AEROFLEX
GAISLER

Quick Start Guide for GR712 Development Board

Quick Start Guide for GR712RC-BOARD

Table of Contents

1. Introduction

1.1. Overview

This document is a quick start guide for the GR712RC Development Board.

The purpose of this document is to get users quickly started using the board. For a complete description of the board please refer to the GR712RC Development Board User Manual, the GR712RC system-on-chip is described in the GR712RC User Manual. This quick start guide does not contain as many technical details and is instead how-to oriented. However, to make the most of the guide the user should have glanced through the aforementioned documents and should ideally also be familiar with the GRMON debug monitor.

1.2. References

Table 1.1. References

RD-1	GR712RC Development Board User Manual
RD-2	GR712RC User Manual [http://gaisler.com/doc/gr712rc-usermanual.pdf]
RD-3	GR712RC Data Sheet [http://www.gaisler.com/doc/gr712rc-datasheet.pdf]
RD-4	GRMON User's Manual [http://www.gaisler.com/doc/grmon2.pdf]
RD-5	RTEMS homepage [http://www.rtems.org]
RD-6	RTEMS Cross Compilation System (RCC) [http://www.gaisler.com/index.php/products/operating-systems/rtems]
RD-7	RCC User's manual [http://gaisler.com/anonftp/rcc/doc]
RD-8	Aeroflex Gaisler RTEMS driver documentation [http://gaisler.com/anonftp/rcc/doc]
RD-9	GRTOOLS homepage [http://www.gaisler.com/index.php/downloads/grtools]
RD-10	Bare C Cross-Compilation System [http://www.gaisler.com/index.php/products/operating-systems/bcc]
RD-11	BCC User Manual [http://www.gaisler.com/doc/bcc.pdf]
RD-12	MKPROM2 User Manual [http://gaisler.com/doc/mkprom.pdf]

The referenced documents can be downloaded from http://www.gaisler.com.

AEROFLEX
GAISLER

2. Board Configuration

2.1. Overview

The primary source of information for board configuration is the GR712RC Development Board User Manual. The board requires some hardware configuration to fit with the customer requirements. In particular, the number of the GR712RC-BOARD's processor I/O pins limits the simultaneously available connections to external interfaces. To overcome this limitation, the SoC features an internal switch matrix, and a set of jumpers must be configured accordingly to route the signals to the appropriate headers on the board. The internal switch matrix is configured by enabling the respective interfaces via software. Additionally, clock selection might need to be configured by a set of jumpers and possibly the insertion of custom oscillators.
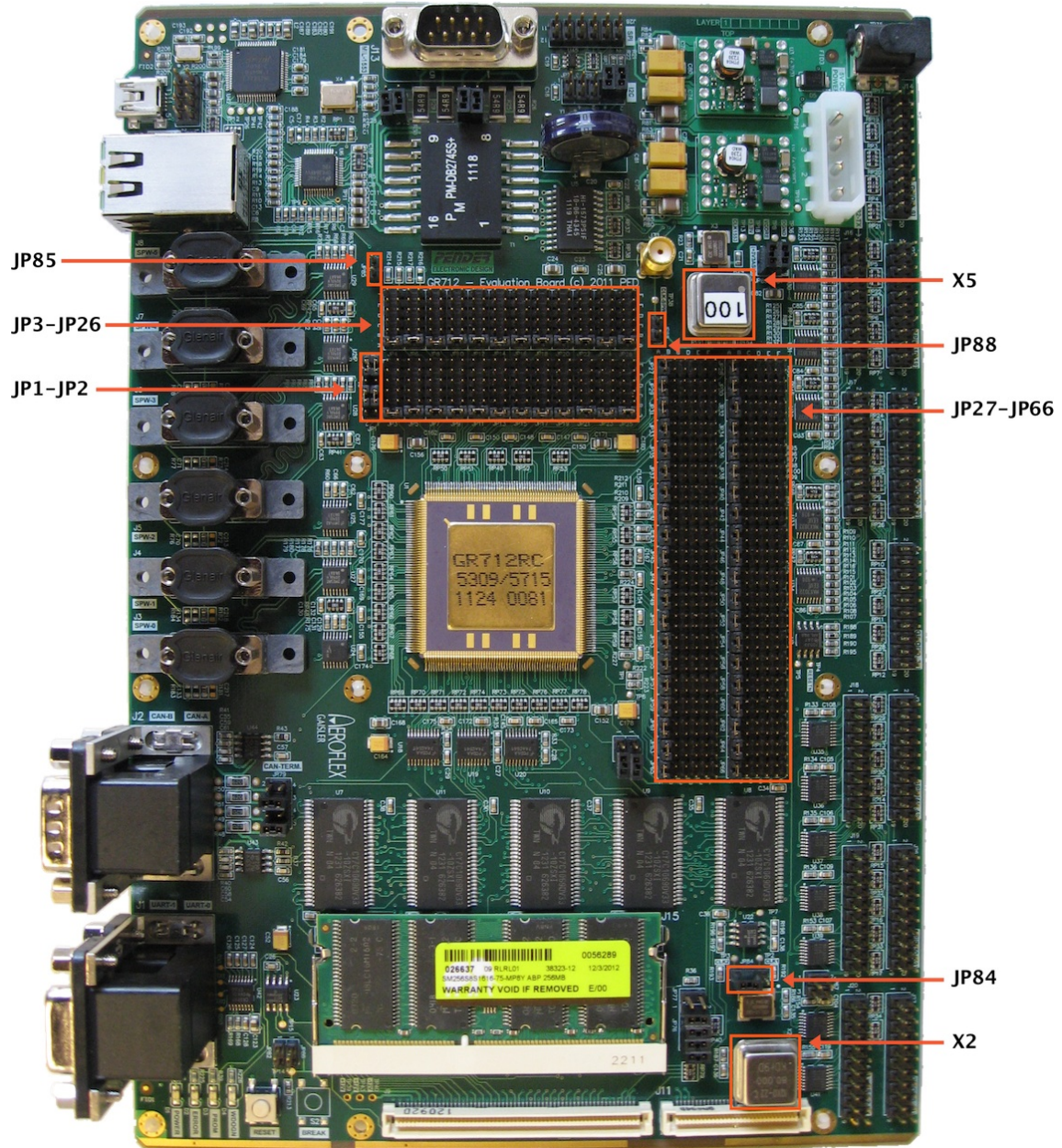

Figure 2.1. GR712RC-BOARD default configuration as delivired

2.2. Clock Sources

The minimum requirement in order for the board to work and to be able to connect to it, is that the clock sources are properly configured. The 80 MHz oscillator in socket X2 provided by default with the board is

connected to the system clock input through the JP84 jumper in the default configuration 2-3. The on-board soldered 48 MHz oscillator can be used instead by positioning the JP84 jumper on pins 1-2. Alternatively a custom oscillator can be installed in X2.

The SpaceWire clock is, by default, driven by an on board additional 100 MHz oscillator. If the user wants to use the system clock configured in the paragraph above as the source of the SpaceWire clock, then jumper JP88 must be inserted and the oscillator in socket X5 must be removed.

Refer to Section 2.14 of [RD-1] for further information about oscillators and clock inputs and more information about the system and SpaceWire clock.

Once the external clock sources are selected, further clock configuration can be done in software. The SpaceWire external clock source can be used as 1X, 2X or 4X, or the external system clock can be used in its place. This selection is done by configuring the SoC's General Purpose Register (GPREG). At reset the 1X SpaceWire clock received from the board is used internally.

