only GPIO9 can be used for PDAC_CLK input (recommended mux mode 0x0).

54.2.2 Modulator sample clock

The modulator sample clock can be internally generated by dividing the system clock by an integer (APWMDAC.CLKSEL=0) or taken from a GPIO pin (APWMDAC.CLKSEL=1).

When using a clock input from a GPIO pin, the GPIO clock input is sampled by the falling edge of the internal system clock. The PDAC_A/B/C outputs are updated on the next rising edge of the system clock This will happen between 0.5 to 1.5 system clock cycles after a rising edge on the GPIO clock input plus an output delay of order 20 ns (with a strong dependence on temperature and supply voltage operating condition). The sampling jitter is not random, and aliasing effects between the system clock and the sampling clock will cause modulation of the LP-filtered output unless the LP-filter topology includes an external DFF (as in the bottom two rows of Figure 144).

When the sample clock is generated by dividing the system clock, then the sample clock period will be APWMDAC.CLKDIV+1 system clock cycles. This clock can optionally be output on a GPIO muxed for the PDAC_CLK function. The PDAC_A/B/C outputs change on falling edges of PDAC_CLK and any external DFF if present should therefore sample the modulator outputs on rising edges of PDAC_CLK. When CLKDIV+1 is an even number, the duty cycle of PDAC_CLK is 50%. When CLKDIV+1 is odd, either the high or low time will be one system clock cycle longer than the other polarity. It is possible to force the low period to be longer than the high period by first setting CLKDIV=0 and then setting CLKDIV to the intended value. This maximizes the setup time of the external DFF.

The clock divider for PDAC_CLK can optionally be synchronized to one of the 16 TIMER32 A/B tick outputs (section 53.9). A tick source to synchronize against is selected with APWM-

DAC.TICK.SEL. If APWMDAC.TICK.EN=1, then the PDAC_CLK generator will restart if the selected tick occurs.

Optionally a tick output and bus interrupt can be generated on rising edges of PDAC_CLK. For example, this allows an RTA to update the modulator input value once per sample clock cycle. See section 54.3.1. The tick output for APWMDAC*n* is connected to TickLines0(*n*) (see section 52.5) and to system interrupt line 43+*n*.

54.2.3 Modulator output spectra

APWMDAC output spectra are presented in figure 143, where it can be noted that the first-order modulator has a slope of about 20 dB/dec (at medium high frequencies), and the second-order modulator has about 40 dB/dec. Therefore, the first-order modulator requires a second-order LP-filter on PCB to suppress the high-frequency noise, whereas the second-order modulator requires a third-order filter.

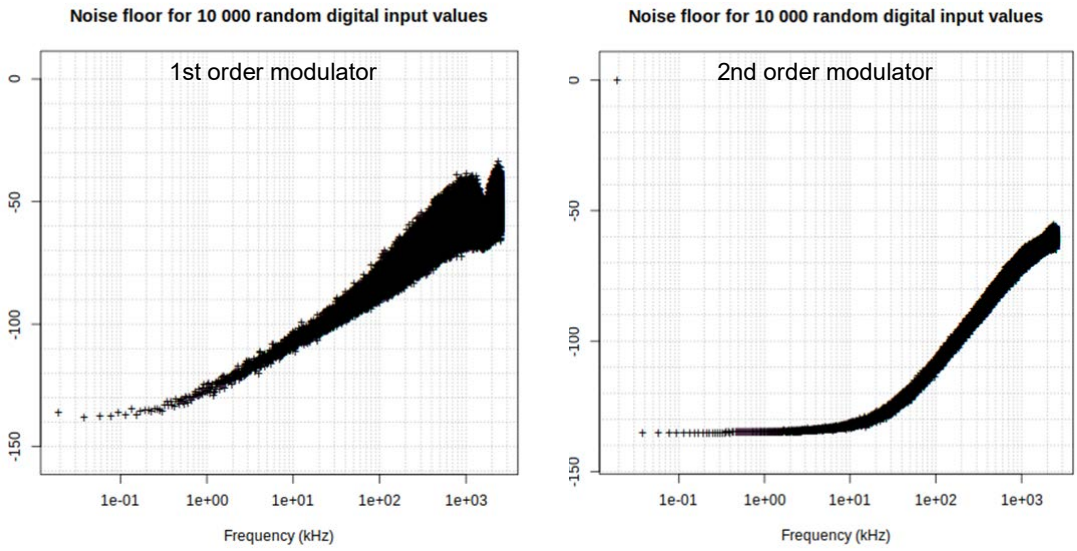


Figure 143. FFT simulation on the modulator digital output. Average over all input values is plotted per FFT frequency, and modulator clock frequency is 5 MHz. First-order modulator to the left and second-order modulator to the right.

54.2.4 Application Examples of PCB Filters

In the simplest form, the LP-filter on PCB can be two cascaded RC-links directly connected to the modulator output GPIO pin on GR716B. Then, the V_{DD_IO} supply on GR716B acts as reference/fullscale voltage for this analog output, see top left in figure 144 (“2nd-order low-pass filter”). If better accuracy is required, a CMOS buffer can be added on PCB, with VDD connected to a precision reference voltage, see mid schematic in figure 144 (“Precision reference, Output buffer”).

The GR716B on-chip implementation supports 24-bit DAC resolution, meaning that it will be the PCB analog implementation that limits the analog output performance. Hence, the higher complexity and cost that is acceptable in the PCB implementation, the better analog accuracy, higher bandwidth, lower analog ripple, etc, can be achieved.

Excellent analog accuracy can be achieved by re-clocking the modulator output through a clocked CMOS register on PCB, with VDD connected to a *precision reference voltage* and clock from a *stable oscillator* such as a crystal oscillator, e.g., XO_OUT on GR716B. And *balanced* signal generation would be recommended, fed into a *differential* op-amp-based LP-filter. Then, the CMOS register needs to generate both inverted and non-inverted outputs, see the last schematic in figure 144. Such an implementation might provide accuracy (INL) in the order of 100 µV, assuming fullscale range in the

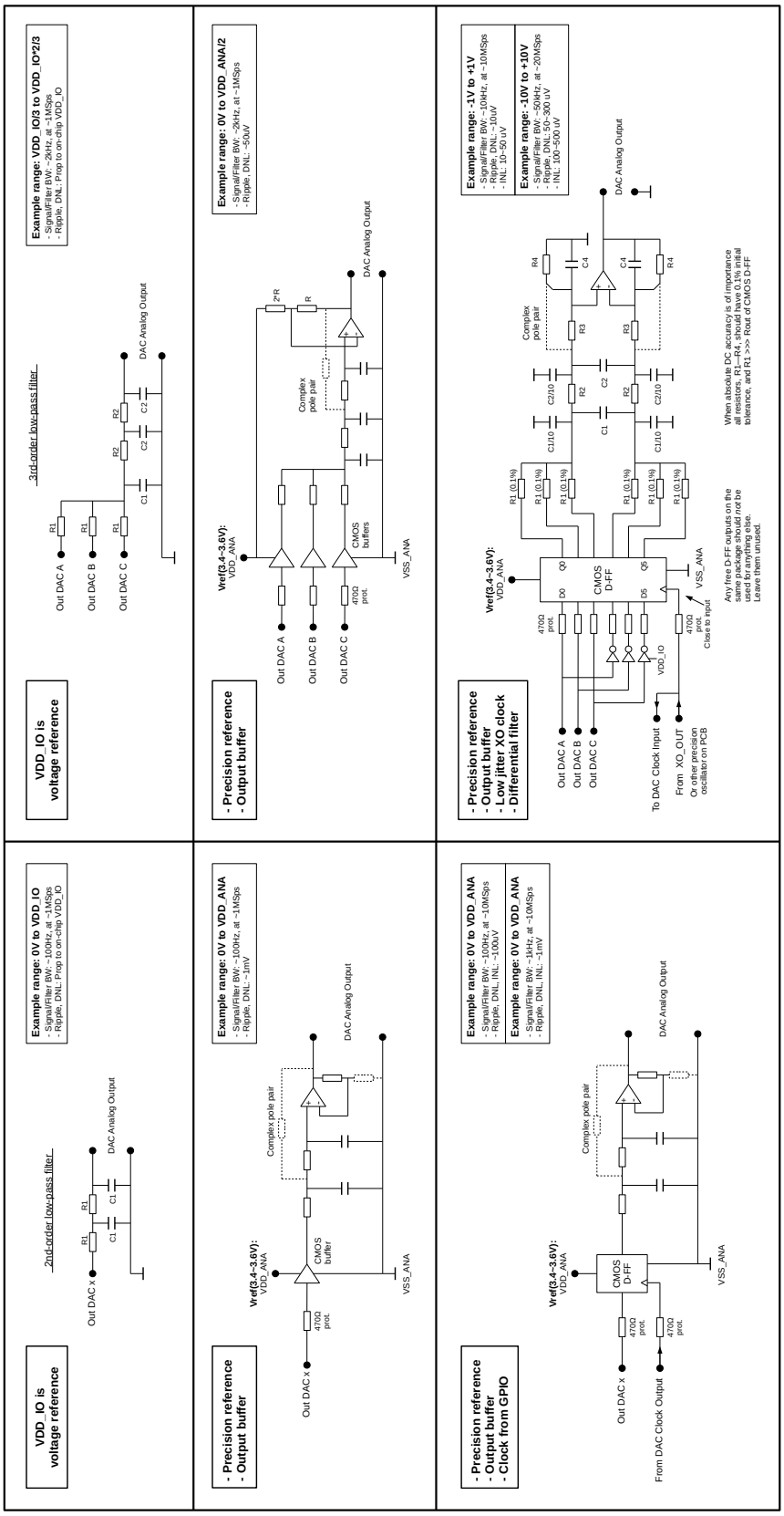


Figure 144. PCB LP-filter examples for APWMDAC. Higher PCB circuit complexity provides better analog performance. The performance numbers might be the expected order of magnitude, and are only provided here for understanding of each filter topology's properties in comparison to the other topologies.

order of a couple of volts, but it will depend on many details in the PCB implementation, e.g., how the grounds are implemented in PCB layout and on system level.

When high analog bandwidth is desired, the second-order modulator with third-order LP-filter needs to be used, together with high modulator frequency. To maintain correct pulse-width duty cycle at high modulator frequency, balanced differential signal generation on PCB is highly beneficial. In the order of 100 kHz analog bandwidth or possibly up to 1 MHz might be feasible to achieve, with analog ripple in the order of 1 mV, at modulator frequency of about 20 MHz, but it depends on many details in the PCB implementation.

To the extent that rising and falling edges of the PWM output are symmetrical and have time to settle within a modulator clock cycle, linearity is unaffected by finite slew rates. But any asymmetry in timing or waveform shape will increasingly affect linearity at high modulator frequencies. The properties of the GR716B GPIO output drivers may therefore be a limiting factor for the two filter configurations in the top row of Figure 144. For the two middle rows, the slew rate of the GR716B outputs also factors in, if the input threshold of the CMOS buffer is different from VDD_IO/2. Any asymmetry in propagation delay between falling and rising edge may also affect linearity in this case. For the configurations in the bottom two rows, the linearity is independent of GR716B GPIO output driver properties. The magnitude of all these effects can be reduced by decreasing the modulator frequency.

