FRONTGRADE

APPLICATION NOTE

UT699-AN4-01

GRMON Scripts for the LEON 3FT

5/11/2011
Version #: 1.0.0

Product Name:	Manufacturer Part Number	SMD #	Device Type	Internal PIC
LEON3FT	UT699	5962-08228	ALL	WG07

Table 1: Cross Reference of Applicable Products

1.0 Overview

GRMON is the debugger monitor for the LEON 3FT processor and for SOC (System on Chip) designs based on the GRLIB IP library. GRMON includes the following functions:

- Read/write access to all system registers and memory
- Built-in disassembler and trace buffer management
- Downloading and execution of LEON applications
- Breakpoint and watchpoint management
- Remote connection to GNU debugger (GDB)
- Support for USB, JTAG, RS232, PCI, and SpaceWire debug links

An Evaluation and professional version of GRMON is currently supported on Linux and Windows hosts. GRMON is available at frontgrade.com. For installation of GRMON, please referrer to the GRMON Manual.

2.0 GRMON Scripts.

One can use batch files or scripts when using GRMON; they are inputs when starting GRMON, or run in GRMON at the command line interface:

Staring GRMON:

-c batch_file (Run the commands in the batch file at start-up).

GRMON Command line interface:

batch batch_file (Execute a batch file of GRMON commands).

The following batch file performs the following commands:

1. Read**(mem)**/Write**(wmem)** to a memory location in SRAM
2. **wash** SRAM/SDRAM
3. **load** a hello world program
4. **disassemble** the code at 0x4000 0000
5. **break** at the function **stop()**
6. **disassemble stop()**
7. **continue** to run the program
8. unlock the **flash** memory
9. erase the **flash** memory
10. **load** a prom image in to PROM.

    (See GRMON user manual for detail descriptions of commands)

Load_dis_flash.bat:

```
mem 0x40000000
wmem 0x40000000 0x1234abcd mem 0x40000000
wash
load hello disassemble 0x40000000
break stop
run
echo BREAK AT STOP FUNCTION
disassemble stop
cont
flash unlock all
flash erase 0x0 0x100000
flash load hello.prom
flash lock all
echo
echo Please Push Reset to run the PROM image
echo
echo GRMON will now quit.....
echo
quit
```

hello.c: (The program that is compiled using BCC, and downloaded and ran using GRMON)

```
#include <stdio.h>
#include <stdlib.h>
int stop(void)
{
        printf("GO!!!\n");
        return 0;
}
int main(void)
{
        printf("Hello World\n");
        printf("STOP!\n");
        stop();
        return 0;
}
```

Makefile: (Used to create the executable and the PROM image).

```
all:
        sparc-elf-gcc hello.c -o hello
        mkprom2 –v -ramws 2 -romws 5 -baud 38400 -freq 66 hello -o hello.prom
```

Output of Makefile:

```
$ make all
sparc-elf-gcc hello.c -o hello
mkprom2 -v -ramws 2 -romws 5 -baud 38400 -freq 66 hello -o hello.prom

LEON2/3/ERC32 MKPROM prom builder for BCC, ECOS, RTEMS and ThreadX v2.0.38
Copyright Frontgrade Research 2004-2007, all rights reserved.

phead0: type: 1, off: 65536, vaddr: 40000000, paddr: 40000000, fsize: 24720, msize: 25752
phead1: type: 1, off: 91288, vaddr: 40006498, paddr: 40006498, fsize: 0, msize:4
section: .text at 0x40000000, size 21808 bytes
Uncoded stream length: 21808 bytes
Coded stream length: 11537 bytes
Compression Ratio: 1.890
section: .data at 0x40005530, size 2912 bytes
Uncoded stream length: 2912 bytes
Coded stream length: 829 bytes
Compression Ratio: 3.513

creating LEON3 boot prom: hello.prom
Searching for compiler to use (sparc-elf, sparc-rtems or sparc-linux):
sparc-elf-gcc (BCC 4.4.2 release 1.0.36b) 4.4.2
Copyright (C) 2009 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO warranty; not even for MERCHANTABILITY
or FITNESS FOR A PARTICULAR PURPOSE.

sparc-elf-gcc.exe -O2 -g -N -Tc:/opt/mkprom2/linkprom -Ttext=0x0 c:/opt/mkprom2 /promcore.o
c:/opt/mkprom2/prominit.o c:/opt/mkprom2/prominit_leon3.o c:/opt/mkp rom2/promcrt0.o
c:/opt/mkprom2/promload.o c:/opt/mkprom2/promdecomp.o -nostdlib c:/opt/mkprom2/prombdinit.o dump.s -o
hello.prom multidir:
```

Output from GRMON to HyperTerminal:

```
Hello World
STOP!
GO!!!
```

Output after reset in HyperTerminal:

```
MkProm2 boot loader v2.0
Copyright Frontgrade Research - all rights reserved
system clock : 66.0 MHz
baud rate : 38372 baud
prom : 512 K, (5/5) ws (r/w)
sram : 2048 K, 1 bank(s), 2/2 ws (r/w)

decompressing .text to 0x40000000
decompressing .data to 0x40005530

starting hello

Hello World
STOP!
GO!!!
```

GRMON Output:

The following shows the batch file load_dis_flash.bat being executing in GRMON.

```
$ grmon -xilusb -c load_dis_flash.bat
GRMON LEON debug monitor v1.1.47 professional version
Copyright (C) 2004-2010 Frontgrade - all rights reserved.
For latest updates, go to frontgrade.com
Comments or bug-reports to https://frontgrade.com/contact-us

Try to open libusb filter driver (install from http://libusb-win32.sourceforge.net)
Xilinx cable: Cable type/rev : 0x3
JTAG chain: UT699A

Device ID: : 0x699
GRLIB build version: 2564

initializing ......................
detected frequency: 66 MHz
```