For in depth information about configuring the SpaceWire and MIL-STD-1553 clocks through the GPREG, please refer to Chapter 3 and Chapter 13 of [RD-2].

2.3. I/O Switch Matrix

To overcome the limitation on the number of SoC pins, an internal switch matrix selects the input/output signals to connect to the pad. Additionally the chip I/O pins are connected to the board's I/O ports through an array of jumpers. One UART and two SpaceWire interfaces are routed independently of the internal switch matrix and the jumpers JP3 through JP66. In the default position A of jumpers JP3 through JP66, all multiplexed switch matrix signals are connected to the board's GPIO pins.

Six basic example configurations are provided to respond to typical use cases, as seen in Table 2.1. To use one of these configurations, the user has to insert jumpers JP3 through JP66 in the position described in the table. Refer to [RD-1] and GR712RC Development Board Schematic for more information on signal and GPIO configuration.

Table 2.1. Typical configurations

Cfg. description	I/O enabled	Jumper position
CPU for GEO applications	UART0, UART1, UART2, UART3, UART4, UART5 SpaceWire-0, SpaceWire-1, SpaceWire-2, SpaceWire-3, SpaceWire-4, SpaceWire-5 Mil-Std-1553-A, Mil-Std-1553-B SPI I2C	B
CPU for TMTC applications	UART0, UART1, UART2, UART3 SpaceWire-0, SpaceWire-1, SpaceWire-2, SpaceWire-3 SDRAM with optional Reed-Solomon CCSDS/ECSS TC & TM	C
CPU for LEO applications	UART0, UART1, UART2, UART3, UART4, UART5 SpaceWire-0, SpaceWire-1 SDRAM with optional Reed-Solomon ASCS16 CAN-A, CAN-B SLINK I2C	D
Instrument Controller, type A	UART0, UART1, UART2, UART3, UART4, UART5 SpaceWire-0, SpaceWire-1 SDRAM with optional Reed-Solomon CAN-A, CAN-B SLINK	E

Cfg. description	I/O enabled	Jumper position
	I2C	
Instrument Controller, type B	UART0, UART1, UART2, UART3, UART4, UART5 SpaceWire-0, SpaceWire-1, SpaceWire-2, SpaceWire-3 SDRAM with optional Reed-Solomon Ethernet SPI I2C	F

Once the board's jumpers are properly connected, the internal switch matrix must be driven by a set of enabling conditions. It is important to note that to obtain a proper functioning system, the I/O interfaces of the required configurations have to be enabled or clock ungated by software. See Chapter 2 and Table 9 of [RD-2] for further details on the switch matrix.

The I/O matrix is not limited to these pre-defined configurations. Jumpers can be custom configured according to the user requirements. See Section 2.4 of [RD-1] for further details.

2.4. UART

Jumpers JP1 and JP2 are used to select the output standard of the UART0 and UART1 interfaces between RS232 and RS422, and to route the signals to the J1 and J16 connectors respectively. In the default configuration the interfaces are connected to the J1 connectors UART-0 and UART-1 using the RS232 standard. While UART0 is not affected by the internal switch matrix, UART1 Rx is multiplexed and JP3 must be set to 3-4 in order to use it. Refer to the GR712RC Development Board Schematic for more information on how to configure UART0 and UART1 to use the RS422 standard.

2.5. PROM

The PROM width and PROM EDAC conditions are set by the state of the GPIO[3] and GPIO[1] pins at power up of the Processor. These pins are provided with pull-down resistors to set the default mode to 8 bit with no EDAC. If EDAC operation of the Flash PROM is desired, then jumper JP85 should be installed, to pull-up GPIO[1].

3. GRMON hardware debugger

3.1. Overview

GRMON is a debug monitor used to develop and debug GRLIB/LEON systems. The CPU and its peripherals are accessed on the AMBA bus through a debug-link connected to the PC. GRMON has GDB support which makes C/C++ level debugging possible by connecting GDB to the GRMON's GDB socket. With GRMON one can for example:

- Inspect LEON and peripheral registers
- Upload applications and/or program the FLASH
- Control execution flow by starting applications (run), continue execution (cont), single-stepping (step), inserting breakpoints/watchpoints (bp) etc.
- Inspect the current CPU state listing the back-trace, instruction trace and disassemble machine code.

The first step is to set up a debug link in order to connect to the board. The following section outlines which debug interfaces are available and how to use them on the GR712RC Development Board, after that a basic first inspection of the board is exemplified.

Several of the SoC's peripherals may be clock gated off. GRMON will enable all clocks if started with the flag **-cginit**. Within GRMON, the command **grcg enable all** will have the same effect.

GRMON is described on the homepage [http://www.gaisler.com/index.php/products/debug-tools] and in detail in [RD-4].

3.2. Debug-link alternatives

3.2.1. Connecting via the FTDI USB/JTAG interface

Please see GRMON User's Manual for how to set up the required FTDI driver software. Then connect the PC and the board using a standard USB cable into the USB-mini J12 USB-JTAG connector and issue the following command:

```
grmon -ftdi
```

3.2.2. Connecting via SpaceWire RMAP interface

GRMON has support for connecting to boards with SpaceWire interfaces as long as the SpaceWire has RMAP and automatic link start. An Ethernet to SpaceWire bridge (GRESB) is required to tunnel SpaceWire packets from the Ethernet network over to SpaceWire.

Please see the [RD-4] for information about connecting through a GRESB and optional parameters. Connect the GRESB SpW0 connector and the GR712RC-BOARD's J3 (SPW-0) or J4 (SPW-1) connector, then issue the following command:

```
grmon -gresb
```

3.3. First steps

The previous sections have described which debug-links are available and how to start using them with GRMON. The subsections below assume that GRMON, the host computer and the GR712RC-BOARD board have been set up so that GRMON can connect to the board.

When connecting to the board for the first time it is recommended to get to know the system by inspecting the current configuration and hardware present using GRMON. With the **info sys** command more details about the system is printed and with **info reg** the register contents of the I/O registers can be inspected. Below is a list of items of particular interest:

- AMBA system frequency is printed out at connect, if the frequency is wrong then it might be due to noise in auto detection (small error). See -freq flag in [RD-4].
- Memory location and size configuration is found from the **info sys** output. If the board has both SRAM and SDRAM interfaces, SDRAM can be mapped at the SRAM base address using the -nosram option of GRMON. See [RD-4] for further details.

- The GR712RC has a clock-gating unit which is able to disable/enable clocking and control reset signals. Clocks must be enabled for all cores that LEON software or GRMON will be using. The **grcg** command is described in [RD-4].