When high linearity is desired in second-order modulator mode, the linearity can be affected by mismatch between the three output averaging resistors (or six resistors in the last schematic). Resistors with 1% initial tolerance, which commonly have 2% to 3% total tolerance over full temperature, might be advisable for linear dynamic range up to the order of 80 dB (ENOB=~13 bits), or possibly a little bit higher. For linear dynamic range of about 100 dB (ENOB=~16 bits) or better, it is advisable to use resistors with 0.1% initial tolerance (0.2% to 0.3% total tolerance). Moreover, the order of impedance level for these resistors should be selected to about 1000 times higher than the output impedance of the CMOS buffers.

54.3 Registers

The APWMDAC unit is programmed through registers mapped into APB address space.

Table 904. APWMDAC registers

AMBA address	Register	Acronym
0x81009000	APWMDAC 0 Interrupt status register	APWMDAC0.STAT
0x81009004	APWMDAC 0 Interrupt flag register	APWMDAC0.IRQ
0x81009008	APWMDAC 0 Interrupt mask register	APWMDAC0.IMASK
0x8100900C	APWMDAC 0 Interrupt status register	APWMDAC0.IEDGE
0x81009010	APWMDAC 0 Tick input configuration register	APWMDAC0.TICK
0x81009014	APWMDAC 0 Mode register	APWMDAC0.MODE
0x81009018	APWMDAC 0 Clock select register	APWMDAC0.CLKSEL
0x8100901c	APWMDAC 0 Clock division register	APWMDAC0.CLKDIV
0x81009020	APWMDAC 0 Output register A (modulator input)	APWMDAC0.OUTA
0x81009024	APWMDAC 0 Output register B (modulator input)	APWMDAC0.OUTB
0x81009028 - 0x810090FF	This address range contains proprietary and reserved read-only registers. They may be read without side effect, but the values should be ignored.	N/A
0x8100A000 - 0x8100A0FF 1)	APWMDAC 1 registers	APWMDAC1.*
0x8100B000 - 0x8100B0FF 1)	APWMDAC 2 registers	APWMDAC2.*
0x8100C000 - 0x8100C0FF 1)	APWMDAC 3 registers	APWMDAC3.*

Note 1: APWMDAC 1, 2, and 3 have identical register interfaces as APWMDAC 0, accessed at different base addresses.

54.3.1 APWMDAC interrupt registers

*Table 905.*0x00 - APWMDAC.STAT - APWMDAC Interrupt status register

31	4	2	1	0
RESERVED			PAR	TICKO
0			0	0
r			rw*	rw*

31: 2 Reserved

1 Parity error flag (PAR) - Indicates a parity error is detected in the configuration registers for this APWMDAC.

0 Tick output (TICKO) - High for two system clock cycles whenever a rising edge of PDAC_CLK has occurred. (TBC) If this is enabled as an interrupt source, the APWMDAC APWM tick output will assert whenever the interrupt condition occurs.

* For test purposes, bits 1:0 in this register are writable. Writing 1 can be done to inject an interrupt.

*Table 906.*0x00 - APWMDAC.IRQ - APWMDAC Interrupt flag register

31	4	2	1	0
RESERVED			PAR	TICKO
0			0	0
r			wc	wc

Each bit in this register is an interrupt flag for the corresponding bit in the APWMDAC.STAT register. The bit will be set to 1 if an unmasked interrupt occurs.

*Table 907.*0x00 - APWMDAC.IMASK - APWMDAC Interrupt mask register

31	4	2	1	0
RESERVED			PAR	TICKO
0			0	0
r			rw	rw

Each bit in this register is the mask for the corresponding bit in APWMDAC.STAT. Set to 0 to disable the interrupt, 1 to enable.

*Table 908.*0x00 - APWMDAC.IEDGE - APWMDAC Interrupt edge register

31	4	2	1	0
RESERVED			PAR	TICKO
0			1	1
r			rw	rw

Each bit in this register configures the edge on which interrupts will be generated for the corresponding bit in the APWMDAC.STAT register. 1 for rising edge, 0 for falling edge.

54.3.2 APWMDAC Tick input selection register

*Table 909.*0x10 - APWMDAC.TICK - APWMDAC Tick input configuration register

31 5	4	3 0
RESERVED	EN	SEL
0	0	0x0
r	rw	rw

31: 5 Reserved

4 Tick MUX enable (EN) - When EN=1, the PDAC_CLK generator will be reset whenever the TIMER32 tick selected by SEL occurs, provided that APWMDAC.CLKSEL=0 (internally generated clock). If EN=0, then all TIMER32 ticks are ignored by the APWMDAC.

3: 0 Tick MUX select (SEL) - 16 to 1 MUX select signal. The 16 inputs are the TIMER32 A and B tick outputs.

TIMER32 A tick output *n* is selected by SEL=*n*.

TIMER32 B tick output *n* is selected by SEL=8+*n*.

54.3.3 APWMDAC mode register

*Table 910.*0x14 - APWMDAC.MODE - APWMDAC Mode register

31 1	0
RESERVED	MODE
0	0
r	rw

31: 1 Reserved

0 PWMDAC mode (MODE) - Selects between second order modulator (0) and two first order modulators (1).

When MODE=0, the 24-bit output code is controlled by PWMDAC.OUTA and all three outputs PDAC_A, PDAC_B, and PDAC_C should be resistor summed on PCB as in the with a 3rd order low pass filter as in the examples in the right hand side of Figure 141.

When MODE=1, the PWMDAC runs two independent first-order modulators, each with its own output on signals PDAC_A and PDAC_B. The PDAC_C output is unused in this mode. The 24-bit output code for the A output is set in the AWPMDAC.OUTA register while the 24-bit output code for the B output is set by DAC.OUTB. When used, outputs A and B should each be connected to a 2nd order low-pass filter on PCB as in the left half of Figure 141. If the B output is unused, then the corresponding GPIO pin can be used by other functions in the IO switch matrix.

54.3.4 APWMDAC Clock select register

*Table 911.*0x18 - APWMDAC.CLKSEL - APWMDAC Clock select register

31 1	0
RESERVED	CLKSEL
0	0
r	rw

31: 1 Reserved.

0 Modulator clock selection (CLKSEL) - Selects between external clock input pin (1) and internally generated clock (0). When 0, the clock frequency of the modulator will be (system frequency)/(APWMDAC.CLKDIV+1) and PDAC_CLK is an output. When 1, PDAC_CLK is an input and sampled by the local system clock before being used as modulator clock.

Note that when PDAC_CLK is used as input, the corresponding GPIO must be configured as an input (recommended mux mode 0x0). See section 54.2.1 for details.

54.3.5 APWMDAC Clock Generator

*Table 912.*0x1C - APWMDAC.CLKDIV - APWMDAC Clock Period register

31 16	15 0
RESERVED	CLKDIV
0	0x0000
rw	rw

31: 16 Reserved.

15: 0 APWMDAC clock period (CLKDIV) - When APWMDAC.CLKSEL=0, the modulator clock runs at a frequency of (system frequency)/(CLKDIV+1) and can be made available on PCB through the PDAC_CLK signal. When PWMDAC.CLKSEL=1 the value in CLKDIV has no effect.

54.3.6 APWMDAC output register A

*Table 913.*0x20 - APWMDAC.OUTA - APWMDAC Output register A

31 24	23 0
Reserved	OUTA
0	0x000000
r	rw

31: 24 Reserved.

23: 0 Output code A (OUTA) - When APWMDAC.MODE=0, OUTA is the 24-bit digital input to the second order modulator. When APWMDAC.MODE=1, OUTA is instead the 24-bit input to the first-order modulator that has PDAC_A as its output.

54.3.7 APWMDAC output register B

*Table 914.*0x24 - APWMDAC.OUTB - APWMDAC Output register B

31 24	23 0
RESERVED	OUTB
0	0x000000
r	rw

31: 24 Reserved.

23: 0 Output code B (OUTB) - When APWMDAC.MODE=1, OUTB is the 24-bit digital input to the first-order modulator that has PDAC_B as its output. When APWMDAC.MODE=0, OUTB is unused.

55 FPGA Scrubber Controller

The GR716B microcontroller comprises a Field Programmable Gate Array (FPGA) supervisor and scrubber controller (GRSCRUB) unit, which accesses the target FPGA via the SelectMap (SMAP) interface. The GRSCRUB unit controls its own external pins and has a unique AMBA address described in section 2.10. The GRSCRUB control and status registers are located on the APB bus in the address range from 0x80404000 to 0x804040FF. See the GRSCRUB unit connections in the next drawing. The figure shows memory locations and functions used for GRSCRUB configuration and control.

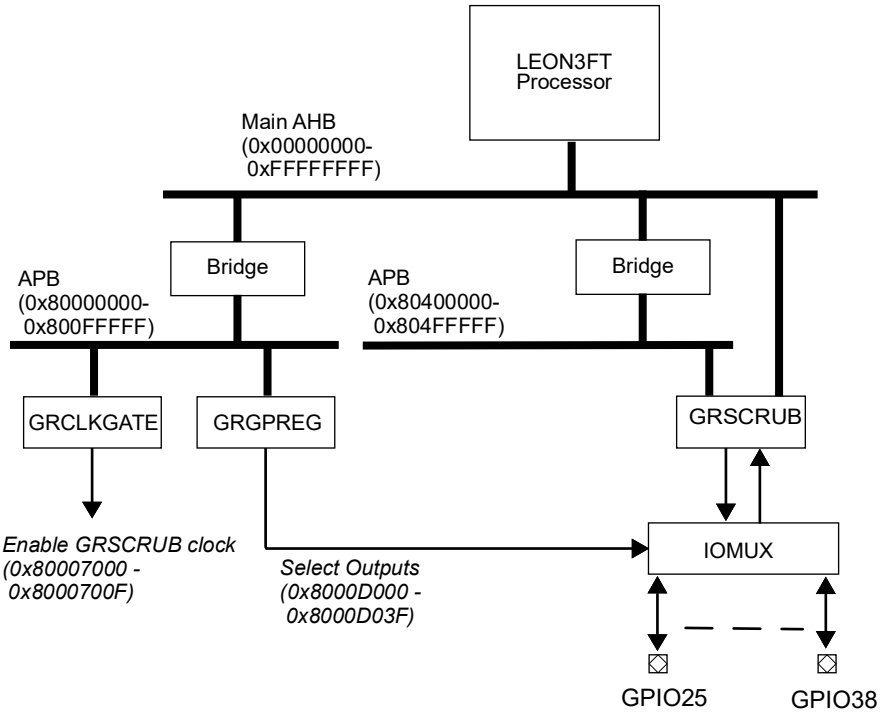


Figure 145. GR716B GRSCRUB bus and pin connection

The secondary clock gating unit **GRCLKGATE** described in chapter 28 is used to enable/disable the GRSCRUB unit. The unit **GRCLKGATE** can also be used to reset the GRSCRUB unit. Software must enable clock and release reset described in chapter 28 before GRSCRUB configuration and transmission can start.

The GRSCRUB unit works in two clock domains, the internal system clock and the SelectMap clock (SCRUB_SCLK). The SCRUB_SCLK is used for internal synchronization and external clocking of the FPGA SelectMap interface. The system clock and SCRUB_SCLK configurations are described in chapter 4.

External IO selection for the GRSCRUB unit is made in the system IO configuration register (**GRGPREG**) in the address range from 0x8000D000 to 0x8000D03F. See section 7.1 for further information.

The GRSCRUB unit can access an external SPI memory via the AHB bus connected to the SPI memory controller units (**SPIMCTRLx**). See chapter 46 for further information. Future versions of the GR716B microcontroller may allow the GRSCRUB to access a NVRAM memory. More details in chapter 23.

