LEON Linux

MKLINUXIMG User's Manual

USER MANUAL

RELEASED NOVEMBER 2026

Frontgrade Gaisler AB | Kungsgatan 12 | SE-411 19 | Göteborg | Sweden | frontgrade.com/gaisler

Table of Contents

1. Introduction 3
    1.1. Requirements 3
2. Installing 4
    2.1. MKLINUXIMG under Buildroot 4
3. Syntax and usage 5
4. XML syntax and usage 7
5. Support 11

1. Introduction

MKLINUXIMG is a tool that creates a RAM image with a second stage boot loader for LEON SPARC systems that sets up MMU and a way for the kernel to populate the device tree and continues to boot the Linux kernel.

Loading and running a Linux kernel is different on different architectures, on SPARC the kernel communicates with the loader, the SPARC OPENPROM, to get information about the system (frequency, number of CPUs, etc.) and for early boot operations like console UART output before the APBUART driver in Linux is initialized. The OPENPROM provides the Linux kernel a device tree with nodes containing settings. The resulting device tree can be investigated via files with binary representation under `/proc/device-tree`. One can also view the OPENPROM parameters as text files in `/proc/openprom` after mounting a special file system:

```
$ mount -t openpromfs none /proc/openprom
```

The loader is also responsible for setting up the MMU and the CPUs before entering the kernel at the virtual address of its .text section, CONFIG_PAGE_OFFSET + 0x4000 (0xF0004000 by default).

mklinuximg is a shell script which will build a OPENPROMv0 compatible PROM for a LEON Linux kernel. The ELF image created thus contains a PROM and a Linux Kernel for RAM. The RAM image can then be run directly from GRMON or as input to MKPROM2. The loader will create a device tree on the fly using AMBA Plug & Play, an optional XML file and other input parameters. Thus, MKLINUXIMG can be used both to create general kernel images that can be used by different boards/CPUs, as well as creating kernel images with highly specific device trees with board specific details.

1.1. Requirements

Some of the requirements in order to run **mklinuximg** are listed below.

- native GCC compiler
- sparc-linux toolchain in PATH, or pointed out via CROSS_COMPILE environment variable
- bash and standard UNIX commands

2. Installing

The **mklinuximg** utility can be downloaded from the Frontgrade Gaisler web site at https://download.gaisler.com/anonftp/linux/linux-2.6/kernel/mklinuximg-x.y.z.tar.bz2 [https://download.gaisler.com/anonftp/linux/linux-2.6/kernel/] .

Once the source is downloaded it is extracted and the path to the newly created **mklinuximg** command directory is added to the PATH variable. In the example below the utility is installed to `/opt`

```
$ cd /opt
$ tar -xf mklinuximg-2.0.22.tar.bz2
$ rm mklinuximg
$ ln -s mklinuximg-2.0.22 mklinuximg
$ export PATH=$PATH:/opt/mklinuximg
```

2.1. MKLINUXIMG under Buildroot

In our Buildroot release, MKLINUXIMG is downloaded and installed internally by Buildroot when it is configured to be used.

3. Syntax and usage

This section describes the syntax of invoking **mklinuximg**. After building the Linux kernel the resulting image is found in `arch/sparc/boot/image` under the build directory for Linux 3.4 or older and in `vmlinux` under the build directory from Linux 3.5 and later. This image is used as "linux-image" input into mklinuximg.

The resulting image is normally linked to start of RAM, the location of RAM can be set by the *-base* option. The PROM/loader will be located at the first 16kbytes and the Linux kernel will be at a 0x4000 offset.

The kernel command line can be set using the *-cmdline* option. After loading the RAM image produced by **mklinuximg**, the kernel command line is placed in physical memory at the address of the symbol `_bootargs_cmdline_lma`. It can there be changed by the bootloader, or e.g. by GRMON or TSIM with the **wmems** command, before execution of the image is started. After execution of the RAM image has started it will be moved and no longer possible to override.

The PROM gives the kernel the eth0 MAC address, it can be set with the *-ethmac* option.

The SMP Linux kernel needs one extra IRQ for soft-IRQ generation communication between CPUs (IPI), the default is determined by the kernel to IRQ13. However, it can be changed by adding the *"-ipi IRQ_NUM"* flag to **mklinuximg**. The -ipi option is only for SMP kernels.

The option *-amp* is a string used to create a device node property named *ampopts* for a specific AMBA core in the OPENPROM device tree. The property can be read by the device driver and a certain operation may be taken, for example only use 3 timers instead of all 5 available timers. The property is for AMP systems where resource allocation or other AMP related should change the device drivers operation. This is very seldom used, and only for AMP systems.