3.4. Connecting to the board

In the following example the FTDI debug-link is used to connect to the board. The auto-detected frequency, memory parameters and stack pointer are verified by looking at the GRMON terminal output below.

```
daniel@daniel:~$ grmon -ftdi

  GRMON2 LEON debug monitor v2.0.35 professional version

  Copyright (C) 2012 Aeroflex Gaisler - All rights reserved.
  For latest updates, go to http://www.gaisler.com/
  Comments or bug-reports to support@gaisler.com


Parsing -ftdi

Commands missing help:
 debug
 datacache

 JTAG chain (1): GR712RC
   Detected system:    GR712RC
   Detected frequency:  80 MHz

   Component                          Vendor
   LEON3-FT SPARC V8 Processor        Aeroflex Gaisler
   LEON3-FT SPARC V8 Processor        Aeroflex Gaisler
   JTAG Debug Link                    Aeroflex Gaisler
   GR Ethernet MAC                    Aeroflex Gaisler
   SatCAN controller                  Aeroflex Gaisler
   GRSPW2 SpaceWire Serial Link       Aeroflex Gaisler
   GRSPW2 SpaceWire Serial Link       Aeroflex Gaisler
   GRSPW2 SpaceWire Serial Link       Aeroflex Gaisler
   GRSPW2 SpaceWire Serial Link       Aeroflex Gaisler
   GRSPW2 SpaceWire Serial Link       Aeroflex Gaisler
   GRSPW2 SpaceWire Serial Link       Aeroflex Gaisler
   AMBA Wrapper for Core1553BRM       Aeroflex Gaisler
   CCSDS Telecommand Decoder          Aeroflex Gaisler
   CCSDS Telemetry Encoder            Aeroflex Gaisler
   SLINK Master                       Aeroflex Gaisler
   Memory controller with EDAC        Aeroflex Gaisler
   AHB/APB Bridge                     Aeroflex Gaisler
   LEON3 Debug Support Unit           Aeroflex Gaisler
   AHB/APB Bridge                     Aeroflex Gaisler
   OC CAN AHB interface               Aeroflex Gaisler
   Generic FT AHB SRAM module         Aeroflex Gaisler
   Generic UART                       Aeroflex Gaisler
   Multi-processor Interrupt Ctrl.    Aeroflex Gaisler
   Modular Timer Unit                 Aeroflex Gaisler
   SPI Controller                     Aeroflex Gaisler
   CAN Bus multiplexer                Aeroflex Gaisler
   General Purpose Register           Aeroflex Gaisler
   ASCS Master                        Aeroflex Gaisler
   General Purpose I/O port           Aeroflex Gaisler
   General Purpose I/O port           Aeroflex Gaisler
   AMBA Wrapper for OC I2C-master     Aeroflex Gaisler
   Clock gating unit                  Aeroflex Gaisler
   AHB Status Register                Aeroflex Gaisler
   Generic UART                       Aeroflex Gaisler
   Generic UART                       Aeroflex Gaisler
   Generic UART                       Aeroflex Gaisler
   Generic UART                       Aeroflex Gaisler
   Generic UART                       Aeroflex Gaisler
   Timer Unit with Latches            Aeroflex Gaisler

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

grmon2> info sys
   cpu0      Aeroflex Gaisler  LEON3-FT SPARC V8 Processor
             AHB Master 0
   cpu1      Aeroflex Gaisler  LEON3-FT SPARC V8 Processor
             AHB Master 1
   ahbjtag0  Aeroflex Gaisler  JTAG Debug Link
             AHB Master 2
   greth0    Aeroflex Gaisler  GR Ethernet MAC
             AHB Master 3
```

```
                   APB: 80000E00 - 80000F00
                   IRQ: 14
        satcan0    Aeroflex Gaisler  SatCAN controller
                   AHB Master 4
                   AHB: FFF20000 - FFF20100
                   IRQ: 14
        grspw0     Aeroflex Gaisler  GRSPW2 SpaceWire Serial Link
                   AHB Master 5
                   APB: 80100800 - 80100900
                   IRQ: 22
                   Number of ports: 1
        grspw1     Aeroflex Gaisler  GRSPW2 SpaceWire Serial Link
                   AHB Master 6
                   APB: 80100900 - 80100A00
                   IRQ: 23
                   Number of ports: 1
        grspw2     Aeroflex Gaisler  GRSPW2 SpaceWire Serial Link
                   AHB Master 7
                   APB: 80100A00 - 80100B00
                   IRQ: 24
                   Number of ports: 1
        grspw3     Aeroflex Gaisler  GRSPW2 SpaceWire Serial Link
                   AHB Master 8
                   APB: 80100B00 - 80100C00
                   IRQ: 25
                   Number of ports: 1
        grspw4     Aeroflex Gaisler  GRSPW2 SpaceWire Serial Link
                   AHB Master 9
                   APB: 80100C00 - 80100D00
                   IRQ: 26
                   Number of ports: 1
        grspw5     Aeroflex Gaisler  GRSPW2 SpaceWire Serial Link
                   AHB Master 10
                   APB: 80100D00 - 80100E00
                   IRQ: 27
                   Number of ports: 1
        b1553brm0 Aeroflex Gaisler  AMBA Wrapper for Core1553BRM
                   AHB Master 11
                   AHB: FFF00000 - FFF01000
                   IRQ: 14
        grtc0      Aeroflex Gaisler  CCSDS Telecommand Decoder
                   AHB Master 12
                   AHB: FFF10000 - FFF10100
                   IRQ: 14
        grtm0      Aeroflex Gaisler  CCSDS Telemetry Encoder
                   AHB Master 13
                   APB: 80000B00 - 80000C00
                   IRQ: 29
        adev14     Aeroflex Gaisler  SLINK Master
                   AHB Master 14
                   APB: 80000800 - 80000900
                   IRQ: 13
        mctrl0     Aeroflex Gaisler  Memory controller with EDAC
                   AHB: 00000000 - 20000000
                   AHB: 20000000 - 40000000
                   AHB: 40000000 - 80000000
                   APB: 80000000 - 80000100
                   8-bit prom @ 0x00000000
                   32-bit static ram: 1 * 8192 kbyte @ 0x40000000
                   32-bit sdram: 2 * 128 Mbyte @ 0x60000000
                   col 10, cas 2, ref 7.8 us
        apbmst0    Aeroflex Gaisler  AHB/APB Bridge
                   AHB: 80000000 - 80100000
        dsu0       Aeroflex Gaisler  LEON3 Debug Support Unit
                   AHB: 90000000 - A0000000
                   AHB trace: 256 lines, 32-bit bus
                   CPU0:  win 8, hwbp 2, itrace 256, V8 mul/div, srmmu, lddel 1, GRFPU
                          stack pointer 0x407ffff0
                          icache 4 * 4 kB, 32 B/line lru
                          dcache 4 * 4 kB, 16 B/line lru
                   CPU1:  win 8, hwbp 2, itrace 256, V8 mul/div, srmmu, lddel 1, GRFPU
                          stack pointer 0x407ffff0
                          icache 4 * 4 kB, 32 B/line lru
                          dcache 4 * 4 kB, 16 B/line lru
        apbmst1    Aeroflex Gaisler  AHB/APB Bridge
                   AHB: 80100000 - 80200000
        occan0     Aeroflex Gaisler  OC CAN AHB interface
                   AHB: FFF30000 - FFF31000
                   IRQ: 5
                   cores: 2
        ahbram0    Aeroflex Gaisler  Generic FT AHB SRAM module
                   AHB: A0000000 - A0100000
                   APB: 80100000 - 80100100
```

```
                  32-bit static ram: 256 kB @ 0xa0000000
        uart0     Aeroflex Gaisler   Generic UART
                  APB: 80000100 - 80000200
                  IRQ: 2
                  Baudrate 38461
        irqmp0    Aeroflex Gaisler   Multi-processor Interrupt Ctrl.
                  APB: 80000200 - 80000300
                  EIRQ: 12
        gptimer0  Aeroflex Gaisler   Modular Timer Unit
                  APB: 80000300 - 80000400
                  IRQ: 8
                  16-bit scalar, 4 * 32-bit timers, divisor 48
        spi0      Aeroflex Gaisler   SPI Controller
                  APB: 80000400 - 80000500
                  IRQ: 13
                  FIFO depth: 16, no slave select lines
                  Maximum word length: 32 bits
                  Controller index for use in GRMON: 0
        adev25    Aeroflex Gaisler   CAN Bus multiplexer
                  APB: 80000500 - 80000600
        grgpreg0  Aeroflex Gaisler   General Purpose Register