55.1 Overview

Devices in the space environment are vulnerable to radiation-induced particles, and SRAM-based FPGAs are particularly susceptible to Single Event Upsets (SEU) that may affect the configuration memory. Soft errors in the FPGA configuration memory (CRAM) can change the device functionality and lead to errors and malfunction in the system. The scrubbing method is essential to avoid an accumulation of upsets in the FPGA configuration memory.

The GRSCRUB is an external FPGA scrubber controller responsible for programming and monitoring the FPGA configuration memory. The GRSCRUB is compatible with the AMD/Xilinx Kintex UltraScale and Virtex-5 FPGA families. The scrubbing functionality can be set to target the entire FPGA configuration memory or just a defined memory area.

The scrubbing mitigation technique only fixes bit-flips in the FPGA configuration memory, being up to the user to apply any additional method to mask errors and re-establish the state of the system. Scrubbing does not cover soft errors affecting user memory data. All dynamic data stored in memory elements, such as shift-registers (SRL), LUT RAMs, and Block RAMs (BRAM), are not verified during scrubbing.

Figure 146 shows a simplified example of the GRSCRUB system. The GRSCRUB accesses the FPGA through the SelectMap interface. Note that the FPGA M[2:0] pins must be pre-configured to Slave SelectMap mode (i.e., M[2:0]=”110”). The GRSCRUB is connected to the SelectMap interface of the FPGA via the GR716B GPIO pins (from GPIO[25] to GPIO[38]). The GR716B allows using 8-bit data and all control pins required to program and scrub the target FPGA.

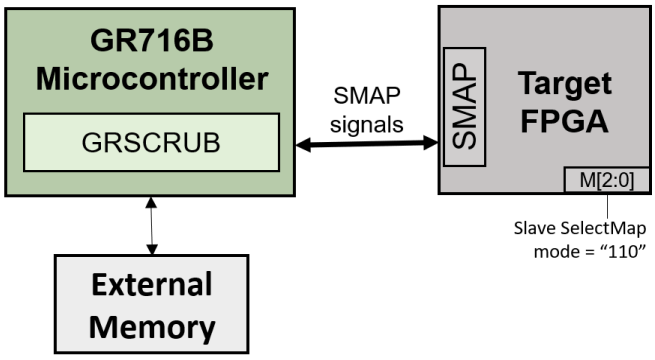


Figure 146. Simplified example of user-case setup.

An external memory stores the bitstream used to program and scrub the target FPGA. The external memory is accessed through an AMBA AHB and is referred to as Golden memory since it stores all the golden data required for the FPGA supervising functions. Besides the bitstream, the external memory should store the FPGA mask data, the FPGA frame addresses (mapping information), and the Cyclic Redundancy Check (CRC) codes (used for error detection during scrubbing). The external memory must have enough storage space and be loaded prior to the usage of GRSCRUB.

55.2 Soft error mitigation

Radiation-induced soft errors are errors provoked by radiation particles that affect the design without damaging the device permanently. SEUs, or bit-flips, in the FPGA configuration memory may lead to persistent errors in the system, changing the architectural implementation of the design. Soft errors can also affect the memory data, registers, and flip-flops, and cause errors in the system outputs. Single Events Transients (SET) are transient pulses that propagate through the combinational logic and may be captured by a memory cell. The Single Event Functional Interrupt (SEFI) occurs when a soft error affects the control logic or a state register and leads to hangs or crashes in the design.

The GRSCRUB targets soft errors affecting the FPGA configuration memory. Note that GRSCRUB does not prevent bit-flips from happening or its effects on the design. The GRSCRUB aims to maintain the configuration memory consistent by repairing the logic and correcting bit-flips, avoiding the

accumulation of faults. It restores the CRAM frames by rewriting the correct configuration frame-by-frame. The scrubbing operation does not interfere with the design execution since it targets the static layer of the FPGA configuration memory. There are two scrubbing approaches. The blind scrubbing rewrites the CRAM frames without a prior error check, which means that all frames are refreshed whether they present an error or not. On the other hand, the readback scrubbing first reads a frame, checks for errors (error detection), and the frame is refreshed only in case of bit-flip detection (error correction). The primary difference between both approaches is that readback provides the error rate per scrubbing cycle.

Memory elements that store dynamic data, such as BRAMs, distributed memory, and flip-flops, are not protected by the CRAM scrubbing. The scrubbing technique is able to verify only the static data in the configuration memory. The configuration memory stores the architectural logic information of a design and defines the function, behavior, and connectivity of each block. The GRSCRUB can be combined with additional fault mitigation methods at the user level to increase the overall system reliability.

Soft errors affecting the design dynamic elements can be mitigated by applying fault tolerance techniques such as redundancy or Error Correction Code (ECC). Triplicating the logic is an efficient method to cope with the effects of single faults in the design. Additional user-level techniques can also be applied to deal with Silent Data Corruptions (SDC) that are incorrect results outputs. Moreover, periodic reset may be required to reestablish the system state and restore the initial state of flip-flops. Since SEFIs may also affect internal control elements of the FPGA or the configuration interface, a complete power cycle might be required to restore the system.

55.3 Operation

The GRSCRUB main operational modes are programming the FPGA and scrubbing the configuration memory. Figure 147 shows the basic flow of these two operational modes (OPMODE), and table 915 describes their codes to be set in the Configuration Mode Register (CONFIG). Optionally, the GRSCRUB can also be configured to compute and save the FPGA design CRC codes to a memory.

After the system reset, the GRSCRUB remains in the idle state until the enable (EN) bit of the Configuration Mode Register (CONFIG) is set high. The operational mode must be configured before, or at the same time, the GRSCRUB is enabled. The OPDONE and SCRERR bits in the Status Register (STATUS) should be cleared before enabling the core, see section 55.12.1 for more details. The GRSCRUB returns to the idle state when the EN bit of the Configuration Mode Register (CONFIG) is disabled (logically 0).

If the programming mode is selected, the GRSCRUB starts the configuration sequence to program the FPGA. At the end of the programming phase, the OPDONE bit of the Status Register (STATUS) is set high, and the GRSCRUB returns to the idle state.

For the scrubbing operation mode, there are two types of execution: blind and readback scrubbing. The scrubbing can be defined as periodic with a delay between each memory scrubbing, or just one time. If a periodic execution is configured in its registers, the GRSCRUB repeats the scrubbing operation after a configured delay. If the GRSCRUB is disabled, the OPDONE bit of the Status Register (STATUS) is set, and the GRSCRUB returns to the idle state.

Note: After enabling the GRSCRUB to run an operation mode, the values of the configuration registers should not be changed. Otherwise, it may cause an error in the GRSCRUB execution. If the GRSCRUB is disabled during execution, it will stop the operation, send a command to desynchronize the FPGA slave SelectMap and return to the idle state. Therefore, the current operation mode will be aborted.

The following sections describe in more detail the GRSCRUB operational modes.

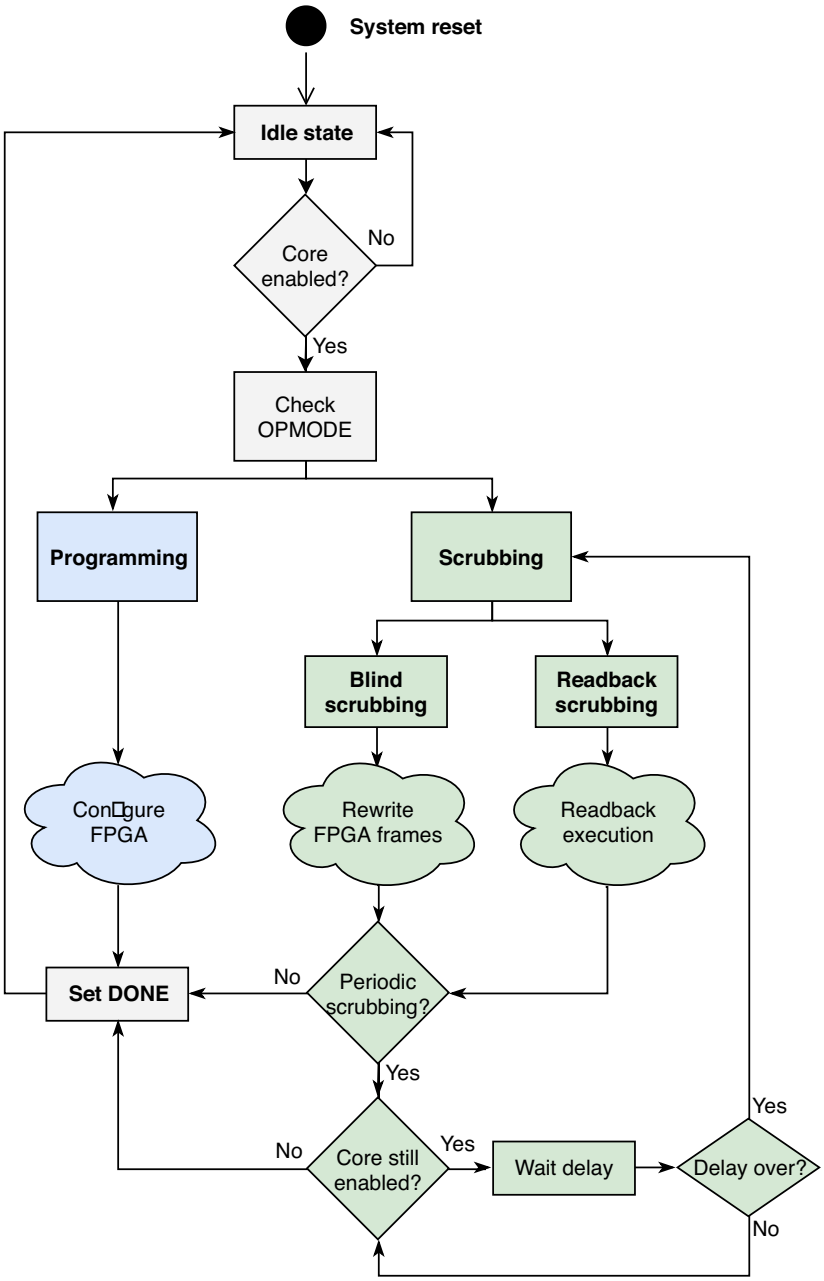


Figure 147. GRSCRUB flow for programming and scrubbing operation modes

Table 915. GRSCRUB operation modes description

Operation Mode	Code	Description
Programming	0001	FPGA configuration. Program the configuration bitstream into the FPGA.
Scrubbing	0010	FPGA scrubbing. Perform the blind or readback scrubbing.
Golden CRC	0100	Golden CRC codes generation. In this OPMODE the GRSCRUB computes the CRC codes for the FPGA design and saves the golden CRC copy to the memory. Note that this feature is not fully available in the GR716Bwith CQFP package.

55.3.1 Programming mode

The GRSCRUB can configure the FPGA by programming the configuration bitstream through the configuration interface. The configuration bitstream must be stored in the Golden memory, and the configuration bitstream address must be set in the GRSCRUB registers.

The following steps must be performed before enabling the GRSCRUB to execute the programming mode:

- The configuration bitstream must be stored in the Golden memory;
- The GRSCRUB must be disabled to configure the registers;
- The OPDONE and SCRERR bits in the Status Register (STATUS) should be cleared;
- Set the Configuration Mode Register (CONFIG) with the programming OPMODE (0001);
- Set the low and high configuration bitstream addresses in the Low Golden Bitstream Address Register (LGBAR) and High Golden Bitstream Address Register (HGBAR), respectively;
- Set the FPGA Device Identifier Register (IDCODE).

