MKPROM2 User's Manual

Table of Contents

1. MKPROM2

This document describes MKPROM2 boot image generator.

1.1. Introduction

MKPROM2 is a utility program which converts a LEON RAM application image into a bootable ROM image. The resulting bootable ROM image contains system initialization code, an application loader and the RAM application itself. The RAM application is compressed with a modified LZSS algorithm, typically achieving a compression factor of 2.

The system initialization code and the application loader operates in the following steps:

- The register file of IU and FPU (if present) is initialized.
- The memory controller, UARTs and timer unit are initialized according to the specified options.
- The RAM application image stored in ROM is decompressed into RAM.
- Finally, the application is started, setting the stack pointer to the top of RAM.

MKPROM2 can create ROM images for ERC32 and LEON2/3/4 systems.

The words ROM and PROM are used in this document to denote normally non-volatile memories such as ROM, PROM, EPROM, EEPROM, Flash PROM, MRAM etc.

The word RAM is used in this document to denote normally volatile memory such as SRAM, DRAM, SDRAM, and sometimes DDR and DDR2 SDRAM.

1.2. Source code

MKPROM2 comes with full source code included. The source code is located in the `<mkprom-dir>/src` directory. To recompile mkprom issue a **make** command inside the source directory. This will compile MKPROM2 into the default location, which is `/opt/mkprom2` on Linux and `c:/opt/mkprom` on Windows. On Windows you should use the MINGW/Msys compile system.

1.3. Usage

mkprom2 is a command line utility that takes a number of options and files to encapsulate:

```
mkprom2 [options] files
```

To generate a boot-prom for a typical system, use:

```
$ mkprom2 -v -rmw -ramsize 1024 hello
```

which generates terminal output similar to

```
MKPROM v2.0.62 - boot image generator for LEON applications
Copyright Cobham Gaisler AB 2004-2017, all rights reserved.

phead0: type: 1, off: 65536, vaddr: 40000000, paddr: 40000000, fsize: 27584, msize: 28008
section: .text at 0x40000000, size 26272 bytes
Uncoded stream length: 26272 bytes
Coded stream length: 14091 bytes
Compression Ratio: 1.864
section: .rodata at 0x400066a0, size 128 bytes
Uncoded stream length: 128 bytes
Coded stream length: 38 bytes
Compression Ratio: 3.368
section: .data at 0x40006720, size 1184 bytes
Uncoded stream length: 1184 bytes
Coded stream length: 572 bytes
Compression Ratio: 2.070
Creating LEON3 boot prom: prom.out
[...]
Success!
```

When executed, the PROM loader prints a configuration message at start-up:

```
tsim> run
```

```
MKPROM2 boot loader v2.0.62
Copyright Cobham Gaisler AB - all right reserved


system clock : 50.0 MHz
baud rate : 19171 baud
prom : 512 K, (2/2) ws (r/w)
sram : 1024 K, 1 bank(s), 0/0 ws (r/w)


decompressing .text
decompressing .data
decompressing .jcr


starting hello


Hello world!
```

NOTE: It is essential that the same `-mflat`, `-qsvt` and `-msoft-float` parameters are given to mkprom2, as was used when the binary was compiled.

1.4. Creating ROM resident applications

mkprom2 can also create applications that execute from ROM, and have data and stack in RAM. This is only supported for applications developed with BCC version 1.

A ROM resident application is created in two steps:

- Compile the application into on or more object file, but do not link:

  ```
  sparc-elf-gcc -msoft-float -c -g -O2 hello.c
  ```
- Create the ROM image with mkprom2 and the **-romres** option, listing all object files on the command line:

  ```
  mkprom2 -romres -freq 40 -rmw hello.o -msoft-float
  ```

A ROM resident application has its code (.text segment) in ROM, and data (.data and .bss) in RAM. At startup, the .data segment is copied from the ROM to the RAM, and the .bss segment is cleared. A ROM resident application is linked to start from address 0x0. The data segment is by default linked to 0x40000000, but can be changed by giving the -Tdata=<address> option of gcc to mkprom2. Note that if no FPU is present, the -msoft-float option must also be given to mkprom2 in this case since it is needed during the final linking.