                  APB: 80000600 - 80000700
        adev27    Aeroflex Gaisler   ASCS Master
                  APB: 80000700 - 80000800
                  IRQ: 16
        gpio0     Aeroflex Gaisler   General Purpose I/O port
                  APB: 80000900 - 80000A00
        gpio1     Aeroflex Gaisler   General Purpose I/O port
                  APB: 80000A00 - 80000B00
        i2cmst0   Aeroflex Gaisler   AMBA Wrapper for OC I2C-master
                  APB: 80000C00 - 80000D00
                  IRQ: 28
        grcg0     Aeroflex Gaisler   Clock gating unit
                  APB: 80000D00 - 80000E00
                  GRMON did NOT enable clocks during initialization
        ahbstat0  Aeroflex Gaisler   AHB Status Register
                  APB: 80000F00 - 80001000
                  IRQ: 1
        uart1     Aeroflex Gaisler   Generic UART
                  APB: 80100100 - 80100200
                  IRQ: 17
                  Baudrate 38461
        uart2     Aeroflex Gaisler   Generic UART
                  APB: 80100200 - 80100300
                  IRQ: 18
                  Baudrate 38461
        uart3     Aeroflex Gaisler   Generic UART
                  APB: 80100300 - 80100400
                  IRQ: 19
                  Baudrate 38461
        uart4     Aeroflex Gaisler   Generic UART
                  APB: 80100400 - 80100500
                  IRQ: 20
                  Baudrate 38461
        uart5     Aeroflex Gaisler   Generic UART
                  APB: 80100500 - 80100600
                  IRQ: 21
                  Baudrate 38461
        grtimer0  Aeroflex Gaisler   Timer Unit with Latches
                  APB: 80100600 - 80100700
                  IRQ: 7
                  8-bit scalar, 2 * 32-bit timers, divisor 48

   grmon2> info reg
     GR Ethernet MAC
         0x80000e00  Control register                      0x04000080
         0x80000e04  Status register                       0x0000000a
         0x80000e08  MAC address MSB                       0x00000412
         0x80000e0c  MAC address LSB                       0x10884440
         0x80000e10  MDIO register                         0x7849084a
         0x80000e14  Tx descriptor register                0x10004000
         0x80000e18  Rx descriptor register                0xc8000000
         0x80000e1c  EDCL IP register                      0x00000000
     GRSPW2 SpaceWire Serial Link
         0x80100800  Control register                      0xa0010002
         0x80100804  Status/Interrupt-source               0x00600000
         0x80100808  Node address                          0x000000fe
         0x8010080c  Clock divisor                         0x00000000
         0x80100810  Destination key                       0x00000000
         0x80100814  Time                                  0x00000000
         0x80100818  Timer and Disconnect                  0x00000000
         0x80100820  DMA Channel 0 control/status          0x00000000
         0x80100824  DMA Channel 0 rx maximum length       0x00431000
```

```
     0x80100828   DMA Channel 0 tx desc. table address   0x40004000
     0x8010082c   DMA Channel 0 rx desc. table address   0x00000000
GRSPW2 SpaceWire Serial Link
     0x80100900   Control register                       0xa0010002
     0x80100904   Status/Interrupt-source                0x00200000
     0x80100908   Node address                           0x000000fe
     0x8010090c   Clock divisor                          0x00000000
     0x80100910   Destination key                        0x00000000
     0x80100914   Time                                   0x00000000
     0x80100918   Timer and Disconnect                   0x00000000
     0x80100920   DMA Channel 0 control/status           0x00000010
     0x80100924   DMA Channel 0 rx maximum length        0x00820000
     0x80100928   DMA Channel 0 tx desc. table address   0x00000000
     0x8010092c   DMA Channel 0 rx desc. table address   0x02000000
GRSPW2 SpaceWire Serial Link
     0x80100a00   Control register                       0x20000100
     0x80100a04   Status/Interrupt-source                0x00800000
     0x80100a08   Node address                           0x000000fe
     0x80100a0c   Clock divisor                          0x00000000
     0x80100a10   Destination key                        0x00000000
     0x80100a14   Time                                   0x00000000
     0x80100a18   Timer and Disconnect                   0x00000000
     0x80100a20   DMA Channel 0 control/status           0x00000000
     0x80100a24   DMA Channel 0 rx maximum length        0x00100040
     0x80100a28   DMA Channel 0 tx desc. table address   0x15000000
     0x80100a2c   DMA Channel 0 rx desc. table address   0x80000000
GRSPW2 SpaceWire Serial Link
     0x80100b00   Control register                       0x20000000
     0x80100b04   Status/Interrupt-source                0x00a00000
     0x80100b08   Node address                           0x000000fe
     0x80100b0c   Clock divisor                          0x00000000
     0x80100b10   Destination key                        0x00000000
     0x80100b14   Time                                   0x00000000
     0x80100b18   Timer and Disconnect                   0x00000000
     0x80100b20   DMA Channel 0 control/status           0x00000014
     0x80100b24   DMA Channel 0 rx maximum length        0x00323084
     0x80100b28   DMA Channel 0 tx desc. table address   0x5c406400
     0x80100b2c   DMA Channel 0 rx desc. table address   0xa701b800
GRSPW2 SpaceWire Serial Link
     0x80100c00   Control register                       0x20000000
     0x80100c04   Status/Interrupt-source                0x00a00000
     0x80100c08   Node address                           0x000000fe
     0x80100c0c   Clock divisor                          0x00000000
     0x80100c10   Destination key                        0x00000000
     0x80100c14   Time                                   0x00000000
     0x80100c18   Timer and Disconnect                   0x00000000
     0x80100c20   DMA Channel 0 control/status           0x00000000
     0x80100c24   DMA Channel 0 rx maximum length        0x01410104
     0x80100c28   DMA Channel 0 tx desc. table address   0x488b0800
     0x80100c2c   DMA Channel 0 rx desc. table address   0x20aaf800
GRSPW2 SpaceWire Serial Link
     0x80100d00   Control register                       0x20000200
     0x80100d04   Status/Interrupt-source                0x00a00000
     0x80100d08   Node address                           0x000000fe
     0x80100d0c   Clock divisor                          0x00000000
     0x80100d10   Destination key                        0x00000000
     0x80100d14   Time                                   0x00000000
     0x80100d18   Timer and Disconnect                   0x00000000
     0x80100d20   DMA Channel 0 control/status           0x00000004
     0x80100d24   DMA Channel 0 rx maximum length        0x0040032c
     0x80100d28   DMA Channel 0 tx desc. table address   0x01800000
     0x80100d2c   DMA Channel 0 rx desc. table address   0x06002000
AMBA Wrapper for Core1553BRM
     0xfff00100   B1553BRM status/control register       0xc5040001
     0xfff00104   B1553BRM interrupt settings            0xc5040001
     0xfff00108   AHB page address register              0xc5040001
CCSDS Telecommand Decoder
     0xfff10000   Global reset register                  0x00000000
     0xfff10004   Global control register                0x00000000
     0xfff1000c   Spacecraft Identifier Register         0x00000000
     0xfff10010   Frame acceptance report register       0x00000000
     0xfff10014   CLCW register 1                        0x00000000
     0xfff10018   CLCW register 2                        0x00000000
     0xfff1001c   Physical Interface Register            0x00000000
     0xfff10020   Control Register                       0x00000000
     0xfff10024   Status Register                        0x00000000
     0xfff10028   Address Space Register                 0x00000000