After setting the registers, the GRSCRUB can be enabled by writing the logical 1 in the EN bit of the Configuration Mode Register (CONFIG). When the programming phase is finished, the OPDONE bit goes to high, and an interrupt is generated (if interrupts are enabled).

55.3.2 Scrubbing mode

The user can define the scrubbing operational mode as blind or readback. Since the FPGAs are frame-oriented, the configuration memory is scrubbed considering the frame size. For instance, the frame length of the AMD/Xilinx UltraScale FPGA family is 123 words of 32 bits each, whereas the Virtex-5 frames have 41 32-bit words. The FPGA frame length and the total number of frames must be configured in the Frame Configuration Register (FCR).

A scrubbing run is defined as one execution of the blind or readback scrubbing in all the selected configuration frames. The scrubbing can be set to be periodic, which means that scrubbing runs are periodically executed in time.

Note: The scrubbing mitigation technique only fixes bit-flips in the FPGA configuration memory, it is up to the user to apply any additional method to mask errors and re-establish the state of the system. In addition, scrubbing does not cover soft errors affecting user memory and dynamic data.

The total number of frames, frame length, and the starting FPGA frame address should be configured in the registers in advance. Only frames from the configuration memory must be set. The frames from the FPGA configuration block (type 000) include the configuration elements, such as CLBs, I/Os, and clocks. See the FPGA configuration manual for more details about block types.

The steps below must be performed before enabling the GRSCRUB to execute the scrubbing mode. If a register has been previously set and the data has not changed, there is no need to reconfigure it.

- The configuration bitstream must be stored in the Golden memory;
- The FPGA must be previously programmed;
- The GRSCRUB must be disabled to configure the registers;
- The OPDONE and SCRERR bits in the Status Register (STATUS) should be cleared;
- Set the Configuration Mode Register (CONFIG) with the scrubbing OPMODE (0010);
- Set whether the scrubbing mode is blind (logical 0) or readback (logical 1) in the SCRM bit of the Configuration Mode Register (CONFIG). Blind scrubbing is set by default;
- Set whether the scrubbing run is just one time (logical 0) or periodic (logical 1) in the SCRUN bit of the Configuration Mode Register (CONFIG). One time is set by default. If a periodic run is set, the Delay Register (DELAY) should also be configured. No delay between runs is set by default;

- Set the low and high configuration bitstream addresses in the Low Golden Bitstream Address Register (LGBAR) and High Golden Bitstream Address Register (HGBAR), respectively;
- Set the address for the first golden frame in the Low Golden Start Frame Address Register (LGSFAR). Note that the first golden frame address is different from the low bitstream address. The LGSFAR should be the address in the Golden memory that saves the first word of the first frame in the bitstream;
- Set the address for the first frame of the FPGA configuration memory to be scrubbed in the Low Frame Address Register (LFAR);
- Set the FPGA Device Identifier Register (IDCODE);
- Set the number of frames and the frame length in the Frame Configuration Register (FCR).

Additional configuration is required for readback mode:

- The mask data must be stored in the Golden memory;
- The frame addresses must be stored in the Golden memory (required also for blind scrubbing if the BLKFRAME > 0 in the SETUP register);
- Set the low mask address in the Low Golden Mask Address Register (LMASKAR) (required also for blind scrubbing if the BLKFRAME > 0 in the SETUP register);
- Set which data check is enabled to detect errors in the FFCEN and CRCEN bits of the Configuration Mode Register (CONFIG);
- Set if the readback scrubbing mode is only detection (logical 1) or detection and correction (logical 0) in the CORM bit of the Configuration Mode Register (CONFIG). Detection and correction is set by default.

Set the Low Golden Frame Mapping Address Register (LFMAPAR) with the low address of the mapped frames in the Golden memory (only required if correction is selected).After setting the registers, the GRSCRUB can be enabled by writing 1 in the EN bit of the Configuration Mode Register (CONFIG). The OPDONE bit is only asserted if one time scrubbing is selected. The SCRUND bit goes high after each scrubbing execution.

I. Blind scrubbing mode

In the blind scrubbing operation mode, the GRSCRUB will rewrite each FPGA frame from the configuration memory without any verification. It does not provide the error detection capability. The blind scrubbing is configured to reprogram the configuration memory frames set in the Frame Configuration Register (FRC) and Low Frame Address Register (LFAR). Blind scrubbing does not rewrite user memory elements.

II. Readback scrubbing mode

In the readback scrubbing operation mode, the GRSCRUB verifies the integrity of each FPGA frame of the configuration memory and, in case of errors, rewrites the frame with correct data from the Golden memory. The error detection can be performed through CRC verification and by comparing all frame bits with the golden copy, here defined as Full Frame Check (FFC). Each type of verification can be configured to be enabled or not. The error detection requires at least one type to be enabled, and both can be enabled simultaneously. The CRC and FFC data checks are further described in the next sections.

The GRSCRUB can operate in two types of readback scrubbing modes: only error detection, and error detection and correction. For error detection operation, the mask data must be previously stored in the Golden memory. The correction mode is used to correct a faulty frame during readback scrubbing. The GRSCRUB replaces the erroneous frame in the FPGA with the golden frame read from the Golden memory. To enable the error correction, the CORM bit in the Configuration Mode Register (CONFIG) must be set to 0. By default, the correction is enabled. However, if the CORM bit is set to 1, the errors are only detected and not corrected in the readback phase.

Optionally, the GRSCRUB has a debug feature to save the readback data to a RAM memory. For that, the WRBKEN bit of the Configuration Mode Register (CONFIG) must be enabled. Also, the low memory address to save the readback data must be set in the Low Golden Readback Address Register (LGRBKAR). These steps are not required for other modes. Note that if the WRBKEN bit is enabled, the GRSCRUB does not detect or correct any error, it only copies the configuration data frame read from the FPGA to the memory. The GR716B microcontroller available in the CQFP package includes a small amount of RAM memory which restricts the usage of this feature. Future versions of the component that comprises the memory controller described in chapter 23 may allow using this feature to save the readback of all frames in the FPGA configuration memory.

Figure 148 describes the flow for the GRSCRUB scrubbing operation mode for readback execution with error detection and correction enabled. The GRSCRUB starts reading a frame word from the FPGA configuration memory. If CRC is enabled, the GRSCRUB computes the CRC signature of the word. In sequence, if FFC is enabled, the word is compared with the golden copy. At the end of the frame, the CRC code for the entire frame is checked and the frame is corrected if an error is detected. For FFC, the frame is corrected immediately when a faulty word is identified. After correcting a frame, the corrected frame is reread and rechecked to ensure the erroneous bits are fixed. If the corrected frame still presents an error, an uncorrectable error is counted, and the next frame is read. In case of uncorrectable errors, the UNCORRECTABLE_BIT_ERROR is set, and the SCRERR bit is asserted. It is recommended to reprogram the FPGA after two consecutive scrubbing runs with uncorrectable error reports.

Differently from the blind scrubbing, the readback mode allows detecting errors and correcting the frame only if necessary, at the expense of more access to the Golden memory to read golden and mask data.

i. Cyclic Redundancy Check (CRC)

CRC is an error detection code that applies redundancy to check inconsistencies. A 32-bit CRC (CRC32C) checksum algorithm is computed for each FPGA frame. The CRC32C polynomial is:

$$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^{8}+x^{7}+x^{5}+x^{4}+x^{2}+x^{1}+1$$

The defined initial CRC signature for each frame is 0xFFFFFFFF. The first frame word is read and masked, using the corresponding mask word from the Golden memory. Then, the CRC32C is applied using the signature. The output of this computation results in the next signature for the next readback frame word. That cycle is repeated for all words in the frame (frame length). The last computed signature corresponds to the frame CRC code.

During regular readback operation, the GRSCRUB reads the frame word from the FPGA, reads the mask word from the Golden memory, and computes the CRC signature. When it reaches the end of the frame, the generated CRC code is compared with the golden code from the Golden memory. If an error is detected, the faulty frame is corrected (if correction is enabled). Otherwise, the next frame is verified.

The CRC mode does not provide the total number of errors in a frame, and neither the error position. The value reported in the Number of Errors Detected Register (ECNT) is related to the number of detected faulty frames. The CRC method has single error detection ensured. However, for more accurate error counting, the FFC method is recommended.

Before performing readback with CRC checking, the golden CRC codes must be previously saved into the Golden memory.

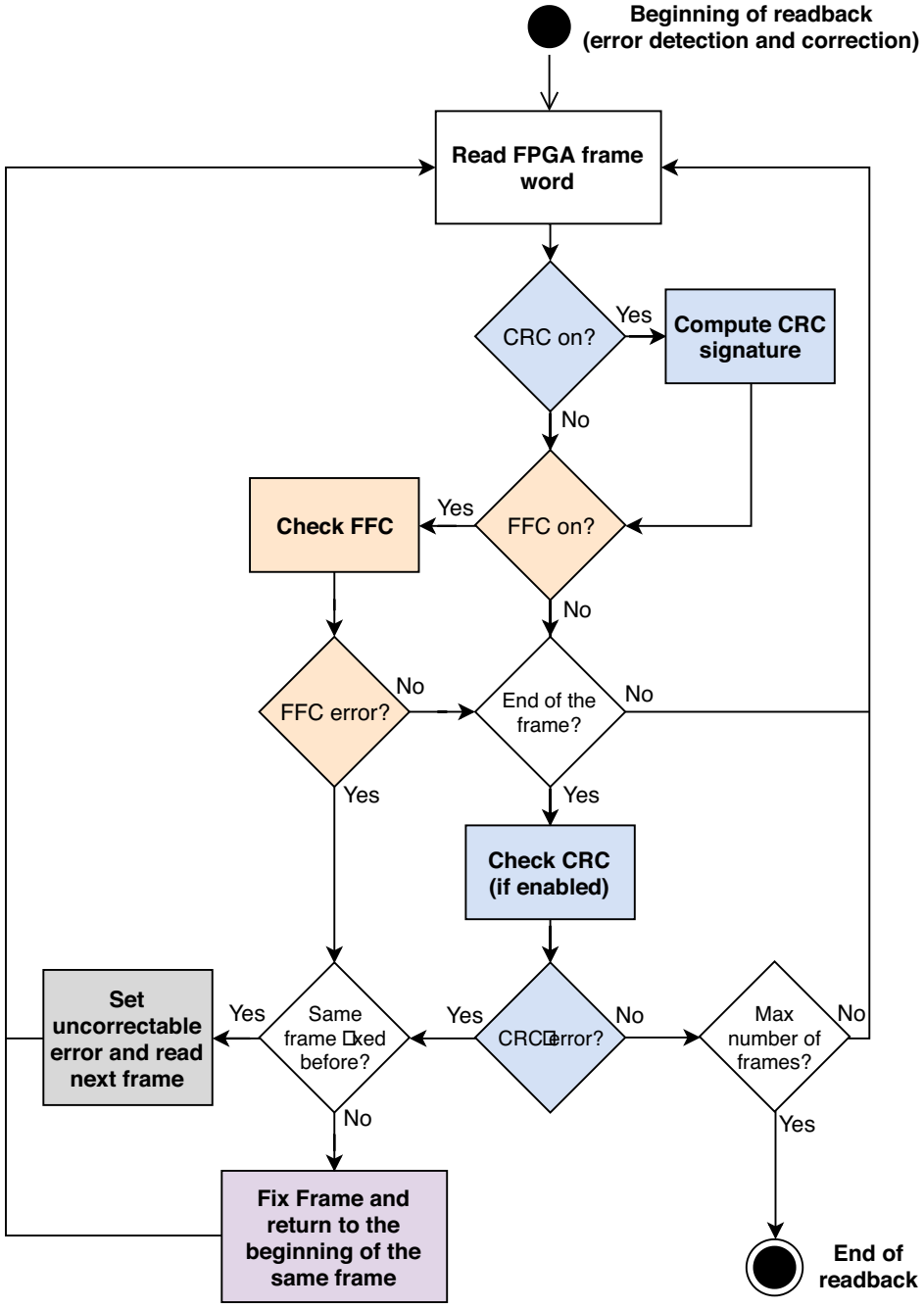


Figure 148. GRSCRUB flow in readback scrubbing mode for error detection and correction

ii. Full Frame Check (FFC)

In the FFC mode, the GRSCRUB compares all the configuration frame words with the golden words. Therefore, the number of error bits per readback word is known. However, when the correction is enabled, the frame is corrected as soon as the first error word is detected. Even though more faulty bits might be present, only bit errors in the first error word are counted for that frame. FFC is a more reliable method to detect errors than CRC since each word is compared with its golden.