When debugging ROM resident applications with GRMON or gdb, hardware breakpoints (hbreak command) have to be used. ROM resident applications cannot be compressed, and thus the **-nocomp** option is implied.

When generating a ROM resident image, a symbol image with name <ofile>.sym is created that can be used for debugging. The actual ROM output image <ofile> does not contain symbol information.

As mentioned earlier, ROM resident MKPROM2 images are supported only for application developed with BCC version 1.0.x. The reason for this is because MKPROM2 utilises knowledge about the application run-time and patches itself into the initializations. Such shared knowledge is not available for other run-times.

1.5. Internals

mkprom2 is delivered with source code. mkprom2 is compiled from source file mkprom.c. mkprom2 creates a PROM image through the following steps:

- Parse option switches
- Calculate the register initialization values from the switches.
- Read in elf-format object files and extract load location and section data from it.
- Dump register values and sections data into a file called dump.s. You can preserve and read this file using the -dump option.
- Use the crosscompile toolchain to compile dump.s and link this file against the boot-loader object files. You can see the command that is issued by adding the -v (-V) switch to mkprom2.

The MKPROM2 binary distribution includes precompiled object code which is used when linking the boot loader. It is available in the installation subdirectory named `lib/`. The object code has been compiled with workarounds enabled for UT699. In addition, workarounds for the following technical notes have been taken into account for the included object code:

- GRLIB-TN-0009
- GRLIB-TN-0010 (MMU is not enabled when MKPROM boot loader executes))
- GRLIB-TN-0011 (MMU is not enabled when MKPROM boot loader executes))
- GRLIB-TN-0012 (FPU operations are not performed by MKPROM boot loader)
- GRLIB-TN-0013 (FPU operations are not performed by MKPROM boot loader)

1.6. MKPROM2 general options

In GRMON, the command **info mkprom2** is available for extracting mkprom2 parameters for memory controller, timer, uart and interrupt controller. The extracted parameters can be used as a starting point for mkprom2.

```
grmon2> info mkprom2
  Mkprom2 switches:
  -leon3 -freq 48 -rmw -ramsize 8192 -sdram 128 -sdrambanks 2 -trfc 83
```

For hardware without an FPU, the `-msoft-float` has to be given to mkprom2. Note the FPU registers will be cleared regardless of the `-msoft-float` flag if an FPU is present, however the FPU will be turned off when entering the application if `-msoft-float` has been given.

Table 1.1. Linking options

Option	Description
-msoft-float	Link for application not using FPU.
-mflat	Link for application using the GCC 3.4.4 flat register window model.
-qsvt	Link application using the single vector trapping model.

Table 1.2. General options

Option	Description
-leon2	Generate a LEON2 executable.
-leon3	Generate a LEON3 executable. This is the default.
-erc32	Generate a ERC32 executable.
-agga4	Generate an AGGA4 executable.
-baud baudrate	Set rate of UART A to baudrate. Default value is 19200.
-bdinit	The user can optionally define three hook routines, `bdinit0()`, `bdinit1()` and `bdinit2()`, which are called by CPU0 during the boot process. `bdinit0()` is called before and `bdinit1()` is called after the LEON peripheral registers have been initialized but before the memory has been cleared. `bdinit2()` is called after the memory has been initialized but before the application is loaded. Note that when `bdinit0()` and `bdinit1()` is called, the stack has not been setup meaning that `bdinit0()` and `bdinit1()` must be leaf routines and not use RAM. `bdinit0()` is called before FPU registers are initialized. When the option `-bdinit` is used, a file named `bdinit.o` must exist in the current directory, containing the three routines `bdinit0()`, `bdinit1()` and `bdinit2()`.