     0xfff1002c   Receive Read Pointer Register          0x00000000
     0xfff10030   Receive Write Pointer Register         0x00000000
CCSDS Telemetry Encoder
     0x80000b00   DMA control register                   0x00000004
     0x80000b04   DMA status register                    0x00000000
     0x80000b08   DMA length register                    0x00400002
```

```
       0x80000b0c  DMA descriptor pointer register       0x00000000
       0x80000b14  DMA revision register                 0x00010001
       0x80000b80  Control register                      0x00000000
       0x80000b84  Status register                       0x00000000
       0x80000b88  Configuration register                0x001affdf
       0x80000b90  Physical layer register               0x00000000
       0x80000b94  Coding sub-layer register             0x00000000
       0x80000b98  Attached Synchronization Marker       0x352ef853
       0x80000ba0  All frames generation register        0x00000000
       0x80000ba4  Master frame generation register      0x00000000
       0x80000ba8  Idle frame generation register        0x00000000
       0x80000bd0  OCF register                          0x00000000
Memory controller with EDAC
       0x80000000  Memory config register 1              0x0003c0ff
       0x80000004  Memory config register 2              0x8ac05460
       0x80000008  Memory config register 3              0x08174000
LEON3 Debug Support Unit
       0x90000024  Debug mode mask register              0x00000003
       0x90000000  CPU 0 Control register                0x000000ef
       0x90400020  CPU 0 Trap register                   0x000000b0
       0x90100000  CPU 1 Control register                0x6911d034
       0x90500020  CPU 1 Trap register                   0x6911d034
Generic FT AHB SRAM module
       0x80100000  Configuration Register                0x00200000
Generic UART
       0x80000104  UART Status register                  0x00000086
       0x80000108  UART Control register                 0x80000003
       0x8000010c  UART Scaler register                  0x0000009b
Multi-processor Interrupt Ctrl.
       0x80000200  Interrupt level register              0x00000000
       0x80000204  Interrupt pending register            0x00000000
       0x80000210  Interrupt status register             0x180c0002
       0x80000240  Interrupt mask register   0           0x00000000
       0x80000244  Interrupt mask register   1           0x00000000
       0x80000280  Interrupt force register  0           0x00000000
       0x80000284  Interrupt force register  1           0x00000000
Modular Timer Unit
       0x80000300  Scalar value register                 0x0000002f
       0x80000304  Scalar reload value register          0x0000002f
       0x80000308  Configuration register                0x00000144
       0x80000310  Timer 0 Value register                0xffffffff
       0x80000314  Timer 0 Reload value register         0xffffffff
       0x80000318  Timer 0 Control register              0x00000043
       0x80000320  Timer 1 Value register                0x00000000
       0x80000324  Timer 1 Reload value register         0x00000000
       0x80000328  Timer 1 Control register              0x00000040
       0x80000330  Timer 2 Value register                0x00000000
       0x80000334  Timer 2 Reload value register         0x00000000
       0x80000338  Timer 2 Control register              0x00000040
       0x80000340  Timer 3 Value register                0xfa16f3e8
       0x80000344  Timer 3 Reload value register         0xfffffffe
       0x80000348  Timer 3 Control register              0x00000040
SPI Controller
       0x80000400  Capability register                   0x01001002
       0x80000420  Mode register                         0x00000000
       0x80000424  Event register                        0x00000000
       0x80000428  Mask register                         0x00000000
       0x8000042c  Command register                      0x00000000
       0x80000430  Transmit register                     0x00000000
       0x80000434  Receive register                      0x20880021
General Purpose Register
       0x80000600  GR712RC general purpose register      0x00000000
General Purpose I/O port
       0x80000900  I/O port data register                0x419ff955
       0x80000904  I/O port output register              0x00000000
       0x80000908  I/O port direction register           0x00000000
       0x8000090c  I/O interrupt mask register           0x00000000
       0x80000910  I/O interrupt polarity register       0x00000058
       0x80000914  I/O interrupt edge register           0x00001100
       0x80000918  I/O bypass register                   0x00000000
General Purpose I/O port
       0x80000a00  I/O port data register                0xfff19ad9
       0x80000a04  I/O port output register              0x00000000
       0x80000a08  I/O port direction register           0x00000000
       0x80000a0c  I/O interrupt mask register           0x00000000
       0x80000a10  I/O interrupt polarity register       0x00000001
       0x80000a14  I/O interrupt edge register           0x0000e00c
       0x80000a18  I/O bypass register                   0x00000000
AMBA Wrapper for OC I2C-master
       0x80000c00  Clock prescale register               0x0000005f
       0x80000c04  Control register                      0x00000000
       0x80000c08  Receive register                      0x00000000
       0x80000c0c  Status register                       0x00000000
```

```
    Clock gating unit
        0x80000d00  Unlock register                         0x00000000
        0x80000d04  Clock enable register                   0x00000007
        0x80000d08  Reset register                          0x00000ff8
    AHB Status Register
        0x80000f00  Status register                         0x00000012
        0x80000f04  Failing address register                0x80000f04
    Generic UART
        0x80100104  UART Status register                    0x00000086
        0x80100108  UART Control register                   0x80000003
        0x8010010c  UART Scaler register                    0x0000009b
    Generic UART
        0x80100204  UART Status register                    0x00000086
        0x80100208  UART Control register                   0x80000003
        0x8010020c  UART Scaler register                    0x0000009b
    Generic UART
        0x80100304  UART Status register                    0x00000086
        0x80100308  UART Control register                   0x80000003
        0x8010030c  UART Scaler register                    0x0000009b
    Generic UART
        0x80100404  UART Status register                    0x00000086
        0x80100408  UART Control register                   0x80000003
        0x8010040c  UART Scaler register                    0x0000009b
    Generic UART
        0x80100504  UART Status register                    0x00000086
        0x80100508  UART Control register                   0x80000003
        0x8010050c  UART Scaler register                    0x0000009b
    Timer Unit with Latches
        0x80100600  Scalar value register                   0x0000002f
        0x80100604  Scalar reload value register            0x0000002f
        0x80100608  Configuration register                  0x0000003a
        0x8010060c  Latch configuration register            0x00000000
        0x80100610  Timer 0 Value register                  0xffffffff
        0x80100614  Timer 0 Reload value register           0xffffffff
        0x80100618  Timer 0 Control register                0x00000043
        0x8010061c  Timer 0 Latch register                  0x00000000
        0x80100620  Timer 1 Value register                  0xa0080048
        0x80100624  Timer 1 Reload value register           0xa0080048
        0x80100628  Timer 1 Control register                0x00000040
        0x8010062c  Timer 1 Latch register                  0x00000000
```