Below is the **mklinuximg** command output:

```
mklinuximg linux-image out-file [optional parameters]...
  linux-image   linux image, for Linux 3.4 and older linux-dir/arch/sparc/boot/image.
                and for Linux 3.5 and newer linux-dir/vmlinux.
  out-file      output image that can be uploaded using GRMON or run in TSIM.

 optional parameters:
  -base baseaddr    optional baseaddress. The default is 0x40000000.
  -cmdline string   kernel parameter string. Default is "console=ttyS0,38400".
  -freq frequency   optional frequency parameter in case if it cannot be retrieved from the Timer scaler.
  -amp string       optional string of format idx0=val0:idx1=val1:... that sets for core index
                    n property ampopts to value n. Example 0=4:1=6 will set core index 0's ampopts to 4
                    and core index 1's ampopts to 6.
  -ethmac string    set the ethernet mac address as 12 dgt hex. Default: 00007ccc0145
  -ipi irq_num      IRQ number used by Linux SMP for IPIs. May not be shared. Allowed values: 1..14
  -ioarea ioarea    GRLIB AMBA Plug&Play I/O AREA Base address. Defaults to 0xfff00000
  -uartidx index    Select UART by index for PROM Console. Default 0 (first UART0)
  -maxcpu num       Disable/Limit number of Linux CPUs 1..8 (default 8)
  -mcpu name        Use -mcpu=name instead of the default -mcpu=leon3.
                    Specify '-' to not pass on any -mcpu, getting the default of the toolchain.
  -wakecpus mask    Wake the disabled/limited CPUs in bitmask, above the ones specified by -maxcpu.
  -wakediscpus      Wake all the disabled/limited CPUs above the ones specified by -maxcpu.
  --version         Print version of mklinuximg
  -xml file         Specify an xml file that adds extra properties and nodes to the device tree.
  -flat             Create a flat device-tree.
  -grgpio-noprobe   Do not probe grgpio cores (otherwise mklinuximg probes grgpio cores for info).
  -grgpio-nbits comma-separated-list
                    Provide number of lines for grgpio cores (in scan order).
  -grgpio-imask comma-separated-list-of-bitmasks
                    Provide the imask generic (which gpio lines can generate interrupts)
                    for grgpio cores (in scan order).
  -grgpio-irqgen comma-separated-list
                    Provide the irqgen generic for grgpio cores (in scan order).
  -cflags string    Add compilation flags. Can be used multiple times.
  -sym filename     Export debug information for mklinuximg to file.
  -strip            Strip output image of symbols and debug information.
  -startupheap sz   Set startup heap size in bytes. Memory later reclaimed by the kernel.
  -promheap sz      Set prom heap size in bytes. Never reclaimed by the kernel.
  -promstack sz     Set prom and startup stack size in bytes. Never reclaimed by the kernel.
  -watchdog         Enables and adds device tree information for the GPTIMER watchdog timer.
                    If a GRWATCHDOG core is found, MKLINXUXIMG will handle the challenge/response
                    protocol during boot but not add any device tree information
  -watchdog-user-timeout timeout_sec
                    Set the user-timeout for the GPTIMER watchdog which overrides the
                    default timeout (30 seconds).
```

```
-clockgate        Enables clock gating related device tree additions for supported components.
-pinmux           Removes cores from the device tree that cannot be routed due to
                  pin multiplexing settings detected on boot.
-hrtimers         Provide device tree information for using high resolution timers in Linux.
-dbg-uart addr
                  Set address to APBUART0, for use before AMBA Plug & Play scan (default: 0x80000100).
-leon-clksrc      Provide device tree information for using the LEON up-counter as clocksource.
-irqamp5-irq      IRQ number used by the IRQAMP5 Timer. May not be shared. Allowed values: 1..14
```

4. XML syntax and usage

Using the *-xml* option of mklinuimg, the prom tree can be changed to remove core nodes, to add and remove properties to core nodes and to add subtree nodes to core nodes.