Option	Description
	One additional function, named `bdcpuinit0()`, can also be defined. This function will be called by all processors at the same boot stage as `bdinit0()` is called by CPU0. The execution environment is the same as for `bdinit0()`. However, resources configured by CPU0 may now be available for the other processors. CPU0 calls `bdcpuinit0()` just before `bdinit0()`.
-ccprefix <prefix>	On startup mkprom2 will search for sparc-elf-gcc, sparc-gaisler-elf-gcc, sparc-rtems-gcc and sparc-linux-gcc. The first found will be used to create the PROM image. The -ccprefix option lets you override the compiler prefix. Example: **-ccprefix sparc-elf**
-dump	The intermediate assembly code with the compressed application and the LEON register values is put in dump.s (only for debugging of mkprom2). This switch is very useful to verify the calculated initialization values of the registers.
-dsubaud rate	Sets the baudrate of the debug support unit (DSU). Default: 0
-duart addr	Sets the address of the debug uart registers. Default: 0x80000700
-ecos	Use eCOS realtime library options
-edac	Clear all memory specified by the memory parameters and enable EDAC.
-edac-clean [bank0-addr] [bank0-size] [bank1-addr] [bank1-size]	Explicitly specify the 2 banks [[bank0-addr],[bank0-size]] and [[bank1-addr],[bank1-size]] that should be cleared at bootup to prepare for EDAC enable. If only one bank should be cleared specify 0 as size. The switch -edac has to be given also.
-entry addr	Sets the application.s start address (after decompression). Default: the ELF start address
-freq system_clock	Defines the system clock frequency in MHz. This value is used to calculate the divider value for the baud rate generator and the real-time clock. Default is 50 for LEON.
-noinit	Suppress all code which initializes on-chip peripherals such as UARTs, timers and memory controllers. This option requires -bdinit to add custom initialisation code, or the boot process will fail.
-nomsg	Suppress the boot message.
-romres	Create a ROM resident image. See Section 1.4.
-nocomp	Do not compress application. Decreases loading time on the expense of ROM size.
-o outfile	Put the resulting image in outfile, rather than prom.out (default).
-rstaddr addr	Sets the PROM start address. In case of an execute-in-prom configuration addr is limited to 0x0-0x20000000. Default: 0x0
-stack addr	Sets the initial stack pointer to addr. If not specified, the stack starts at top-of-ram.
-sparcleon0	Normally objects with load address 0 will force MKPROM2 into execute-from-rom mode. To avoid this the option -sparcleon0 can be specified. This option can be used if the application was linked with -msparcleon0.
-sparcleon0rom	Use this switch to force creation of a execute-from-rom image for applications with ram-load addess 0.
-v	Be verbose; reports compression statistics and compile commands
-V	Very verbose output (as opposed to -v, which is just verbose)

Option	Description
input_files	The input files must be in aout or elf32 format. If more than one file is specified, all files are loaded by the loader and control is transferred to the first segment of the first file.

1.7. LEON2/3 memory controllers options

Table 1.3. EDAC output file options

Option	Description
-bch8	Generate an additional output file <output>.bch8 with a .bch section that contains the EDAC BCH checksums used with 8-bit wide PROM memories. 4/5 of the PROM size is for user data and 1/5 for EDAC BCH checksums. The .bch section is positioned at the end of the PROM (growing in reverse address order). The total PROM size is specified with the -romsize option. The -romcs option must be 1 (default). The -romwidth option must be 8. The 4/5 EDAC scheme is supported by FTMCTRL (e.g. UT699, LEON3FT-RTAX CID-3 through CID-8) and LEON2FT MCTRL (e.g. AT697F, AT9713E/F). Note that only one PROM bank is supported.
-bch8q	Generate an additional output file <output>.bch8q with a .bch section that contains the EDAC BCH checksums used with 8-bit wide PROM memories. 3/4 of the PROM size is for user data and 1/4 for EDAC BCH checksums. The .bch section is positioned at 3/4 of the total PROM (growing in forward address order). The total PROM size is specified with the -romsize option. The -romcs option must be 1, 2, 4 or 8. The -romwidth option must be 8. The 3/4 EDAC scheme is supported by FTSRCTRL (e.g. LEON3FT-RTAX CID-1 through CID-2) for multiple PROM banks, with the EDAC size matching the total PROM size specified with the -romsize option. The 3/4 EDAC scheme is also supported by the old FTMCTRL and the old LEON2FT MCTRL (e.g. AT697E), but only for one PROM bank, i.e. -romcs option must be 1.