One can limit the output to certain cores by specifying the core(s) name(s) to the **info sys** and **info reg** commands. As seen below the memory parameters, first UART and first Timer core information is listed.

```
grmon2> info sys mctrl0
  mctrl0     Aeroflex Gaisler  Memory controller with EDAC
             AHB: 00000000 - 20000000
             AHB: 20000000 - 40000000
             AHB: 40000000 - 80000000
             APB: 80000000 - 80000100
             8-bit prom @ 0x00000000
             32-bit static ram: 1 * 8192 kbyte @ 0x40000000
             32-bit sdram: 2 * 128 Mbyte @ 0x60000000
             col 10, cas 2, ref 7.8 us

grmon2> info sys uart0 gptimer0
  uart0      Aeroflex Gaisler  Generic UART
             APB: 80000100 - 80000200
             IRQ: 2
             Baudrate 38461
  gptimer0  Aeroflex Gaisler  Modular Timer Unit
             APB: 80000300 - 80000400
             IRQ: 8
             16-bit scalar, 4 * 32-bit timers, divisor 80
```

The GR712RC has a clock-gating unit which can disable and enable clock gating and generate reset signals of certain cores in the SOC. With the GRMON **grcg** command the current setting of the clock-gating unit can be inspected and changed, the command line switch -cginit also affects the clock-gating unit. See [RD-4] for more information. Below is an example where the GRETH Ethernet core's clocks are turned on (not gated).

```
grmon2> grcg
  GRCLKGATE GR712RC info:
  Unlock register:        0x00000000
  Clock enable register:  0x00000006
  Reset register:         0x00000ff9

  GR712RC decode of values:
  +------+----------+---------------------------+----------+---------+-------+
  | Gate | Core(s)  | Description               | Unlocked | Enabled | Reset |
  +------+----------+---------------------------+----------+---------+-------+
```

```
| 0  | GRETH    | 10/100 Ethernet MAC      | 0  | 0  | 1 |
| 1  | GRSPW    | Spacewire link 0         | 0  | 1  | 0 |
| 2  | GRSPW    | Spacewire link 1         | 0  | 1  | 0 |
| 3  | GRSPW    | Spacewire link 2         | 0  | 0  | 1 |
| 4  | GRSPW    | Spacewire link 3         | 0  | 0  | 1 |
| 5  | GRSPW    | Spacewire link 4         | 0  | 0  | 1 |
| 6  | GRSPW    | Spacewire link 5         | 0  | 0  | 1 |
| 7  | CAN      | CAN core 1 & 2           | 0  | 0  | 1 |
| 8  | SatCAN   | SatCAN controller        | 0  | 0  | 1 |
| 9  | GRTM     | Telemetry Encoder        | 0  | 0  | 1 |
| 10 | GRTC     | Telecommand Decoder      | 0  | 0  | 1 |
| 11 | B1553BRM | MIL-STD-1553 BRM         | 0  | 0  | 1 |
+------+----------+--------------------------+----------+---------+-------+

grmon2> grcg enable 0

grmon2> grcg
  GRCLKGATE GR712RC info:
  Unlock register:        0x00000000
  Clock enable register: 0x00000007
  Reset register:        0x00000ff8

  GR712RC decode of values:
  +------+----------+--------------------------+----------+---------+-------+
  | Gate | Core(s)  | Description              | Unlocked | Enabled | Reset |
  +------+----------+--------------------------+----------+---------+-------+
  | 0  | GRETH    | 10/100 Ethernet MAC      | 0  | 1  | 0 |
  | 1  | GRSPW    | Spacewire link 0         | 0  | 1  | 0 |
  | 2  | GRSPW    | Spacewire link 1         | 0  | 1  | 0 |
  | 3  | GRSPW    | Spacewire link 2         | 0  | 0  | 1 |
  | 4  | GRSPW    | Spacewire link 3         | 0  | 0  | 1 |
  | 5  | GRSPW    | Spacewire link 4         | 0  | 0  | 1 |
  | 6  | GRSPW    | Spacewire link 5         | 0  | 0  | 1 |
  | 7  | CAN      | CAN core 1 & 2           | 0  | 0  | 1 |
  | 8  | SatCAN   | SatCAN controller        | 0  | 0  | 1 |
  | 9  | GRTM     | Telemetry Encoder        | 0  | 0  | 1 |
  | 10 | GRTC     | Telecommand Decoder      | 0  | 0  | 1 |
  | 11 | B1553BRM | MIL-STD-1553 BRM         | 0  | 0  | 1 |
  +------+----------+--------------------------+----------+---------+-------+
```

4. Software

4.1. Overview

Aeroflex Gaisler provides a comprehensive set of software tools to run several different operating systems. The LEON platform supports the following workflows:

Table 4.1.

BCC	the Bare C Cross-Compiler System is a toolchain to compile bare C or C++ applications directly on top of the processor without the burden and the servises provided by an operating system
RTEMS	a hard Real Time Operating System. Aeroflex Gaisler provides RCC, a toolchain to develop and compile RTEMS applications specifically for the LEON
Linux	the open source operating system. Board Support Packages and tools to ease the compilation and deployment of the kernel are provided
VxWorks	an embedded real-time operating system developed by WindRiver. Aeroflex Gaisler provides a LEON architectural port (HAL) and a Board Support Package (BSP) in full source code

Hardware simulators for the LEON platform are available in the form of TSIM and GRSIM. The TSIM simulator emulates a mono-processor computer system based on either the ERC32 or LEON processors. It can be extended to emulate custom I/O functions through loadable modules. The GRSIM simulator emulates a multi-processor LEON2 or LEON3 system, and has a more accurate modelling of the on-chip IP cores and AMBA buses. It is time-based rather that instruction-based (as TSIM), and can be attached to other simulation frameworks such as System-C.

Developer tools are generally provided for both Linux and Windows operating systems. Aeroflex Gaisler also provides an integrated, easy-to-use solution to help programmers with the task of developing for the LEON. The LEON Integrated Development Environment for Eclipse (LIDE) is an Eclipse plug-in integrating compilers, software and hardware debuggers in a graphical user interface. The plugin makes it possible to cross-compile C and C++ application for LEON and ERC32, and to debug them on either simulator and target hardware (TSIM or GRMON).

The recommended method to load software onto a LEON board is by connecting to a debug interface of the board through the GRMON hardware debugger. Execution of programs by a PROM-loaded bootloader, like u-boot, is also possible. Aeroflex Gaisler provides the free MKPROM2 tool to encapsulate BCC or RTEMS applications in a simple boot-loader.

Currently u-boot for the GR712RC Development Board is not provided by Aeroflex Gaisler

4.2. Bare C Cross-Compiler System

4.2.1. Overview

The Bare C Cross-Compiler (BCC for short) is a GNU-based cross-compilation system for LEON processors. It allows cross-compilation of C and C++ applications for LEON3/4. This section gives the reader a brief introduction on how to use BCC together with the GR712RC Development Board. It will be demonstrated how to install BCC, build an existing sample project and run it on the board using GRMON.

4.2.2. Installing BCC

The BCC toolchain includes the GNU C/C++ cross-compiler (4.4.2 and 3.4.4), GNU Binutils, Newlib embedded C library, the Bare-C run-time system with interrupt support and the GNU debugger (GDB). The toolchain can be downloaded from [RD-10] and is available for both Linux and Windows. Further information about BCC can be found in [RD-11].

The installation process of BCC is straight forward by first extracting the toolchain into `C:\opt` or `/opt` on Linux, and, in order for the compiler to be found, by adding the binary directory `/opt/sparc-elf-*/bin` into the PATH variable.

BCC for Windows is provided for native Windows (MinGW) and for the Cygwin environment.