55.3.3 Golden CRC mode

The golden CRC codes must be stored in the Golden memory before enabling the readback scrubbing with CRC data check, see section 55.3.2 for more details about CRC. The GRSCRUB has the operation mode to compute and save the golden CRC codes to memory. In this case, the memory must be writable to allow writing the CRC golden codes (i.e., RAM memory). Note that the GR716B with the CQFP package has limited internal RAM memory, which restricts the usage of this feature. Future versions of the microcontroller that use the NVRAM controller (see chapter 23) may allow utilizing this feature in full.

The following steps must be performed before enabling the GRSCRUB to execute the golden CRC operational mode:

- The FPGA must be programmed;
- The frames addresses must be stored in the Golden memory;
- The GRSCRUB must be disabled to configure the registers;
- The OPDONE and SCRERR signals in the Status Register (STATUS) must be cleared;
- Set the Configuration Mode Register (CONFIG) with the golden CRC OPMODE (0100);
- Set the address for the first frame of the FPGA configuration memory to be scrubbed in the Low FPGA Frame Address Register (LFAR);
- Set the FPGA Device Identifier Register (IDCODE);
- Set the number of frames and the frame length in the FCR register;
- Set the low address for the first CRC code in the LCRCAR. Note that in this case the LCRCAR should point to the start address of the RAM memory to write the CRC codes.
- The mask data must be previously stored in the Golden memory;
- Set the low mask address in the Low Golden Mask Address Register (LMASKAR).

After setting the registers, the GRSCRUB can be enabled by writing the logical 1 in the EN bit of the configuration register. When the GRSCRUB finishes the CRC computation, the OPDONE bit goes to high, and an interrupt is generated (if interrupts are enabled).

55.4 Mask data information

The FPGA configuration memory also includes dynamic elements that change their values during design execution, such as LUTRAMs and shift registers (SRL). During the configuration bitstream generation, the synthesis tool can create a mask file that indicates all dynamic bits as '1'. These bits should not be verified during the error detection steps. Otherwise, errors would be triggered when the bits are dynamically updated at normal design execution.

The GRSCRUB applies the mask word to the readback word before each data checking, following the steps below:

1) Read word from FPGA configuration memory (*readback_word)*;

2) Read mask word from Golden memory (*mask_word)*;

3) All bits of the mask word are inverted: *NOT mask_word*;

4) An *AND* operation of the readback word and the inverted mask word is performed.

- *Example:*

  mask_word: 0x0000FFFF
  readback_word: 0x0FE50400
  Result: readback_word AND (NOT mask_word) = 0x0FE50000
  The 0x0FE50000 result is applied to the fault detection methods.

The mask data must be saved on the Golden memory, and the start address should be set in the Low Golden Mask Address Register (LMASKAR). Since the GRSCRUB accesses only the masked data from the configuration frames (i.e., configuration block 000), it is not required to store the entire mask file in the Golden memory. However, the mask words must exactly corresponding to the readback words. The first mask word, whose address is set in the LMASKAR register, is related to the first word of the first readback frame, whose address is set in the Low FPGA Frame Address Register (LFAR).

The mask file has the same format as the configuration bitstream file. If the user decides to store only the mask words related to the scrubbed frames, the correlation between both must be matched. A misalignment would cause erroneous verification of the bits.

Since the masked bits are not verified in the readback mode, faults affecting those bits cannot be detected.

55.5 FPGA frame information

Table 916 shows the configuration bitstream and fame length for the AMD/Xilinx Virtex-5 and Kintex UltraScale FPGAs.

Note: Each FPGA device has a different configuration bitstream length, number of frames, and frame length. See the FPGA documentation for more information.

*Table 916.*Configuration bitstream information for the AMD/Xilinx Virtex-5 and Kintex UltraScale FPGAs

	Virtex-5 (XC5VFX130T)	**Kintex UltraScale (KU060)**
Configuration bitstream length	6,154,368 bytes	24,124,908 bytes
Total number of frames	37,520 frames	49,030 frames (plus 537 overhead words)
Block configuration frames[ab]	25,980 frames (4,260,720 bytes)	37,499 frames (18,449,508 bytes)
Frame length	41 32-bit words	123 32-bit words

a.Only the configuration frames are verified (block type 000: CLBs, DSPs, and IOBs).
b.Block RAM contents are not included (block type 001).

55.6 Golden memory

The GRSCRUB requires access to a Golden memory in which the golden data must be stored. The external memory access is made through the AMBA AHB bus. The stored data is related to the configuration bitstream, mask file, mapped frame addresses, and golden CRC codes. All the required data must be stored in the external memory so that GRSCRUB can properly execute its operations. The required data depends on the applicable GRSCRUB operations planned for a specific system. Figure 149 describes an example of data saved in the Golden memory and the GRSCRUB registers that must be configured, as follows:

- Golden configuration bitstream: The entire FPGA configuration bitstream (*.bit*) must be saved on memory. The bitstream is mandatory for all operations since it is necessary for programming and scrubbing the FPGA. The Low Golden Bitstream Address Register (LGBAR) and High Golden Bitstream Address Register (HGBAR) correspond to the addresses of the first and last configuration bitstream words, respectively. The Low Golden Start Frame Address Register (LGSFAR) is the address in the Golden memory of the first word of the first frame to be scrubbed.
- Mask data: The FPGA configuration mask (*.msk*) is required for readback scrubbing. GRSCRUB uses the mask data to identify the dynamic bits in the FPGA CRAM frames. Only the data related to the scrubbed frames are needed to be stored. For simplicity, the entire mask file can be loaded to the golden memory. The mask file has the same format as the bitstream file. If the user decides to store only the mask data related to the scrubbed frames, the correlation between the mask and bitstream must be matched. Misalignment will cause erroneous verification of the bits. The Low

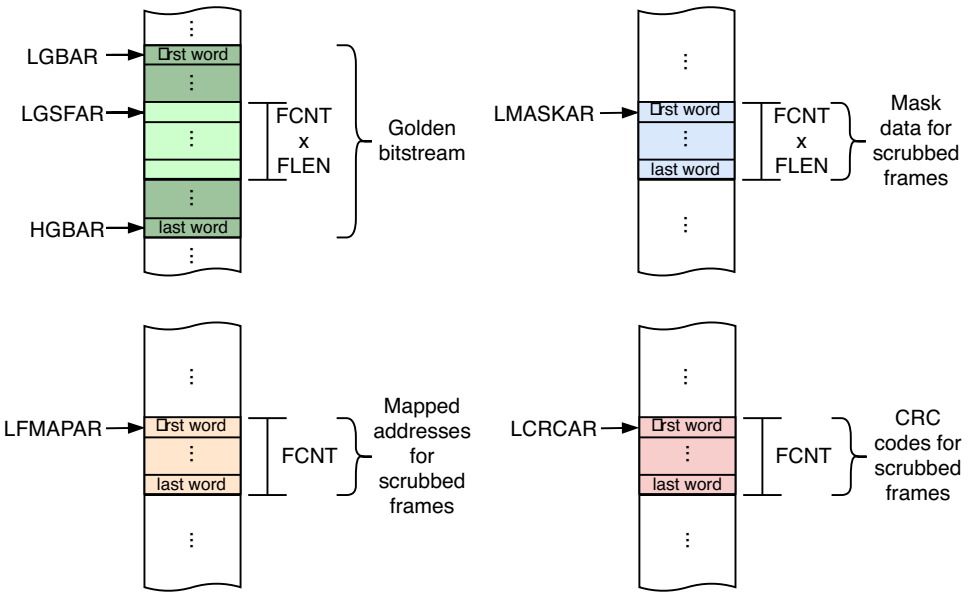


Figure 149. Example of data storage configuration in the Golden memory and the relation with the GRSCRUB registers

Golden Mask Address Register (LMASKAR) is the address of the first mask word of the first frame word to be scrubbed.

- Mapped frames addresses: The FPGA CRAM frame addresses are required in case of error correction of a specific frame. Therefore, frame mapping is mandatory for the readback scrubbing. The blind scrubbing only uses the frame mapping if a limited number of frames is configured to be scrubbed at once (BLKFRAME>0) in the SETUP register. If no maximum limit is established (BLKFRAME=0), the frame mapping is not required for blind scrubbing. The Low Golden Frame Mapping Address Register (LFMAPAR) is the address in the Golden memory that stores the FPGA frame address of the first frame to be scrubbed.
- CRC codes: The golden CRC codes are only required to be stored in the Golden memory for the readback scrubbing with CRC error detection. One CRC code is saved per frame. The Low Golden CRC Address Register (LCRCAR) is the address in the Golden memory of the first golden CRC code of the first frame to be scrubbed.

The FCNT and FLEN bitfields of the Frame Configuration Register (FCR) are the number of scrubbed frames and length of each frame, respectively. Therefore, they also represent the number of memory positions that must be sequentially accessed by the GRSCRUB in each operation.

For example, Table 917 describes the storage length required for the AMD/Xilinx Virtex-5 and Kintex UltraScale FPGAs. Note that each FPGA device has a different configuration bitstream length. See the FPGA documentation for more information.

Table 917. Description of the required Golden memory storage for AMD/Xilinx Virtex-5 and Kintex UltraScale FPGAs

Data Memory	**Virtex-5 [XC5VFX130T] (bytes)**	**Kintex UltraScale [KU060] (bytes)**
Configuration bitstream	6,154,368	24,124,908
Mask[a]	4,260,720	18,449,508
Mapped frame addresses[b]	103,920	149,996
CRC codes	103,920	149,996
Total required storage	**10,645,661**	**42,907,220**

a.Since the GRSCRUB accesses only the mask data related to the configuration frames, it is not required to store the entire file.
b.Only configuration frames are mapped.

