



## **MKLINUXIMG User's Manual**

---

*Linux RAM Loader utility for LEON, the user's manual*

*MKLINUXIMG  
Version 2.0.9  
June 2015*

---

# MKLINUXIMG

Copyright © 2015 Aeroflex Gaisler AB

---

## Table of Contents

1. Introduction .....	1
1.1. Requirements .....	1
2. Installing .....	2
3. Syntax and usage .....	3
4. XML syntax and usage .....	4
5. Support .....	8
6. Disclaimer .....	9

---

## 1. Introduction

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. One can view OPENPROM parameters in `/proc/openprom` after mounting the 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 a virtual address of `0xF0004000`.

**mklinuximg** is a shell scripts 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 and other input parameters. This way different boards/CPUs may use the same Linux kernel.

Note that U-Boot has it's own PROM implementation and thus **mklinuximg** is not used together with U-Boot, instead the kernel build target *ulmage* (or *mkimage*) is used to build a image which is readable by U-Boot.

### 1.1. Requirements

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

- native GCC compiler
  - sparc-linux toolchain in path
  - bash and standard UNIX commands
-

## 2. Installing

The **mklinuximg** utility is downloaded from Aeroflex Gaisler FTP site at <http://gaisler.com/anonftp/linux/linux-2.6/kernel/mklinuximg-x.y.z.tar.bz2> [<http://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-2.0.9.tar.bz2
$ rm mklinuximg
$ ln -s mklinuximg-2.0.9 mklinuximg
$ export PATH=$PATH:/opt/mklinuximg
```

### 3. Syntax and usage

This section describes the syntax of invoking **mklinuximg**. After building the Linux kernel the resulting image is found in `linux-2.6/arch/sparc/boot/image` for Linux 3.4 or older and `linux-2.6/vmlinux` from 3.5 and later, the 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.

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 scalar.
-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)
-wakediscpus    Wake the disabled/limited CPUs
--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).
```

## 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>

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

<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>

<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.
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>  
 Valid children:  
 - <add-node>  
 - <add-prop>

<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> and <corehandle> 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>

<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>  
 A 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

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>0xabdef</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"/>
</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">
</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-node>
</match-bus>

</matches>
```

---

## 5. Support

For support contact the Aeroflex Gaisler support team at [support@gaisler.com](mailto:support@gaisler.com).

---

## 6. Disclaimer

Aeroflex Gaisler AB, reserves the right to make changes to any products and services described herein at any time without notice. Consult Aeroflex or an authorized sales representative to verify that the information in this document is current before using this product. Aeroflex 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 Aeroflex; nor does the purchase, lease, or use of a product or service from Aeroflex convey a license under any patent rights, copyrights, trademark rights, or any other of the intellectual rights of Aeroflex or of third parties.

---