The options in Table 1.4 are used by the MKPROM2 tool to configure the memory controller. MKPROM2 uses these options to calculate suitable values which are used by the boot code to initialize the memory controller. The options in Table 1.4 are interpreted by MKPROM2 at the time when the boot image is generated.

Table 1.4. Memory controller configuration register options

Option	Description
-cas delay	Set the SDRAM CAS delay. Allowed values are 2 and 3 (default is 2).
-col bits	Set the number of SDRAM column address bits. Allowed values are 8 - 11 (default is 9).
-nosram	Disables the static SRAM and maps SDRAM at address 0x40000000.
-ramcs chip_selects	Set the number of SRAM banks to chip_selects. Default is 1.
-ramrws ws	Sets the SRAM read wait states -ramws value
-ramsize size	Defines the total available RAM in kBytes. Used to initialize the in the memory configuration register(s). The default value is 2048 (2 MByte).
-ramwidth width	Sets the SRAM bit width to 8, 16, 32, or 39 bits. Default: 32 bits
-ramws ws	Set the number of waitstates during SRAM reads and writes to ws. Default is 0.
-ramwws ws	Sets the SRAM write wait states -ramws value
-refresh delay	Set the SDRAM refresh period (in us). Default is 7.8 us, although many SDRAM actually use 15.6 us. -romcs chip_selects Set the number of ROM

Option	Description
	banks to chip_selects. Default is 1, possible values are 1, 2, 4 and 8. This options is used by -bch8q where it becomes mcfg1.ebsz.
-romsize kb	Sets the total size of the PROM in kByte. Default: 0x80000
-rmw	Perform read-modify-write cycles during byte and halfword writes.
-romwidth width	Sets the PROM bit width to 8, 16, 32, or 39 bits. Default: 8 bits
-romws ws	Set the number of PROM waitstates during read and write to ws. Default is 2.
-sdram size	The total amount of attached SDRAM in MByte. To use -sdram in the calculation of the stack also specify -nosram. 0 by default
-sdrambanks num_banks	Set the number of populated SDRAM banks (default is 1).
-trfc delay	Set the SDRAM tRFC parameter (in ns). Default is 66 ns.

-trp delay	Set the SDRAM tRP parameter (in ns). Delay defaults to 20 ns. If two system clock periods is shorter than the given tRP value the MCFG2.TRP bit is set to increase to 3 system clocks.
	
	The formula used is:
	`IF (2*1E9/FREQ_HZ) < delay THEN`
	`  set MCFG2.TRP=1`
	`ELSE`
	`  set MCFG2.TRP=0`
	`END IF`
	
	Note that the system clock is used in the calculation, if the SDRAM controller is clocked on a different clock frequency the `-mcfg{1|2|3}` parameters should be used.

-iowidth width	Sets the IO bit width to 8, 16, or 32 bits. Default: 32 bits
-iows ws	Sets the IO wait states. Default: 7

An optional way to specify the initialization values for the memory controller configuration registers is to use the options in Table 1.5. The `-mcfg{1|2|3}` options have priority over the parameters in Table 1.4 for memory controller initialization during boot. However, the options in Table 1.4 are still used for calculating the initial stack pointer, RAM wash size and EDAC allocation when using 8-bit SRAM bus.

It is recommended *NOT* to use the `-mcfg{1|2|3}` options at all. Instead go through the memory related options in Table 1.4 and match them with the target system. This also provides a straight forward way to enable/disable individual options by only changing the human readable command line switch.