55.7 SelectMap configuration interface

The SelectMap configuration interface is used to load the configuration bitstream into the FPGA and for scrubbing operations. The GRSCRUB connects directly to the SelectMap pins, which are controlled and used as follows:

- **CCLK (connected to GR716B SCRUB_SCLK I/O):** The SelectMap configuration clock (SCRUB_SCLK) is generated from the SYS_CLK, and it is controlled by the GRSCRUB clock enable signal (SMAPO.clk_en). By stopping the CCLK, it is possible to pause the data loading or reading from the SelectMap.
- **PROGRAM_B (connected to GR716B SCRUB_PROG I/O):** This signal is used to clean the FPGA configuration logic before starting the programming phase. The GRSCRUB sets the PROGRAM_B to low during the number of clock cycles defined in the TPROG register. See in the FPGA documentation the minimum pulse width required. By default, TPROG is 150 clock cycles.
- **INIT_B (connected to GR716B SCRUB_INITN I/O):** It indicates when the FPGA is ready to receive configuration data. It also indicates any configuration or readback errors by low pulse.
- **DONE (connected to GR716B SCRUB_DONE I/O):** The GRSCRUB monitors the FPGA DONE signal to identify the completion of the programming sequence.
- **CSI_B (connected to GR716B SCRUB_CSIN I/O):** This signal enables (logical 0) and disables (logical 1) the SelectMap interface.
- **RDWR_B (connected to GR716B SCRUB_RDWR I/O):** It is used to select the direction of the data bus. When RDWR_B is high (logical 1), the SelectMap data pins are outputs (reading from the FPGA). Otherwise, the bus is input (loading data to the FPGA). The GRSCRUB only changes the RDWR_B signal when the SelectMap interface is disabled (i.e., when CSI_B is logically 1) to avoid errors in the interface.
- **D[7:0] (connected to GR716B SCRUB_DATA[7:0] I/Os):** The bidirectional data bus is used for configuration and scrubbing operations. The RDWR_B signal defines the direction of the pins. The GR716B supports 8-bit data bus width.
- **M[2:0] (not connected to GR716B):** The FPGA configuration mode pins must be configured to logical "110" to select the slave SelectMap interface. For that, the FPGA pins M[2:1] can be tied to a high level and pin M[0] can be tied to a low level on the FPGA board.

In order to allow the GRSCRUB to access and control the slave SelectMap interface, the generated FPGA configuration bitstream must be configured following the requirements presented in Table 918.

*Table 918.*Example of configuration bitstream settings and constraints for enabling the FPGA slave SelectMap interface

Configuration bitstream settings	- Enable the mask file generation - Do not prohibit readback in the configuration bitstream security settings - Do not use encryption in the configuration bitstream - Set the SelectMap pins to persistent in the configuration bitstream generator (bitgen)[a]: -g Persist:Yes or BITSTREAM.CONFIG.PERSIST YES - Set the constraints to allow programming and readback the FPGA through slave SelectMap interface: CONFIG CONFIG_MODE=S_SELECTMAP32+READBACK

a. Note that the ICAP interface cannot be used in the FPGA design if Persist:Yes is set.

55.8 Soft errors affecting the FPGA configuration interface

The FPGA SelectMap configuration interface might also be affected by soft errors, which may lead to catastrophic results during the scrubbing operation. For instance, if during a blind scrubbing, the FPGA frame address register is affected and its value is changed to another valid address, all the following frames would be wrongly overwritten, and the design would be compromised.

A safer scrubbing approach is to set up the configuration interface for each scrubbed frame, instead of configuring all frames at once. For instance, writing one frame at a time during blind scrubbing avoids overwriting the entire memory in case of errors in the FPGA frame address register. The BLKFRAME bitfield in the Setup Register (SETUP) defines the number of frames to be scrubbed sequentially during blind or readback scrubbing. If BLKFRAME is 0, all frames are scrubbed at once. The default is scrubbing only one frame at a time (BLKFRAME = 1). If BLKFRAME>0 is used, the mapping frame addresses must be stored in the Golden memory for any scrubbing mode.

55.9 Interrupts

Interrupts are generated to indicate the task is finished (when the OPDONE bit is asserted) and internal GRSCRUB errors (when the SCRERR bit is asserted). Additionally, when the SCRUND bit is asserted, an interrupt can be generated.

Interrupts are disabled by default. The IRQDEN, IRQEEN, and IRQSDEN bits of the Configuration Mode Register (CONFIG) should be high to allow interrupts for OPDONE, SCRERR, and SCRUND, respectively. All interrupts are connected to the interrupt controller to inform the processor of the events. The interrupts are pulse signals, being asserted for one clock cycle.

The usual procedure is that an interrupt routine handles the interrupt bits in the Status Register (STATUS). The status bits are cleared by writing one.

55.10 GRSCRUB error codes

Table 919 describes the GRSCRUB internal error codes. When an internal error occurs, the GRSCRUB sets the ERRID bit of the Status Register (STATUS) with the error code, sets the SCRERR to high, and launches an interrupt (if interrupts are enabled).

For all errors, except for UNCORRECTABLE_BIT_ERROR, the GRSCRUB safely stops the current operation, finishes the synchronization with the FPGA, waits for the end of the last memory access, and returns to the idle state. The SCRERR bit must be cleared before enabling the GRSCRUB again and starting a new operation.

The UNCORRECTABLE_BIT_ERROR means that a bit-error was not corrected during a readback scrubbing run. When an error is detected in the last corrected frame, the GRSCRUB flags an uncorrectable error for that frame and continues the scrubbing operation for the other frames. Often, an uncorrectable error is reported along with other correctable errors in the same scrubbing run, and a sequential scrubbing run corrects the stuck bit error.

The DMA errors are related to the Golden memory and are most likely to occur due to invalid memory accesses. For the other errors, it is recommended to properly reset and reprogram the FPGA.

*Table 919.*GRSCRUB internal error codes

Error	Code	Description
NO_ERROR	00000	GRSCRUB without errors. The GRSCRUB is working correctly. The SCRERR signal remains low, and no interrupt is launched.

*Table 919.*GRSCRUB internal error codes

Error	Code	Description
PROGRAM_ERROR	00001	Error programming the FPGA configuration bitstream. This error is triggered when the INIT_B signal from the SelectMap interface pulses low during the programming phase, which means an error to configure the FPGA. Possible causes: - Frame length and number of frames wrongly set in the register; - Wrong configuration bitstream data stored in the Golden memory; - Wrong addresses set in the registers; - Wrong TPROG; - Slave SelectMap interface not set in the FPGA configuration mode pins M[2:0].
UNCORRECTABLE_BIT_ERROR	00010	Uncorrectable error(s) in the configuration memory during readback scrubbing. After correcting a faulty frame, the replaced frame still presents an error. That might occur if there is a stuck error bit, or a new error affected the same frame in the time between correction and verification. Some CRAM bit errors require two sequential scrubbing runs to be corrected. The SRAM-based FPGAs present two types of configuration memory errors: a single point error is defined when an upset affects a function logic bit, and a burst of errors is defined when a single upset directly affects the state of multiple bits. The burst of errors may occur due to upsets in configuration bits responsible for controlling the status of other bits in the FPGA. When a control bit is flipped, all other bits under its control are also affected and may only return to the correct state after the original control bit is corrected. In some scenarios, it is required two scrubbing runs to recover all bits and return the configuration memory to a consistent state. The first scrubbing run will report UNCORRECTABLE_BIT_ERROR, and the second run will correct the remaining bits. If after a few scrubbing runs, the uncorrectable error is still flagged, that means a true stuck error. This is the only internal error that does not lead the GRSCRUB to return to the idle state. The readback scrubbing continues the execution. In the UE_FRMID bitfield of the Error Frame Id Code Register (ERRFRMID) is informed the frame id where the last uncorrectable error occurred. The register is overwritten at each uncorrectable error.
DMA_MEM_READ_ERROR	00011	DMA error reading the Golden memory. Possible cause: - Illegal memory access; - The external memory is not properly configured.
DMA_MEM_WRITE_ERROR	00100	DMA error writing the Golden memory. Possible causes: - Illegal access; - The external memory is not properly configured; - It is a PROM memory.

*Table 919.*GRSCRUB internal error codes

Error	Code	Description
READBACK_ERROR	00101	Error during readback operation. This error is triggered when the INIT_B signal from the SelectMap interface pulses low during the readback phase, which means an error to read the FPGA frames. In the Frame Id Code Register (FRAMEID) is informed the id of the last scrubbed frame. Possible causes: - Frame length and number of frames wrongly set in the register; - Wrong addresses set in the registers; - The FPGA is not programmed; - The programmed bitstream does not allow readback or does not follow all constraints from Table 918. - Frames addresses are not stored in the Golden memory; - Slave SelectMap interface not set in the FPGA configuration mode pins M[2:0].
WRITE_FRAME_ERROR	00111	Error in writing an FPGA frame. Error during frame correction during readback, or error during blind scrubbing. In the Frame Id Code Register (FRAMEID) is informed the id of the last written frame. Possible causes: - Frame length and number of frames wrongly set in the register; - Wrong configuration bitstream data stored in the Golden memory; - Wrong addresses set in the registers; - The FPGA is not programmed; - Frames addresses are not stored in the Golden memory.

55.11 Enabling and disabling the GRSCRUB

Before enabling the GRSCRUB, all the registers should be configured according to the specific operation mode. The GRSCRUB is enabled when the EN bit of the Configuration Mode Register (CONFIG) goes to high. The EN bit must stay high while the GRSCRUB is executing the operation. At the end of the execution, the GRSCRUB sets the OPDONE bit to high, sets the EN bit to low, and returns to the idle state.

After enabling the GRSCRUB, the registers should not be updated. Otherwise, a malfunction in the operation sequence can occur. In addition, if the GRSCRUB is disabled (i.e., by setting the EN bit to low) during execution, the operation will not be finished properly. In that case, the GRSCRUB stops the current operation, finishes the synchronization with the FPGA, waits for the end of the last memory access, and returns to the idle state. The EN bit should only be enabled again after the GRSCRUB has returned to an idle state.

The OPDONE and SCRERR bits should always be cleared before enabling the GRSCRUB and starting a new operation.

55.12 Registers

The GRSCRUB is programmed through registers mapped into APB address space, which are shown in table 920. Only 32-bit single-accesses to the registers are supported.

Table 920. GRSCRUB registers

APB address offset	Register description
0x00	Status register (STATUS)
0x04	Configuration mode register (CONFIG)
0x08	FPGA device identifier register (IDCODE)
0x0C	Delay register (DELAY)
0x10	Frame configuration register (FCR)
0x14	Low FPGA frame address register (LFAR)
0x18	Low golden bitstream address register (LGBAR)
0x1C	High golden bitstream address register (HGBAR)
0x20	Low golden start frame address register (LGSFAR)
0x24	Low golden mask address register (LMASKAR)
0x28	Low golden frame mapping address register (LFMAPAR)
0x2C	Low golden CRC address register (LCRCAR)
0x34	Number of errors detected register (ECNT)
0x38	FPGA setup register (SETUP)
0x3C	Capability register (CAP)
0x40	Frame id code register (FRAMEID)
0x44	Error frame id code register (ERRFRMID)

55.12.1 Status Register (STATUS)

*Table 921.*0x00 - STATUS - Status register

31 13	12	11	10 6	5	4	3 0
RESVD	SCRUND	HOLD	ERRID	SCRERR	OPDONE	STATE
0	0	0	0	0	0	0
r	wc	r	r	wc	wc	r

31: 13 RESVD RESERVED
12 SCRUND Scrubbing run done. It indicates the end of a scrubbing run.
The SCRUND bit goes high after each scrubbing execution in periodic run.
Write one to clear.
Generate interrupt (if enabled).
11 HOLD GRSCRUB in hold during periodic scrubbing (read only).
It identifies when the GRSCRUB is waiting during the delay period.
See the DELAY Register for more details.
0 = release; 1 = GRSCRUB is holding due to delay.
10: 6 ERRID Internal error code (read only)
It is cleared when SCRERR is set to 0.
Table 919 presents all error codes.
5 SCRERR GRSCRUB internal error: 0 = no error; 1 = error.
Write one to clear.
Generate interrupt (if enabled).
See ERRID to error details.
4 OPDONE Operation mode completed
Write one to clear.
Generate interrupt (if enabled).
3:0 STATE Current state (read only).
It indicates the current operation mode of the GRSCRUB.
STATE = OPMODE.