Below is the documentation on the xml format (output from **scanxml -d**):

```
The xml file format consists of the following tags:
- <matches>
- <match-core>
- <del-core>
- <match-bus>
- <corelabel>
- <add-node>
- <add-prop>
- <del-prop>
- <int>
- <corehandle>
- <string>
- <match-root>
- <nodelabel>
- <nodehandle>

<matches>
   Top level tag
Valid children:
- <match-core>
- <del-core>
- <match-bus>
- <match-root>

<match-core>
   Matches a core. There are two variants:
   1) Matching on vendor and device (and optionally index)
      A mandatory "vendor" attribute is set to match the core's AMBA Plug&Play
      vendor number and a mandatory "device" attribute is set to match the
      core's AMBA Plug&Play device number. The values for these two can be
      written wither in two possible ways, either as numbers (C style decimal,
      octal and hexadecimal notations are supported) or as the defined
      constants in include/ambapp_ids.h. If there is more than one core in the
      system with the same vendor:device ID, an optional "index" attribute can
      be set to select which one of those counting from 0 in the AMBA Plug&Play
      scan order. If the "index" attribute is not set all cores with the given
      vendor:device ID are matched.
   2) Matching on coreindex
      A mandatory "coreindex" attribute is set to the AMBA Plug&Play index
      number of the core.
Valid as a child of:
- <matches>
Valid children:
- <corelabel>
- <add-node>
- <add-prop>
- <del-prop>
- <nodelabel>

<del-core>
   Deletes a core from the prom tree. There are two variants with "vendor",
   "device" and optional "index" attributes on one hand and a "coreindex"
   attribute on the other hand, just as for the <match-core> tag.
Valid as a child of:
- <matches>
No valid children

<match-bus>
   Matches a bus. A mandatory "bus" attribute names the bus number to match
Valid as a child of:
- <matches>
Valid children:
- <add-node>

<corelabel>
   A mandatory "name" attribute sets a label for the parent core that is used
   by <corehandle> tags added elsewhere to be able to reference that parent
   core. Only one corelabel can be set per match and the label name needs to be
   unique among all corelabels and nodelabels.
Valid as a child of:
- <match-core>
No valid children

<add-node>
```

```
   Adds a child node to the prom tree node of the matched core, or parent node
   when nested withing an <add-node>. The value of the mandatory "name"
   attribute names the node. Furthermore, this adds, under this node, a string
   property with the name "name" with that same value.
Valid as a child of:
- <match-core>
- <match-bus>
- <add-node>
- <match-root>
Valid children:
- <add-node>
- <add-prop>
- <nodelabel>

<add-prop>
   Adds a property. A mandatory "name" attribute names the property. A property
   can be of three different kinds: empty, integer or string. A property
   becomes empty if it has no child tags. Child tags of type <int>, <corehandle>
   or <nodehandle> makes the property integer type, whereas child tags of type
   <string> makes it string type. Setting more than one child tag makes the
   property an array of values, but it can only have integer type or string
   type values, not both.
      If the name attribute matches an existing property of the core, the new
   value will replace the old value. This can even change the property from one
   type to another.
Valid as a child of:
- <match-core>
- <add-node>
Valid children:
- <int>
- <corehandle>
- <string>
- <nodehandle>

<del-prop>
   Deletes a property. A mandatory "name" attribute names the property that is
   to be deleted.
Valid as a child of:
- <match-core>
No valid children

<int>
   An integer value of an added property. The character data between the
   starting <int> and the ending </int> needs to be a valid integer (C style
   decimal, octal and hexadecimal notations are supported).
Valid as a child of:
- <add-prop>
No valid children

<corehandle>
   An integer value of an added property that contains the address of the prom
   tree node of the referenced core. To reference a core that is labeled using
   the <corelabel> tag, the "ref" attribute should have the value of that
   label. If the "ref" attribute is omitted, the corehandle will reference the
   core that the property is a descendant of.
Valid as a child of:
- <add-prop>
No valid children

<string>
   An string value of an added property. The character data between the
   starting <string> and the ending </string> is taken as the string value.
Valid as a child of:
- <add-prop>
No valid children

<match-root>
   Matches the root node.
Valid as a child of:
- <matches>
Valid children:
- <add-node>

<nodelabel>
   A mandatory "name" attribute sets a label for the parent node that is used
   by <nodehandle> tags added elsewhere to be able to reference that parent
   node. Only one nodelabel can be set per node and the label name needs to be
   unique among all nodelabels and corelabels.
Valid as a child of:
- <match-core>
- <add-node>
No valid children

<nodehandle>
   An integer value of an added property that contains the address of the prom
```

```
   tree node of the referenced node. To reference a node that is labeled using
   the <nodelabel> tag, the "ref" attribute should have the value of that
   label.
Valid as a child of:
- <add-prop>
No valid children
```