4.2.3. Compiling with BCC

The following command shows an example of how to compile a simple Hello World program with BCC.

```
sparc-elf-gcc -O2 -g hello.c -o hello.exe
```

All gcc options are described in the gcc manual, but some useful options are reported below:

Table 4.2. BCC's GCC compiler relevant options

`-g`	generate debugging information - must be used for debugging with gdb
`-msoft-float`	emulate floating-point - must be used if no FPU exists in the system
`-mcpu=v8`	generate SPARC V8 mul/div instructions - needs hardware multiply and divide
`-O2` or `-O3`	optimize code maximum performance and minimal code size
`-qsvt`	use the single-vector trap model
`-qnoambapp`	disable AMBA Plug & Play recursive scanning across AHB2AHB bridges - creates a leaner software image

4.2.4. Running and debugging with GRMON

Once your executable is compiled, connect to your GR712RC-BOARD with GRMON. The following log shows how to load and run an executable. Note that the console output is redirected to GRMON by the use of the `-u` command line switch, so that printf output is shown directly in the GRMON console.

```
[andrea@localhost Desktop]$ grmon -ftdi -u

  GRMON2 LEON debug monitor v2.0.42 professional version

  Copyright (C) 2013 Aeroflex Gaisler - All rights reserved.
  For latest updates, go to http://www.gaisler.com/
  Comments or bug-reports to support@gaisler.com

Parsing -ftdi
Parsing -u

[...]

grmon2> load hello.exe
  40000000 .text                   23.6kB /  23.6kB   [===============>] 100%
  40005E70 .data                    2.7kB /   2.7kB   [===============>] 100%
  Total size: 26.29kB (803.58kbit/s)
  Entry point 0x40000000
  Image /home/andrea/Desktop/hello.exe loaded

grmon2> run
Hello world!

  CPU 0:  Program exited normally.
  CPU 1:  Power down mode
```

To debug the compiled program you can insert break points, step and continue directly from the GRMON console. Compilation symbols are loaded automatically by GRMON once you load the executable. An example is provided below.

```
grmon2> load hello.exe
  40000000 .text                   23.6kB /  23.6kB   [===============>] 100%
  40005E70 .data                    2.7kB /   2.7kB   [===============>] 100%
  Total size: 26.29kB (806.59kbit/s)
  Entry point 0x40000000
  Image /home/andrea/Desktop/hello.exe loaded

grmon2> bp main
  Software breakpoint 1 at <main>

grmon2> run

  CPU 0:  breakpoint 1 hit
          0x40001928: b0102000  mov  0, %i0  <main+4>
  CPU 1:  Power down mode
```

```
grmon2> step
  0x40001928: b0102000  mov  0, %i0  <main+4>

grmon2> step
  0x4000192c: 11100017  sethi  %hi(0x40005C00), %o0  <main+8>

grmon2> cont
Hello world!

  CPU 0:  Program exited normally.
  CPU 1:  Power down mode

grmon2> Exiting GRMON
```

Alternatively you can run GRMON with the `-gdb` command line option and then attach a gdb session to it. For further information see Chapter 3 of [RD-11].

4.3. RTEMS Real Time Operating System

4.3.1. Overview

RTEMS is a real time operating system maintained at [RD-5] that supports the LEON CPU family. Aeroflex Gaisler distributes a precompiled RTEMS toolchain for LEON called RCC [RD-6]. This section gives the reader a brief introduction on how to use RTEMS together with the GR712RC Development Board. It will be demonstrated how to install RCC and build an existing sample RTEMS project from RCC and run it on the board using GRMON.

The RCC toolchain includes a prebuilt toolchain with GNU BINUTILS, GCC, NewlibC and GDB for Linux and Windows (mingw). It also contains prebuilt RTEMS kernels for the LEON2, LEON3/4 BSPs single-core and for multi-core development, see [RD-7] for more information. The LEON BSP specific drivers are documented in [RD-8].

Samples RTEMS projects are available within the toolchain package, installed into `rtems-x.y/src/samples`.

4.3.2. Installing RCC

The RCC toolchain is downloadable from the RCC homepage at [RD-6]. The full installation procedure is found in the RCC manual [RD-7]. Windows users are recommended to install the UNIX-like environment MSYS before proceeding.

The installation process of RCC is straight forward by first extracting the toolchain into `C:\opt` or `/opt` on Linux, then extracting the source distribution into the `/opt/rtems-x.y/src/` directory. In order for the compiler to be found one has to add the binary directory `/opt/rtems-x.y/bin` into the PATH variable as below:

```
$ cd /opt
$ tar -xf sparc-rtems-4.10-...-linux.tar.bz2
$ cd rtems-4.10/src
$ tar -xf rtems-4.10-...-src.tar.bz2
$ export PATH=$PATH:/opt/rtems-4.10/bin
```

The GRTOOLS package [RD-9] includes a set of UNIX tools together with a MSYS environment suitable for RCC.

4.3.3. Building an RTEMS sample application

Once the toolchain is set up, you can compile and link a sample RTEMS application by doing:

```
sparc-rtems-gcc -g -O2 rtems-hello.c -o rtems-hello
```

RCC's gcc creates executables for LEON3/4 by default. The default load address is at the start of the RAM, i.e. 0x40000000. All compilation options are described in [RD-7], but some useful options are reported below:

Table 4.3. RCC's GCC compiler relevant options

`-g`	generate debugging information - must be used for debugging with gdb
`-msoft-float`	emulate floating-point - must be used if no FPU exists in the system
`-mcpu=v8`	generate SPARC V8 mul/div instructions - needs hardware multiply and divide
`-O2 or -O3`	optimize code maximum performance and minimal code size
`-qleon3std`	generate LEON3/4 executable without driver manager startup initialization
`-qleon3mp`	generate LEON3/4 Multiprocessor executable (AMP)

4.3.4. Running and debugging with GRMON

Once your executable is compiled, connect to your GR712RC-BOARD with GRMON. The following log shows how to load and run an executable. Note that the console output is redirected to GRMON by the use of the −u command line switch, so that printf output is shown directly in the GRMON console.

```
[andrea@localhost samples]$ grmon -ftdi -u

  GRMON2 LEON debug monitor v2.0.42 internal version

  Copyright (C) 2013 Aeroflex Gaisler - All rights reserved.
  For latest updates, go to http://www.gaisler.com/
  Comments or bug-reports to support@gaisler.com

Parsing -ftdi
Parsing -u

[...]

grmon2> load rtems-hello
  40000000 .text                   136.4kB / 136.4kB   [===============>] 100%
  400221A0 .data                     4.4kB /   4.4kB   [===============>] 100%
  40023350 .jcr                         4B             [===============>] 100%
  Total size: 140.83kB (780.05kbit/s)
  Entry point 0x40000000
  Image /home/andrea/Desktop/samples/rtems-hello loaded

grmon2> run
Hello World

  CPU 0:  Program exited normally.
  CPU 1:  Power down mode
```

To debug the compiled program you can insert break points, step and continue directly from the GRMON console. Compilation symbols are loaded automatically by GRMON once you load the executable. An example is provided below.

```
grmon2> load rtems-hello
  40000000 .text                   136.4kB / 136.4kB   [===============>] 100%
  400221A0 .data                     4.4kB /   4.4kB   [===============>] 100%
  40023350 .jcr                         4B             [===============>] 100%
  Total size: 140.83kB (781.11kbit/s)
  Entry point 0x40000000
  Image /home/andrea/Desktop/samples/rtems-hello loaded

grmon2> bp Init
  Software breakpoint 1 at <Init>

grmon2> run

  CPU 0:  breakpoint 1 hit
          0x400011f8: 1110007f  sethi  %hi(0x4001FC00), %o0  <Init+4>
  CPU 1:  Power down mode

grmon2> step
  0x400011f8: 1110007f  sethi  %hi(0x4001FC00), %o0  <Init+4>

grmon2> step
  0x400011fc: 4000003b  call  0x400012E8  <Init+8>

grmon2> cont
Hello World

  CPU 0:  Program exited normally.
  CPU 1:  Power down mode
```

```
grmon2> Exiting GRMON
```

Alternatively you can run GRMON with the `-gdb` command line option and then attach a gdb session to it. For further information see Chapter 3 of [RD-7].