55.12.2 Configuration Mode Register (CONFIG)

*Table 922.*0x04 - CONFIG - Configuration mode register

31 14	13	12	11	10	9	8	7 4	3	2	1	0
RESVD	IRQSD EN	FFC EN	CRC EN	WRBK EN	IRQE EN	IRQD EN	OPMODE	CORM	SCRM	SCRUN	EN
0	0	0	0	0	0	0	0	0	0	0	0
r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

31: 14 RESVD RESERVED
13 IRQSDEN Interrupt enable for SCRUND signal: 0 = disable; 1 = enable (pulsed interrupt).
12 FFCEN Data check FFC enable: 0 = disable; 1 = enable.
11 CRCEN Data check CRC enable: 0 = disable; 1 = enable.
10 WRBKEN Enable writing readback data to memory: 0 = disable; 1 = enable
If enabled, the GRSCRUB does not detect or correct errors. The WRBKEN bit has priority over the CORM bit.
Note that this feature is only available for RAM memory. See the LGRBKAR register (section 55.12.13) for more information.
9 IRQEEN Interrupt enable for SCRERR signal: 0 = disable; 1 = enable (pulsed interrupt).
8 IRQDEN Interrupt enable for OPDONE signal: 0 = disable; 1 = enable (pulsed interrupt).
7: 4 OPMODE Operational mode:
0000 = idle state (default);
0001 = programming mode;
0010 = scrubbing mode;

		0100 = golden CRC mode; others = reserved. For more details, see the operational modes section 55.3.
3	CORM	Correction mode: 0 = detection and correction (default); 1 = only detection. For the CORM bit to have effect the SCRM bit must be 1 (readback mode enabled), and the WRBKEN bit must be 0 (write to memory disabled).
2	SCRM	Scrubbing operational mode: 0 = blind (default); 1 = readback. Blind scrubbing is set by default.
1	SCRUN	Scrubbing run: 0 = one-time (default); 1 = periodic. A delay between runs is applied in the periodic execution. See the DELAY register. In this case, the OPDONE only goes to high if the GRSCRUB is disabled during the delay period (when the HOLD bit is high). The SCRUND bit goes high after each scrubbing execution in the periodic run. One time scrubbing is set by default.
0	EN	GRSCRUB enable - starts operation mode: 0 = disable; 1 = enable. After enabling the GRSCRUB, EN bit must be preserved high until the end of the operation (when OPDONE bit goes to high). Otherwise, the GRSCRUB is disabled, and the operation is not finished. See section 55.11 for more details.

55.12.3 FPGA Device Identifier Register (IDCODE)

*Table 923.*0x08 - IDCODE - FPGA device identifier register

31	0
IDCODE	
0	
rw	

31: 0 IDCODE FPGA device identifier code

The 32-bit FPGA identification code is defined in the FPGA specification and represents the specific device id based on the IEEE Std 1149.1 (JTAG).

55.12.4 Delay Register (DELAY)

*Table 924.*0x0C - DELAY - Delay register

31	0
DELAY	
0	
rw	

31: 0 DELAY Delay, in number of ticks, between full blind or readback scrubbing cycles. The delay is set only for periodic scrubbing runs and is applied after all configured frames being scrubbed. Therefore, after the last scrubbed frame, the GRSCRUB waits the number of ticks set in the DELAY register, then returns to the first frame configured, and restarts the scrubbing operation. No delay is set by default.
A tick is counted for each rising edge of the tick_in input signal of the GRSCRUB. The period in which the GRSCRUB stays in hold depends on the tick_in frequency directly.

The GRSCRUB tick_in is connected to the general purpose timer unit (GPTIMER0), timer 5, which can be configured as described in chapter 35.

During the delay period while the GRSCRUB is waiting, the HOLD bit of the Status Register (STATUS) is set to high. When the delay is over, the HOLD bit is set to low.

If the EN bit of the Configuration Mode Register (CONFIG) is disabled while the GRSCRUB is waiting during the delay period, the GRSCRUB stops the delay, sets the OPDONE bit, and returns to idle state.

55.12.5 Frame Configuration Register (FCR)

*Table 925.*0x10 - FCR - Frame configuration register

31 25	24 9	8 2	1 0
RESVD	FCNT	FLEN	RESVD
0	0	0	0
r	rw	rw	r

31: 25 RESVD RESERVED
24: 9 FCNT Number of FPGA memory frames to be scrubbed
8: 2 FLEN Frame length: number of 32-bit words in a frame
1: 0 RESVD RESERVED

55.12.6 Low Frame Address Register (LFAR)

*Table 926.*0x14 - LFAR - Low frame address register

31 0
LFAR
0
rw

31: 0 LFAR The lowest frame address in the FPGA range to be scrubbed. This is the starting frame address.

55.12.7 Low Golden Bitstream Address Register (LGBAR)

*Table 927.*0x18 - LGBAR - Low golden bitstream address register

31 0
LGBAR
0
rw

31: 0 LGBAR The lowest golden configuration bitstream address in the Golden memory. This must be the address of the first dummy word(0xFFFFFFFF) in the configuration bitstream, after the initial header.

All configuration bitstreams have an initial header with ASCII characters that provides some file information, which it is not required to program the FPGA. The synchronization phase starts at the first dummy word (0xFFFFFFFF).

55.12.8 High Golden Bitstream Address Register (HGBAR)

*Table 928.*0x1C - HGBAR - High golden bitstream address register

31 0
HGBAR
0
rw

31: 0 HGBAR The highest golden configuration bitstream address in the Golden memory.

55.12.9 Low golden start frame address register (LGSFAR)

*Table 929.*0x20 - LGSFAR - Low golden start frame address register

31	0
LGSFAR	
0	
rw	

31: 0 LGSFAR The lowest address to the first configuration bitstream frame in the Golden memory.

55.12.10Low Golden Mask Address Register (LMASKAR)

*Table 930.*0x24 - LMASKAR - Low golden mask address register

31	0
LMASKAR	
0	
rw	

31: 0 LMASKAR The lowest mask address in the Golden memory.

55.12.11Low Golden Frame Mapping Address Register (LFMAPAR)

*Table 931.*0x28 - LFMAPAR - Low golden frame mapping address register

31	0
LFMAPAR	
0	
rw	

31: 0 LFMAPAR The lowest address of the FPGA frame mapping in the Golden memory.

55.12.12Low Golden CRC Address Register (LCRCAR)

*Table 932.*0x2C - LCRCAR - Low golden CRC address register

31	0
LCRCAR	
0	
rw	

31: 0 LCRCAR The lowest address of the CRC data check in the Golden memory.

55.12.13Low Golden Readback Data Address Register (LGRBKAR)

*Table 933.*0x30 - LGRBKAR - Low golden readback data address register

31	0
LGRBKAR	
0	
rw	

31: 0 LGRBKAR The lowest address to save readback data in the memory.

This register stores the lowest address in RAM memory to save data from the FPGA readback. This is a debug feature that can be enabled by setting the WRBKEN bit in the CONFIG register. Note that the memory needs to be writeable and accessible via the AMBA AHB bus connected to GRSCRUB. The GR716B microcontroller available in the CQFP package includes a small amount of RAM memory

which restricts the usage of this feature. Future versions of the component that comprises the memory controller described in chapter 23 may allow using this feature to save the readback of all frames in the FPGA configuration memory.

55.12.14Number of Errors Detected Register (ECNT)

*Table 934.*0x34 - ECNT - Number of errors detected register

31 16	15 0
UECNT	RECNT
0	0
rw	rw

31: 16 UECNT Number of uncorrectable errors. UECNT is the number of frames that still presenting an error after correction. Only active in readback scrubbing mode and when correction is enabled (CORM=0).

15: 0 RECNT Number of errors detected. Only enable in readback scrubbing mode. If the error correction is enabled (CORM=0), this counter represents the number of frames detected with error. Since the frame is corrected when the first error is detected, only errors in the first faulty word are counted in the frame. If only detection is enabled (CORM=1) with FFC, this counter represents the total biterror detected in the configuration memory. However, only one error is counter per frame for CRC data check.

The register is only cleared at the system reset, and the error counters accumulate over scrubbing runs. The user should clear the register to initiate a new count.

55.12.15FPGA Setup Register (SETUP)

*Table 935.*0x38 - SETUP - FPGA setup register

31 30	29 22	21	20	19 12	11 4	3 0
RESVD	BLKFRAME	BITSWPEN	CISEL	TPROG	ROWBND	RESVD
0	1	1	0	150	2	0
r	rw	rw	r	rw	rw	r

31: 30 RESVD RESERVED

29: 22 BLKFRAME Maximum number of frames to be scrubbed at time (i.e., reading sequentially during readback, or writing sequentially during blind scrubbing).
If BLKFRAME > 0, the frames are scrubbed by blocks, and a maximum of *BLKFRAME* frames is scrubbed each time. Thus, the number of frames defined in the FCNT bitfield of the Frame Configuration Register (FCR) is split into the *BLKFRAME* size. This feature is used to have more control of the FPGA configuration interface, and to avoid catastrophic results when the interface is affected by soft errors.
If BLKFRAME = 0, all frames set in the FCR are scrubbed sequentially at once.
Default value = 1 (only one frame is scrubbed at time).

21 BITSWPEN FPGA bit swapping enable: 0 = disable; 1 = enable. Enabled by default.

20 CISEL Configuration interface selected (read only):
0 = SelectMap;
1 = Not used.

19: 12 TPROG Number of clock cycles to wait with PROGRAM_B signal asserted for FPGA programming (PROGRAM_B pulse width). Check the target FPGA Data Sheet for the TPROGRAM pulse width. One microsecond is safe for most AMD/Xilinx FPGAs.
TPROG >= 1 us / smapclki_period
Default value = 150 (wait 150 clock cycles).

11: 4 ROWBND Row boundary alignment: number of FPGA frames in the rows boundaries.
Default value = 2 (two frames in the rows boundaries - used for AMD/Xilinx Virtex-5 and Kintex UltraScale FPGAs).
It must be zero if there is no row boundaries in the FPGA. Check the target FPGA Data Sheet for the internal architecture.

3: 0 RESVD RESERVED

55.12.16Capability Register (CAP)

*Table 936.*0x3C - CAP - Capability register

31 23	22 20	19 18	17 16	15 4	3 0
RESVD	DATACK	SMBUS	CINT	FAMILY	RESVD
0	0	0	0	0	0
r	r	r	r	r	r

31: 23 RESVD RESERVED
22: 20 DATACK Supported data checks (read only):
bit#20 = FFC;
bit#21 = CRC;
bit#22 = Not used.
19: 18 SMBUS SelectMap bus wide in bits (read only):
00 = 8-bit wide;
others = Not used.
17: 16 CINT Supported configuration interfaces (read only):
bit#16 = SelectMap
others = Not used.
15: 4 FAMILY Supported FPGA families (read only):
bit#4 = AMD/Xilinx Virtex-5;
bit#5 = AMD/Xilinx Kintex UltraScale;
others = Not used.
3: 0 RESVD RESERVED

55.12.17Frame Id Code Register (FRAMEID)

*Table 937.*0x40 - FRAMEID - Frame id code register

31 16	15 0
RESVD	FRMID
0	0
r	rw

31: 16 RESVD RESERVED
15: 0 FRMID Frame id of the current scrubbed frame.
It is updated at every new scrubbed frame.
Frame id is also updated during mapping phase.