Below is an example file showing how to use the xml format:

```
<?xml version="1.0"?>
<matches>
  <!--
       Match core by vendor:device:index
       Strings for vendors and devices can be found in include/ambapp_ids.h or written numerically
  -->
  <match-core vendor="VENDOR_GAISLER" device="GAISLER_APBUART" index="0">
    <!-- Add single integer property value -->
    <add-prop name="intprop">
      <int>1024</int>
    </add-prop>

    <!-- Add single string property value -->
    <add-prop name="strprop">
      <string>Test string</string>
    </add-prop>

    <!-- Add array of integers -->
    <add-prop name="intarray">
      <int>17</int> <!-- Decimal notation supported -->
      <int>0xabcdef</int> <!-- Hexadecimal notation supported -->
      <int>015</int> <!-- Octal notation supported -->
      <!-- Corehandle to itself, i.e. address of prom node of the matched core -->
      <corehandle/>
      <!-- Corehandle to core with corelabel named "numero-quattro" -->
      <corehandle ref="numero-quattro"/>
      <!-- Nodehandle to node with nodelabel named "anodelabel" -->
      <nodehandle ref="anodelabel" />
    </add-prop>

    <!-- Add array of strings -->
    <add-prop name="stringarray">
      <string>String arrays</string>
      <string>are sometimes useful</string>
      <string>for multiple</string>
      <string>compatible strings</string>
    </add-prop>

     <!-- Add empty property -->
    <add-prop name="emptyprop"/>

    <!-- Add nodes with properties and subnodes  -->
    <add-node name="newnode">
      <add-prop name="newnode-p1">
        <int>1</int>
      </add-prop>
      <add-prop name="newnode-p2">
        <int>2</int>
      </add-prop>
      <add-node name="subnode1"> <!-- Child of newnode -->
        <add-prop name="sub1-p1">
          <int>3</int>
        </add-prop>
        <add-prop name="sub1-p2">
          <int>4</int>
        </add-prop>
        <add-node name="subsubnode">  <!-- Child of subnode1 -->
          <add-prop name="subsub-p1"/>
        </add-node>
      </add-node>
      <add-node name="subnode2"> <!-- Another child of newnode -->
        <add-prop name="sub2-p1">
          <string>Lorem ipsum</string>
        </add-prop>
      </add-node>
    </add-node>
    <add-node name="anothernode">
      <nodelabel name="anodelabel" />
    </add-node>

  </match-core>
```

```
  <!--
      Match core by absolute core index
  -->
  <match-core coreindex="4"  >
    <add-prop name="core-four"/>

    <!-- Label that a corehandle references to introduce an integer value of the
         address of the prom node of this core. -->
    <corelabel name="numero-quattro"/>

    <!-- Replaces the existing "version" property with a different one
         (even of a different type in this case). -->
         different type even -->
    <add-prop name="version">
      <string>
        Version is now a string!
      </string>
    </add-prop>

    <!-- Delete the description property -->
    <del-prop name="description"/>

    <add-prop name="descriptionisgone"/>
  </match-core>


  <!--
      Match core by vendor:device but matching all indices by not stating any
      index. This also serves as an example of using numeric values to match
      vendor and device (in this case VENDOR_GAISLER and GAISLER_APBUART).
  -->
  <match-core vendor="1" device="0xc">
    <add-prop name="apbuarts">
      <string>That's us</string>
    </add-prop>
  </match-core>


  <!--
      Remove the prom tree node of the first GASLER_GPIO core (if present).
      This is a match type of its own.
  -->
  <del-core vendor="VENDOR_GAISLER" device="GAISLER_GPIO" index="0"/>

  <!--
      Add a node appearing under AMBA bus 0
  -->
  <match-bus bus="0">
    <add-node name="extrabusnode">
      <add-prop name="acorehandle">
        <!-- Corehandles works from here as well -->
        <corehandle ref="numero-quattro"/>
      </add-prop>
      <add-prop name="anodehandle">
        <!-- Nodehandles works from here as well -->
        <nodehandle ref="anodelabel" />
      </add-prop>

    </add-node>
  </match-bus>

  <!--
      Put a new node right under the root node
  -->
  <match-root>
    <add-node name="extra_root_node">
      <add-prop name="extra_prop">
        <string>extra value</string>
      </add-prop>
    </add-node>
  </match-root>
</matches>
```

5. Support

For support contact the support team at support@gaisler.com.

When contacting support, please identify yourself in full, including company affiliation and site name and address. Please identify exactly what product that is used, specifying if it is an IP core (with full name of the library distribution archive file), component, software version, compiler version, operating system version, debug tool version, simulator tool version, board version, etc.

The support service is only for paying customers with a support contract.

Frontgrade Gaisler AB
Kungsgatan 12
411 19 Göteborg
Sweden
frontgrade.com/gaisler
sales@gaisler.com
T: +46 31 7758650
F: +46 31 421407

Frontgrade Gaisler AB, reserves the right to make changes to any products and services described herein at any time without notice. Consult the company or an authorized sales representative to verify that the information in this document is current before using this product. The company does not assume any responsibility or liability arising out of the application or use of any product or service described herein, except as expressly agreed to in writing by the company; nor does the purchase, lease, or use of a product or service from the company convey a license under any patent rights, copyrights, trademark rights, or any other of the intellectual rights of the company or of third parties. All information is provided as is. There is no warranty that it is correct or suitable for any purpose, neither implicit nor explicit.

Copyright © 2026 Frontgrade Gaisler AB