4.4. MKPROM2

4.4.1. Overview

To run application from the on-board PROM, it's necessary to create a bootable PROM image file. 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:

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

4.4.2. Usage of MKPROM2

The MKPROM2 tool can be downloaded from the Aeroflex Gaisler website [http://gaisler.com/index.php/downloads/compilers]. No installation is required, but the directory containing the executable must be included in the system's PATH, together with a valid SPARC toolchain (sparc-elf, sparc-rtems or sparc-linux).

To generate a boot PROM for a GR712RC Development Board and running your program from SRAM:

```
mkprom2 -leon3 -freq 80 -rmw -ramsize 8192 -romsize 65536 -baud 38400 -ramws 1 -o hello.prom hello.exe
```

This example command will work for a board in the default configuration at delivery, with a system clock frequency of 80 MHz. The `-ramsize` and `-romsize` options are expressed in KB. The former value is 8MB, the size of the SRAM. The latter value is 64 MB, the size of the on-board flash. Finally the `-ramws` option sets the number of wait states during SRAM access to 1, needed when running the system at 80 MHz, but might be unnecessary at lower frequencies.

To generate a boot PROM for a GR712RC Development Board and running your program from SDRAM:

```
mkprom2 -leon3 -freq 80 -rmw -nosram -sdram 128 -romsize 65536 -baud 38400 -o hello.prom hello.exe
```

This example command will, again, work for a board in the default configuration at delivery. The `-nosram` option will disable the SRAM and the `-sdram` option sets the size of the available SDRAM. This value is 128 MB, which is the value for the SODIMM provided by default with the board.

When SRAM is disabled, the SDRAM address range is moved from 0x60000000 to 0x40000000, therefore not requiring recompilation of executables. To run your program in SDRAM without disabling SRAM, you need to link your program to the 0x60000000 address at compilation time. See the manual of your toolchain for more information.

It's required that the MKPROM2 command line parameters match your system configuration. For more information about command line options, please refer to [RD-12].

If EDAC is enabled on the board's PROM, then the `-bch8` flag must be included in the command line. The generated PROM image that needs to be flashed on the device in this case would be `hello.prom.bch8`.

Once the PROM file is generated, it can be loaded onto the board with GRMON. Once GRMON is attached to the board, run the following commands to program the PROM.

```
flash
flash erase all
flash load hello.prom
```

```
verify hello.prom
```

For further information about connecting to the board with GRMON, see Chapter 3.

4.5. VxWorks

4.5.1. Overview

VxWorks is an embedded real-time operating system developed by WindRiver. Aeroflex Gaisler provides a LEON architectural port (HAL) and a Board Support Package (BSP) in full source code for VxWorks.

The VxWorks package includes a quick start guide and technical support. Contact support@gaisler.com for more information.

5. Frequently Asked Questions / Common Mistakes / Know Issues

5.1. Clock gating

Several of the design's peripherals may be clock gated off. GRMON will enable all clocks if started with the flag -cginit. Within GRMON, the command **grcg enable all** will have the same effect.

Alternatively, if a boot loader is used instead of GRMON to load an executable, then clock gating must be setup via the General Purpose register. Clock source/divider selection must also be setup for the MIL-1553, SpaceWire and TM cores. See Chapter 13 of [RD-2].

5.2. GRMON issues

When connected to the board, the message "stack pointer not set" will be shown by the command **info sys** in case GRMON doesn't find any memory.

5.3. Clock problems

Ensure that the jumper JP84, selecting the clock source, is always present. A combination of its absence and the presence of jumper JP88, can lead to unexpected processor behavior.

When jumper JP88 is present, the oscillator in socket X5, which is provided by default, must be disconnected or it will short with the main clock source, leading to possible damage to the oscillators and unexpected behavior.

5.4. Switch Matrix Configuration Problems

Ensure that the jumper array is properly configured and that any I/O peripheral required is clock ungated or enabled. The internal switch matrix routing is explained more in depth in Chapter 2 of GR712RC User Manual.

If an IP core behaves correctly, as seen from software, but does not receive/transmit any data from the outside, first check that the jumper array is properly configured. The problem might also arise when conflicting cores are enabled. Check Table 8 from [RD-2] for further information on conflicting cores.

5.5. GPIO

Some of the GPIOs have special meaning on power-up, GPIO[1] and GPIO[3] configure the PROM area of the memory controller and GPIO[42], GPIO[40], GPIO[37] and GPIO[34] are used for the SPW clock divider reset value.

These pins are provided with pull-down resistors by default. If measuring the state of these GPIO pins, please take into account the effect of these pull-down resistors. Conversely, if an external signal is connected to the GPIO[3] and GPIO[1] pins, this may override the expected state of the pin at power up.

See Section 2.3.2 and Section 2.6.2 of [RD-1] for more information.

5.6. SDRAM configuration

SDRAM is, by default, not configured on the board. Ensure that the switch matrix jumper configuration is correctly set as to enable SDRAM. If in doubt, you can use a default configuration that supports SDRAM. See Section 2.3 for more details.

Only half of the installed SDRAM will be available in the system, as reported by GRMON's **info sys** command. This limitation is due to the fact that the SODIMM provides 64 bit data paths, but in the standard LEON model only 32 bits of the SDRAM are used, plus 16 additional data bits for the RS/EDAC memory bits.

5.7. Multiprocessor & legacy support

Code compiled for the single core LEON3 will generally be able to run unmodified on the GR712RC. The second core is inactivated after reset and unless it's activated (by writing a specific bit in the IRQ controller) it will remain inactivated and the chip will behave as a single-CPU system.

5.8. Interrupts

When using a multiprocessor OS like RTEMS-AMP, Linux or VxWorks the default IRQ for interprocessor cross-calls, IRQ 14, clashes with the MIL-1553, Ethernet and Telecommand IP cores. The OS may need to be reconfigured by changing the IRQ value, which is usually a define, in the source code of your operating system and rebuilding it. This should not be an issue with single-core RTEMS.

5.9. GRMON Debug Link Limitations

The GR712RC does not support debugging over Ethernet. EDCL is not included in the Ethernet core design. Refer to Chapter 3 for an introduction to the supported debug links.

5.10. MIL-1553

The 1553 IP core in the GR712RC is an Actel Core1553BRM with an AMBA adapter developed by Aeroflex Gaisler. Actel's core is documented on Actel's website [http://www.actel.com/ip-docs/Core1553BRM_HB.pdf], while the wrapper is documented in [RD-2].

The correct RTEMS driver to use for the MIL-1553 core is B1553BRM. This should not be confused with GR1553B which is the driver for Aeroflex Gaisler's in-house developed core. To use the core, users need to set up clock gating and clock selection with the general purpose register.

There are some restrictions on what clock frequencies can be used, see Section 3.3 of [RD-2].

Users also need to set a register inside the Core1553BRM to match the BRM frequency used. This is usually done by the driver in the RTEMS/VxW case (default is 24 MHz). Below is provided an example routine for setting up GR712RC clocking to external 24 MHz clock. This routine can be used, for instance, as mkprom2 bdinit. In this case it needs to be compiled with -O2 to avoid using stack.

```
static void gr712_init(void)
{
  volatile unsigned long *p;
  /* Select external 1553 clk through GPREG */
  p = (volatile unsigned long *)0x80000600;
  *p |= 0x20;
  /* Ungate 1553 clock and reset */
  p = (volatile unsigned long *)0x80000D00;
  p[0] = (1<<11);
  p[2] = (1<<11);
  p[1] = (1<<11);
  p[2] = 0;
  p[0] = 0;
  /* Set Core1553BRM to 24 MHz operation */
  p = (volatile unsigned long *)0xFFF00000;
  p[32] |= 3;
}
```

6. Support

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