AEROFLEX GAISLER

MKPROM2 User's Manual

1 MKPROM2 BOOT-PROM BUILDER

1.1 Introduction

mkprom2 is a utility program to create boot-images for programs compiled with the BCC or RTEMS cross-compiler. It encapsulates the application in a loader suitable to be placed in a boot PROM. The application is compressed with a modified LZSS algorithm, typically achieving a compression factor of 2. The boot loader operates in the following steps:

1. The register files of IU and FPU (if present) are initialized.
2. The memory controller, UARTs and timer unit are initialized according to the specified options.
3. The application is decompressed and copied into RAM.
4. Finally, the application is started, setting the stack pointer to the top of RAM.

The created boot-prom will run on both ERC32 (**-erc32**), LEON2 (**-leon2**) or LEON3 systems.

Note that the word PROM is used in this document to denote normally non-volatile memory such as ROM, PROM, EPROM, EEPROM, Flash PROM etc.

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

1.2 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, do:

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

LEON MKPROM prom builder for BCC, ECOS, RTEMS and ThreadX v1.0.0
Copyright Gaisler Research 2004-2007, all rights reserved.

loading hello:
section: .text at 0x40000000, size 15744 bytes
Uncoded stream length: 15744 bytes
Coded stream length: 7794 bytes
Compression Ratio: 2.020
section: .data at 0x40003d80, size 2016 bytes
Uncoded stream length: 2016 bytes
Coded stream length: 691 bytes
Compression Ratio: 2.918
section: .jcr at 0x400045c4, size 4 bytes
Uncoded stream length: 4 bytes
Coded stream length: 4 bytes
Compression Ratio: 1.000

creating LEON boot prom: prom.out
```

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

```
tsim> run

  MkProm2 LEON boot loader v1.2
  Copyright Gaisler Research - 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. Any miss-match will produce a faulty PROM image.

1.3 Creating applications that run in PROM

mkprom2 can also create applications that run in PROM, and have data and stack in RAM. A PROM application is created in two steps:

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

```
sparc-elf-gcc -msoft-float -c -g -O2 hello.c
```

2. Create final PROM image with *mkprom2*, listing all object files on the command line:

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

A PROM application has it code (*.text* segment) in PROM, and data (*.data* & *.bss*) in RAM. At start-up, the *.data* segment is copied from the PROM to the RAM, and the *.bss* segment is cleared. A PROM 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 PROM applications with GRMON or gdb, only hardware breakpoints (*hbreak* command) can be used. Applications running from PROM cannot be compressed.

1.4 Internals

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

1. Parse option switches
2. Calculate the register initialization values from the switches.
3. Read in elf-format object files and extract load location and section data from it.
4. Dump register values and sections data into a file called *dump.s*. You can preserve and read this file using the **-dump** option.
5. 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*.

2 MKPROM2 GENERAL OPTIONS

The options *-msoft-float*, *-mv8* (*-mcpu=v8*) have to be given to *mkrom2* according to the hardware setting. For hardware without a FPU the *-msoft-float* has to be given, for hardware with a [s|u]mul/[s|u]div instruction support the *-mv8* option can be given.

Linking options:

-msoft-float

Compile for hardware without a FPU.

-mv8

Compile for hardware that supports the [s|u]mul/[s|u]div instructions.

-mflat

Compile for hardware with flat register window model.

-qsvt

Compile for hardware with single vector traping . See also *-checksvt* option.

General options:

-leon2

Generate a LEON2 executable.

-leon3

Generate a LEON3 executable. This is the default.

-erc32

Generate a ERC32 executable.

-baud *baudrate*

Set rate of UART A to *baudrate*. Default value is 19200.

-bdinit

The user can optionally call two user-defined routines, *bdinit1()* and *bdinit2()*, during the boot process. *bdinit1()* is called after the LEON 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 *bdinit1()* is called, the stack has not been setup meaning that *bdinit1()* must be a leaf routine and not allocate any stack space (no local variables). When the switch **-bdinit** is used, a file called *bdinit.o* must exist in the current directory, containing the two routines.

-ccprefix <*prefix*>

On startup *mkprom2* will search for sparc-elf-gcc, sparc-rtems-gcc and sparc-linux-gcc. Whichever is found first will be used to create the PROM image. the *-ccprefix* option lets you state a prefix directly, i.e. *-ccprefix sparc-elf*

-checksvt

When *-qsvt is* used *-checksvt* can be given. *-checksvt* will prepend a *%tbr* initialization to the svt dispatch to avoid 'X' exceptions in vhdl simulation.

-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.

-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.

-nocomp

Don't compress application. Decreases loading time on the expense of PROM size.

-o *outfile*

Put the resulting image in *outfile*, rather than *prom.out* (default).

-rstaddr *addr*

Sets the PROM start address of the image.

-stack *addr*

Sets the initial stack pointer to *addr*. If not specified, the stack starts at top-of-ram.

-v

Be verbose; reports compression statistics and compile commands

-rstaddr *addr*

Sets the PROM start address. In case of an execute-in-prom configuration *addr* is limited to 0x0-0x20000000. Default: 0x0

-V

Very verbose output (as opposed to **-v**, which is just verbose)

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.

2.1 mkprom2 options for the LEON2/3 memory controllers options

-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.

-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).

-memcfg1 <*hex*>

Specify the memcfg1 register directly.

-memcfg2 <*hex*>

Specify the memcfg2 register directly.

-memcfg3 <*hex*>

Specify the memcfg3 register directly.

-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 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. 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). Default is 20 ns.

-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

2.2 mkprom2 options for LEON3

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

-gpt *addr*

LEON3: Sets the address of the timer unit regs. Default: 0x80000300

-irqmp *addr*

LEON3: Sets the address of the IRQMP controller regs. Default: 0x80000200. This option is only useful when -nopnp is specified.

-memc *addr*

LEON3: 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.

-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.

-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*

LEON3: Sets the address of the UART regs 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. Default: pnp enabled

-pnp *addr*

Define the AMBA plug and play configuration area address where the AHB slave membars are located. Default: 0xfffff800

2.3 mkprom2 options for LEON3 DDR/DDR2 controller

-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.

2.4 mkprom2 options for LEON3 SPI memory controller

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

2.5 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 () {}
```

Aeroflex Gaisler AB
Kungsgatan 12
411 19 Göteborg
Sweden

tel +46 31 7758650
fax +46 31 421407
sales@gaisler.com
www.aeroflex.com/gaisler