Component	Vendor
LEON3FT SPARC V8 Processor	Frontgrade Research
AHB Debug UART	Frontgrade Research
AHB Debug JTAG TAP	Frontgrade Research
Fast 32-bit PCI Bridge	Frontgrade Research
PCI/AHB DMA controller	Frontgrade Research
GR Ethernet MAC	Frontgrade Research
GRSPW Spacewire Link	Frontgrade Research
GRSPW Spacewire Link	Frontgrade Research
GRSPW Spacewire Link	Frontgrade Research
GRSPW Spacewire Link	Frontgrade Research
FT Memory Controller	Frontgrade Research
AHB/APB Bridge	Frontgrade Research
LEON3 Debug Support Unit	Frontgrade Research
OC CAN controller	Frontgrade Research
Generic APB UART	Frontgrade Research
Multi-processor Interrupt Ctrl	Frontgrade Research
Modular Timer Unit	Frontgrade Research
Clock gating unit	Frontgrade Research
PCI Arbiter	European Space Agency
General purpose I/O port	Frontgrade Research
AHB status register	Frontgrade Research

Use command 'info sys' to print a detailed report of attached cores

```
mem 0x40000000
        40000000 12345678 00000000 81c123ec 01000000 .4Vx.....•#•....
        40000010 a1480000 a7500000 108011ef ac102001 íH..°P.....•¼. .
        40000020 91d02000 01000000 01000000 01000000 æ• .............
        40000030 91d02000 01000000 01000000 01000000 æ• .............

wmem 0x40000000 0x1234abcd
mem 0x40000000
40000000 1234abcd 00000000 81c123ec 01000000 .4½ .....•#•....
40000010 a1480000 a7500000 108011ef ac102001 íH..°P.....•¼. .
40000020 91d02000 01000000 01000000 01000000 æ• .............
40000030 91d02000 01000000 01000000 01000000 æ• .............

wash
        clearing 8192 kbyte SRAM: 40000000 - 40800000
        clearing 131072 kbyte SDRAM: 60000000 - 68000000

load hello
        section: .text at 0x40000000, size 21808 bytes
        section: .data at 0x40005530, size 2912 bytes
        total size: 24720 bytes (633.8 kbit/s)
        read 163 symbols entry point: 0x40000000

disassemble 0x40000000
        40000000 88100000 clr %g4
        40000004 09100011 sethi %hi(0x40004400), %g4
        40000008 81c123ec jmp %g4 + 0x3ec
        4000000c 01000000 nop
        40000010 a1480000 mov %psr, %10
        40000014 a7500000 mov %wim, %13
        40000018 108011ef ba 0x400047d4
        4000001c ac102001 mov 1, %l6
        40000020 91d02000 ta 0x0
        40000024 01000000 nop
        40000028 01000000 nop
        4000002c 01000000 nop
        40000030 91d02000 ta 0x0
        40000034 01000000 nop
        40000038 01000000 nop
        4000003c 01000000 nop
break stop
run

echo BREAK AT STOP FUNCTION
        BREAK AT STOP FUNCTION
```

```
disassemble stop
        400011a4 03100015 sethi %hi(0x40005400), %g1
        400011a8 901060d0 or %g1, 0xd0, %o0
        400011ac 40000036 call 0x40001284
        400011b0 01000000 nop
        400011b4 82102000 mov 0, %g1
        400011b8 b0100001 mov %g1, %i0
        400011bc 81e80000 restore
        400011c0 81c3e008 retl
        400011c4 01000000 nop
        400011c8 9de3bfa0 save %sp, -96, %sp


        400011cc 03100015 sethi %hi(0x40005400), %g1
        400011d0 901060d8 or %g1, 0xd8, %o0
        400011d4 4000002c call 0x40001284
        400011d8 01000000 nop
        400011dc 03100015 sethi %hi(0x40005400), %g1
        400011e0 901060e8 or %g1, 0xe8, %o0


cont
flash unlock all
        flash erase 0x0 0x100000
        Erase in progress
        Block @ 0x00000000 : code = 0x00800080 OK
        Block @ 0x00040000 : code = 0x00800080 OK
        Block @ 0x00080000 : code = 0x00800080 OK
        Block @ 0x000c0000 : code = 0x00800080 OK
        Block @ 0x00100000 : code = 0x00800080 OK
        Erase complete
flash load hello.prom
        section: .text at 0x0, size 18176 bytes
        total size: 18176 bytes (70.5 kbit/s)
        read 136 symbols
        entry point: 0x00000000

flash lock all
echo
echo Please Push Reset to run the PROM image
echo
echo GRMON will now quit.....
echo
        Please Push Reset to run the PROM image

        GRMON will now quit.....
quit
        Closing Xilinx cable
```

The programs hello and hello.prom are executed on the GR-UT699 evaluation board. GRMON connection is established by the JTAG Debug Link using Xilinx Platform USB cable (-xilusb) and a serial connection for the output of the programs.

3.0 Conclusion

Using batch files in GRMON, one can perform all the built-in commands within a batch file instead of typing in each command one at a time. Batch file are executed in GRMON at start up or while GRMON is running.

4.0 References

1. Frontgrade Colorado Springs Inc., UT699 LEON 3FT/SPARCTM V8 MicroProcessor Advanced User Manual, Aug. 2010
2. Frontgrade, GRMON User Manual, Version 1.1.49 April 2011
3. Frontgrade, MKPROM2 User's Manual, Version 2.0.35 January 2011

Revision History

Date	Revision #	Author	Change Description	Page #
5/11/2011	1.0.0	N/A	Initial Release	