The frame id represents the identification of the frame, which is defined from 0 to FCNT-1. Thus, FRMID 0 means the first frame, whereas the FRMID FCNT-1 means the last one.

In case of READBACK_ERROR or WRITE_FRAME_ERROR, the FRMID represents the last frame processed by the GRSCRUB. In these cases, the reported FRMID might not be precise. The GRSCRUB has an internal FIFO to receive the amount of data from the FPGA, and the current frame id takes into account only the processed data.

The register is only cleared at the system reset.

55.12.18Error Frame Id Code Register (ERRFRMID)

*Table 938.*0x44 - ERRFRMID - Error frame id code register

31 16	15 0
UE_FRMID	ERR_FRMID
0	0
rw	rw

31: 16	UE_FRMID	Frame id of the last frame with uncorrectable error. It is only updated if a new uncorrectable error occurs. This bitfield is only valid if the UECNT from the Number of Errors Detected Register (ECNT) is higher then 0.
15: 0	ERR_FRMID	Frame id of the last frame detected with error during readback. It is only updated if a new frame with error is detected.

The frame id represents the identification of the frame, which is defined from 0 to FCNT-1. Thus, FRMID 0 means the first frame, whereas the FRMID FCNT-1 means the last one.

The register is only cleared at the system reset.

55.13 Implementation

55.13.1 Scrubbing run period

The period to execute one scrubbing run, which means the time to scrub all frames configured in the registers once, depends on the characteristics of the system presented below. Blind and readback scrubbing modes are affected by the following factors:

- Number of frames to be scrubbed, which is defined in the FCNT bitfield of Frame Configuration Register (FCR);
- Number of words in each frame, which is defined in the FLEN bitfield of Frame Configuration Register (FCR);
- The target AMD/Xilinx FPGA family;
- Clock frequencies: internal system clock, SelectMap clock (SCRUB_SCLK), and memory controller clock;
- The required time to access the Golden memory.

Additional factors affect the readback scrubbing, as following:

- The selected error detection type, which can be FFC and/or CRC defined in the Configuration Mode Register (CONFIG);
- If error correction is enabled or not, which is defined in the CORM bit of the Configuration Mode Register (CONFIG);
- The number of errors detected in the scrubbing run, which is presented in the RECNT bitfield of the Number of Errors Detected Register (ECNT) (if correction is enabled).

Table 939 generically describes the time parameters that affects the scrubbing period.

*Table 939.*Generic timing description

Name	Description
SCRUBP	Scrubbing period
TFRD	Time to read one frame from the FPGA
TFWR	Time to write one frame to the FPGA
TD	Time to detect an error in a frame
TC	Time to correct one frame

Considering the readback scrubbing mode with only detection enabled, the scrubbing period would be (detection time):

$$SCRUBP = [time\ to\ read\ a\ frame\ and\ detectan\ error] * [number\ of\ frames]$$

$$SCRUBP = (TFRD + TD)*FCNT$$

Considering the readback scrubbing mode with detection and correction enabled, the scrubbing period would be:

$$SCRUBP = [detection\ time] + [correct\ erroneous\ frames] + [recheck\ corrected\ frames]$$

$$SCRUBP = (TFRD + TD)*FCNT + (TC + TFWR)*RECNT + (TFRD + TD)*RECNT$$

56 LVDS IO

56.1 Overview

The LVDS transmitters can be configured and used as normal outputs controlled from software via registers describe in this chapter. On Revision 1 of GR716B device the Failsafe output (FS_OUT) from the LVDS receivers are interfaced with this cores status and interrupt registers. On Revision 0 of GR716B device the received data out from the LVDS receivers are interfaced with this cores status and interrupt registers. The inputs can be configured to generate interrupt on high or low level.

Function needs to be enabled in the clock gating unit and LVDS receivers and transmitters must be enabled in the system IO and LVDS configuration registers.

56.2 Operation

56.2.1 system overview

n/a

56.2.2 Detailed description

n/a

56.2.3 Access control

LVDS IO status and configuration can be accessed via registers

56.3 Registers

Table 940. PLL control and status registers

APB address offset	Register
0x8030B000	LVDS Interrupt register
0x8030B074	LVDS Interrupt status register
0x8030B078	LVDS Interrupt mask
0x8030B07C	LVDS Interrupt level
0x8030B010	LVDS output register
0x8030B080	LVDS Input register

Table 941. 0x8030B000 - STS - LVDS interrupt register

31	4	3	2	1	0
RESERVED		R3	R2	R1	R0
0x00000000		-	-	-	-
r		R	R	R	R

31: 4 Reserved

3 Rev 0: LVDS Receiver input 3(R3)
Rev 1: FS_OUT receiver 3

2 Rev 0: LVDS Receiver input 2 (R2)
Rev 1: FS_OUT receiver 2

Table 941. 0x8030B000 - STS - LVDS interrupt register

1 Rev 0: LVDS Receiver input 1 (R1)
Rev 1: FS_OUT receiver 1

0 Rev 0: LVDS Receiver input 0 (R0)
Rev 1: FS_OUT receiver 0

Table 942. 0x8030B004 - IRQ - LVDS interrupt status register

31	4	3	2	1	0
RESERVED		R3	R2	R1	R0
0x00000000		-	-	-	-
r		wc	wc	wc	wc

31: 4 Reserved

3 Rev 0: LVDS Receiver input 3(R3)
Rev 1: FS_OUT receiver 3

2 Rev 0: LVDS Receiver input 2 (R2)
Rev 1: FS_OUT receiver 2

1 Rev 0: LVDS Receiver input 1 (R1)
Rev 1: FS_OUT receiver 1

0 Rev 0: LVDS Receiver input 0 (R0)
Rev 1: FS_OUT receiver 0

Table 943. 0x8030B008 - MASK - LVDS mask register

31	4	3	2	1	0
RESERVED		R3	R2	R1	R0
0x00000000		0	0	0	0
r		rw	rw	rw	rw

31: 4 Reserved

3 Rev 0: LVDS Receiver input 3(R3)
Rev 1: FS_OUT receiver 3

2 Rev 0: LVDS Receiver input 2 (R2)
Rev 1: FS_OUT receiver 2

1 Rev 0: LVDS Receiver input 1 (R1)
Rev 1: FS_OUT receiver 1

0 Rev 0: LVDS Receiver input 0 (R0)
Rev 1: FS_OUT receiver 0

Table 944. 0x8030B00C - LVL - LVDS interrupt level register

31	4	3	2	1	0
RESERVED		R3	R2	R1	R0
0x00000000		0	0	0	0
r		rw	rw	rw	rw

31: 4 Reserved

3 Rev 0: LVDS Receiver input 3(R3)
Rev 1: FS_OUT receiver 3

2 Rev 0: LVDS Receiver input 2 (R2)
Rev 1: FS_OUT receiver 2

Table 944. 0x8030B00C - LVL - LVDS interrupt level register

1 Rev 0: LVDS Receiver input 1 (R1)
Rev 1: FS_OUT receiver 1

0 Rev 0: LVDS Receiver input 0 (R0)
Rev 1: FS_OUT receiver 0

Table 945. 0x8030B010 - CFG - LVDS configuration output register

31 6	5	4	3	2	1	0
RESERVED	T5	T4	T3	T2	T1	T0
0x00000000	0	0	0	0	0	0
r	rw	rw	rw	rw	rw	rw

31: 6 Reserved

5 LVDS transmitter output 5 (T5) - Output register bit for LVDS transmitter 5

4 LVDS transmitter output 4 (T4) - Output register bit for LVDS transmitter 4

3 LVDS transmitter output 3 (T3) - Output register bit for LVDS transmitter 3

2 LVDS transmitter output 2 (T2) - Output register bit for LVDS transmitter 2

1 LVDS transmitter output 1 (T1) - Output register bit for LVDS transmitter 1

0 LVDS transmitter output 0 (T0) - Output register bit for LVDS transmitter 0

Table 946. 0x8030B080 - STS - LVDS input register

31 4	3	2	1	0
RESERVED	R3	R2	R1	R0
0x00000000	0	0	0	0
r	rw	rw	rw	rw

31: 4 Reserved

3 Rev 0: LVDS Receiver input 3(R3)
Rev 1: FS_OUT receiver 3

2 Rev 0: LVDS Receiver input 2 (R2)
Rev 1: FS_OUT receiver 2

1 Rev 0: LVDS Receiver input 1 (R1)
Rev 1: FS_OUT receiver 1

0 Rev 0: LVDS Receiver input 0 (R0)
Rev 1: FS_OUT receiver 0

57 Errata

This chapter describes known issues with current existing silicon.

For more information contact support@gaisler.com.

57.1 Overview

Table 947 lists errata that are further described in section 57.2.

*Table 947.*Errata

ID	Device(s) Affected	Name / Description
-	Currently all devices are affected by all the erratas listed in section 57.2	-

57.2 Errata description

57.2.1 GRSCRUB: Number of frames can be scrubber limited to 2^{16} bits

ID: GR716B-ERRATA-20250901

The FCR, FRAMEID, and ERRFRMID registers are limited to 16 bits to define the number of frames to be scrubbed by GRSCRUB. As a consequence a maximum of 65535 frames can be scrubbed in sequence. The XCKU115, for instance, has a total of 98060 frames. Therefore, only partial scrubbing can be performed on XCKU115.

Workaround: The KU115 FPGA is a 3D IC using SSI (Stacked Silicon Interconnect) technology, and its XCKU115 bitstream consist of “two parts". The GRSCRUB is currently not capable of handling that. A workaround is to perform partial scrubbing of the first part, then reconfigure the IP to scrub the second part. Note that programming mode is not affected. Therefore, GRSCRUB successfully handles the XCKU115 programming.

In summary, for GR716B, an FPGA with a number of configuration frames larger than 65535 frames requires multiple partial scrubbing to cover the entire FPGA.

This errata does not affect XCKU60 devices from AMD-Xilinx.

57.2.2 GRCANFD: Cannot transfer CANFD Frame lengths 48 and 64

ID: GR716B-ERRATA-20250902

The GRCANFD in GR716B cannot able to communicate with frame lengths of 48 and 64.

Workaround: The transfer of CANFD frame lengths 12, 16, 20, 24 and 32 are functioning properly. Classical CAN unaffected.

57.2.3 Packetwire interface removed

ID: GR716B-ERRATA-20250903

A design error in the FIFOs of the Packetwire interface results in data corruption. The Packetwire core cannot be used and should be clock-gated; by default, the core is clock-gated.

Workaround: The core functionality can be implemented in software using a bit-banging technique. In addition to the main processor core, the GR716B also includes two RTAs. The RTAs have access to the GPIO lines, which can be used to realize the Packetwire interface.

57.2.4 CANFD remote boot enables UART and SPI2AHB remote boot pins

ID: GR716B-ERRATA-20250904

When CANFD remote boot is enabled, in addition to the CAN I/Os enabled in the switch matrix, the UART and SPI2AHB I/O remote boot pins are also enabled

Workaround: The affected SPI2AHB signals are GPIO[53], GPIO[54], GPIO[55], and GPIO[56]. A pull-up on GPIO[56] (Slave Select, SPIS_SLV) on the PCB should prevent spurious commands (due to noise) from being accepted by the SPI core.

The affected AHBUART signals are GPIO[49] [50]. A pull-up on GPIO[50] (AHBUART_RX) signal on the PCB should prevent spurious commands (due to noise) from being accepted by the UART core.

Frontgrade Gaisler AB
Kungsgatan 12
411 19 Göteborg
Sweden
www.frontgrade.com/gaisler
sales@gaisler.com
T: +46 31 7758650