If however `-mcfg{1|2|3}` options must be used, make sure that to use all of them and do not include any other memory controller switches that influence that same registers.

NOTE: To get a human readable printout of the current memory controller configuration from GRMON, use:

```
grmon2> info sys mctrl0
[...]
grmon2> info reg -v mctrl0
[...]
```

Table 1.5. Memory controller configuration direct options

Option	Description
-mcfg1 <hex>	Specify the mcfg1 register directly.
-mcfg2 <hex>	Specify the mcfg2 register directly.
-mcfg3 <hex>	Specify the mcfg3 register directly.

1.8. LEON3 options

Currently the following IP cores are detected and initialized using plug and play: DDR2SPA, DDRSPA, SDCTR, IRQMP, APBUART, GPTIMER, GRTIMER, MCTRL, FTMCTRL, FTSRCTRL, FTAHBRAM.

Table 1.6. MKPROM2 options for LEON3

Option	Description
-gpt addr	Sets the address of the timer unit regs. Default: 0x80000300
-irqmp addr	Sets the address of the IRQMP controller regs. Default: 0x80000200. This option is only useful when -nopnp is specified.
-memc addr	Sets the address of the memory controller regs. Default:0x80000000
-mp	Enable multi CPU support. Mutliple stacks, entry points, UARTs etc.
-mpentry ncpu entry1 entry2 .. entryN	Defines the entry points of N CPUs in a multiprocessor system where different entry points are needed, this is typically the case for RTEMS.
-mpirqsel cpu val	In a multiprocessor system specify the value of the TCSELn field of the IRQAMP irq controller's Interrupt Controller Select Register for <cpu>. -mpirqsel can be called several times for each CPU.
-mpstack ncpu stack1 stack2 .. stackN	In a multiprocessor system it may be reqiured to use different stack areas for the different CPUs. This option enables the user to set the stack for each CPU.
-mpstart val	In a multiprocessor system specify a value to write into the MPIRQ status register.
-mpuart nuart UART[1] UART[2] .. UART[N]	Defines the base register address of the first N UARTs. This option is only possible with -nopnp. All uarts defined are initialized with the baudrate given by the -baud option.
-uart addr	Sets the address of the UART base used to output boot messages. Default: 0x80000100
-dsustart addr	Set the DSU start address used by -dsutrace. Default: 0x90000000
-dsutrace	Switches on instruction trace buffer on startup by writing the DSU registers. Default: disabled
-dsubreak	Switches on DSU control regiser.s BZ bit. Default value written into DSU control register: 0xcf
-nopnp	Switches off plug and play initialization. In this case only mctrl, uart and timer are initialized. Addresses can be specified with -memc, -gpt and -uart or left default. If -ddr2spa_cfg[1|3|4] is supplied, instead of (FT)MCtrl, DDR2Ctrl initialization is performed. Default: pnp enabled
-pnp addr	Define the AMBA plug and play configuration area address where the AHB slave membars are located. Default: 0xfffff800

To create a multiprocessor AMP image the options -mp, -mpstack, -mpentry, -mpstart and -mpirqsel can be given. First the user would create different images linked to different RAM addresses. Using the -mpentry option the entry address of each processor can be specified. Processor 0 will handle the setup and decompression, thereafter starting the other processors. The -mpstart option specifies which processors to start. The -mpstack will specify the end-of-stack for each processor. The convention in software is that [bss-end,end-of-stack] defines the available memory region for each processor. Finally the IRQAMP controller can be configured using the -mpirqsel option. Below is an example of a AMP system with 2 processors. One RTEMS image running at 0x0, the other at 0x40000000.

```
$mkprom2 \
    -mp \
    -mpstart 0x3 \
    -mpirqsel 0 0 \
    -mpirqsel 1 1 \
    -mpuart 2 0xF0000000 0xf0001000 \
    -mpstack 2 0x3fffff00 0x400fff00 \
```

```
-mpentry 2 0x0 0x40000000 \
rtems-tasks-0x00000000 rtems-tasks-0x40000000 -o amp.prom
```

1.9. DDR/DDR2 controller options

Table 1.7. MKPROM2 options for DDR/DDR2 controller

Option	Description
-ddrram size	Set memory bank size in MByte. Supported values are: 8-1024. Default: 64
-ddrbanks count	Set number of banks. Default: 1
-ddrfreq freq	Set DDR frequency in MHz. Default: 90.
-ddrrefresh num	Set the DDR refresh period in us. Default is 7.8 us.
-ddrcol size	Set columns size. Supported values are: 512, 1024, 2048, 4096. Default: 1024
-ddr2spa_cfg1 hex	Alternatively specify cfg1 of the DDR2 controller as hex.
-ddr2spa_cfg3 hex	Alternatively specify cfg3 of the DDR2 controller as hex.
-ddr2spa_cfg4 hex	Optionally specify cfg4 of the DDR2 controller as hex.
-ddrspa_cfg1 hex	Alternatively specify cfg1 of the DDR controller as hex.

1.10. SDCTRL64/FTSDCTRL64 controller options

Table 1.8. MKPROM2 options for SDCTRL64/FTSDCTRL64 controller

Option	Description
-ftsdctrl64_cfg1 [val]	Specify the cfg1 register of the SDCTRL64/FTSDCTRL64 controller (SDRAM Configuration register) .
-ftsdctrl64_cfg2 [val]	Specify the cfg2 register of the SDCTRL64/FTSDCTRL64 controller (SDRAM Power-Saving configuration register).

1.11. FTAHBRAM controller options

Table 1.9. MKPROM2 options for FTAHBRAM controller

Option	Description
-ftahbram_edac	If specified the first FTAHBRAM controller's EDAC is enabled. If not specified the first FTAHBRAM controller's configuration register will be written a zero, disabling on-chip memory EDAC. Note that memory is not washed, that can either be done manually from a `bdinit` function or using on of the optional -edac_clean regions.

1.12. SDCTRL controller options

Table 1.10. MKPROM2 options for SDCTRL controller

Option	Description
-sdcfg1 [val]	Specify the cfg1 register of the SDCTRL controller (SDRAM Configuration register) .

1.13. SPI memory controller options

Table 1.11. MKPROM2 options for SPI memory controller

Option	Description
-spimeas	Enables the SPI memory controller alternate scaler early in the boot process.

1.14. Custom controllers

If the target LEON3 system contains a custom controller, the initialization of the controller must be made through the bdinit1 function. Below is an example of a suitable bdinit.c file. The file should be compiled with .sparc-elf-gcc -O2 -c -msoft-float., and mkprom2 should be run with the -bdinit option.

```
void bdinit1() {
<.. your init code here ..>
}


void bdinit2 () {}
```

1.15. Timer initialization

This section describes the default initialization of GPTIMER and GRTIMER cores on LEON3 systems with AMBA plug and play.

- Only timer cores (GPTIMER/GRTIMER) on the first APB bus are initialized by mkprom2.
- GPTIMER cores are initialized before GRTIMER cores.
- The timer core prescaler reload value is set such that it underflows once every microsecond. The -freq parameter is used to calculate the prescaler value.
- The first subtimer of each timer core is configured with reload value 0xffffffff. Its control register is then initialized such that the subtimer is loaded, enabled and set in restart mode.
- The last subtimer of the first timer core (watchdog) is configured with reload value 300000000 (5 minutes). Its control register is not initialized (reset value remains).
- All other subtimers are initialized with 0 in their counter value registers, reload value registers and control registers.

This default timer initialization can be overridden by `bdinit1()` as described in this document.

2. Support

For support contact the Cobham Gaisler 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.

Cobham Gaisler AB
Kungsgatan 12
411 19 Gothenburg
Sweden
www.cobham.com/gaisler
sales@gaisler.com
T: +46 31 7758650
F: +46 31 421407