

Table of Contents

Chapter 1: Introduction	3
1.1 Scope	3
1.2 CAES LEAP Board Overview	3
1.3 AX9-4250252-001 LEAP Board Stack Package Contents (Main SBC Only)	5
1.4 AX9-4250252-000 LEAP Board Stack Package Contents (Main SBC and Mezzanine)	5
1.5 CAES LEAP Board Power Input	5
Chapter 2: Architecture	6
2.1 UT700 Brief Overview	6
2.2 UT700 LEON 3FT SPARC V8 Processor	6
2.3 Memory Interfaces	7
2.4 SpaceWire Links	8
2.5 UT700 Timer Unit	8
2.6 System Reset Button	9
2.7 System Status LED	9
2.8 USB General Purpose UART I/F	9
2.9 UT700 Ethernet Interface	9
2.10 System Clock Selection	10
2.11 UT700 General Purpose I/O	10
2.12 UT700 1553B Port to Mezzanine	11
2.13 UT700 CAN Bus Interface	12
2.14 UT700 SPI Bus Interface	12
2.15 LEAP Core	12
2.16 AHB Core Mapping	13
2.17 Non-Volatile Memory Mapping	14
2.18 SDRAM Memory Mapping	14
Chapter 3: Mezzanine Power Interface	15
3.1 Mezzanine Connector Power	15
Chapter 4: Optional LEAP with Mezzanine Card	16
4.1 Mezzanine Card SPI Interface	16
4.2 Mezzanine Card CAN Bus 1 Interface	17
4.3 Mezzanine Card CAN Bus 2 Interface	18
4.4 Mezzanine Card 1553B Interface	19
4.5 Mezzanine Card SpaceWire Interface	20

Chapter 5: Software Development	21
5.1 "Out of the Box" Experience.....	21
5.2 Tool Chains.....	21
5.3 Downloading Software to the Target System.....	21
5.4 Bare-C cross-compiler system	21
5.4.1 Installation	22
5.4.2 Application development for BCC	22
5.4.3 Booting with mkprom2	23
5.5 RTEMS.....	25
5.5.1 RTEMS using Eclipse	25
5.5.2 Configuring Eclipse for Cross Compilation	26
5.5.3 Remote Debugging with GDB.....	28
5.5.4 Running RTEMS applications on the LEAP board	31
5.6 VxWorks	35
5.6.1 VxWorks Kernel	35
5.6.2 VxWorks Kernel Module.....	36
5.6.3 Booting with Bootrom.....	36
5.6.4 Debugging with Workbench.....	39
5.6.5 Configuring the Target Server/Manager.....	39
5.6.6 Debugging the Target Device	40
5.7 Linux	41
5.7.1 Linux Embedded System	41
5.7.2 Booting Linux with mkprom2	42
Chapter 6: Benchmarks	44

Chapter 1: Introduction

1.1 Scope

This document describes the Leon Expandable Application Platform (LEAP) design implemented for the CAES UT700 LEON3-FT processor. The LEAP card design is intended to be a low-cost development platform to familiarize users with the CAES UT700 processor, as well as allow custom expansion based on application requirements.

The following hardware and software components are required to use the LEAP evaluation board:

- PC work station running Windows or Linux
- CAES GRMON-Pro

For new users to UT700 software development, the following tools are recommended:

- BCC Bare-C LEON Cross-compiler
- RCC RTEMS LEON Cross-compiler system

1.2 CAES LEAP Board Overview

The CAES LEAP provides a flexible development platform for customers wanting to develop software that works on the CAES UT700 Standard Product with minimal cost investment. The CAES LEAP has the following features:

- A CAES UT700 LEON 3 FT standard product
- 8 Mbytes NV memory storage
- 32 Mbytes SDRAM
- One USB UART interface via standard USB
- One 10T/100 Mbit/s Ethernet port
- JTAG interface for programming and debug of UT700 LEON 3FT (requires XILINX USB Platform Cable)
- One 192-pin mezzanine card expansion connector
- On-board Programmable LEAP main clock

A block diagram of the LEAP card is shown in [Figure 1.1](#).

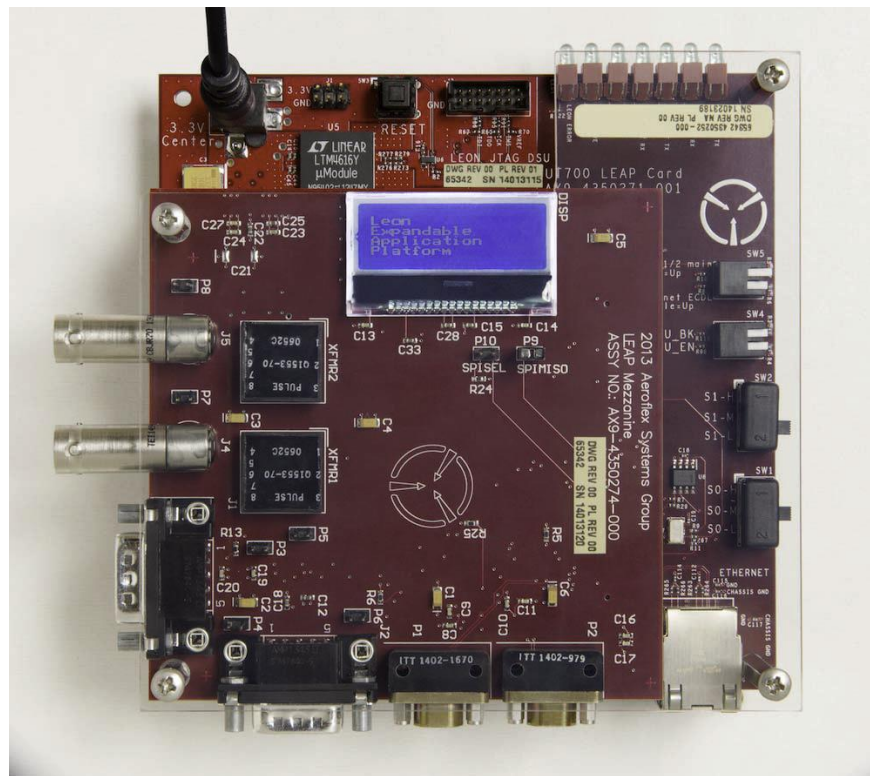
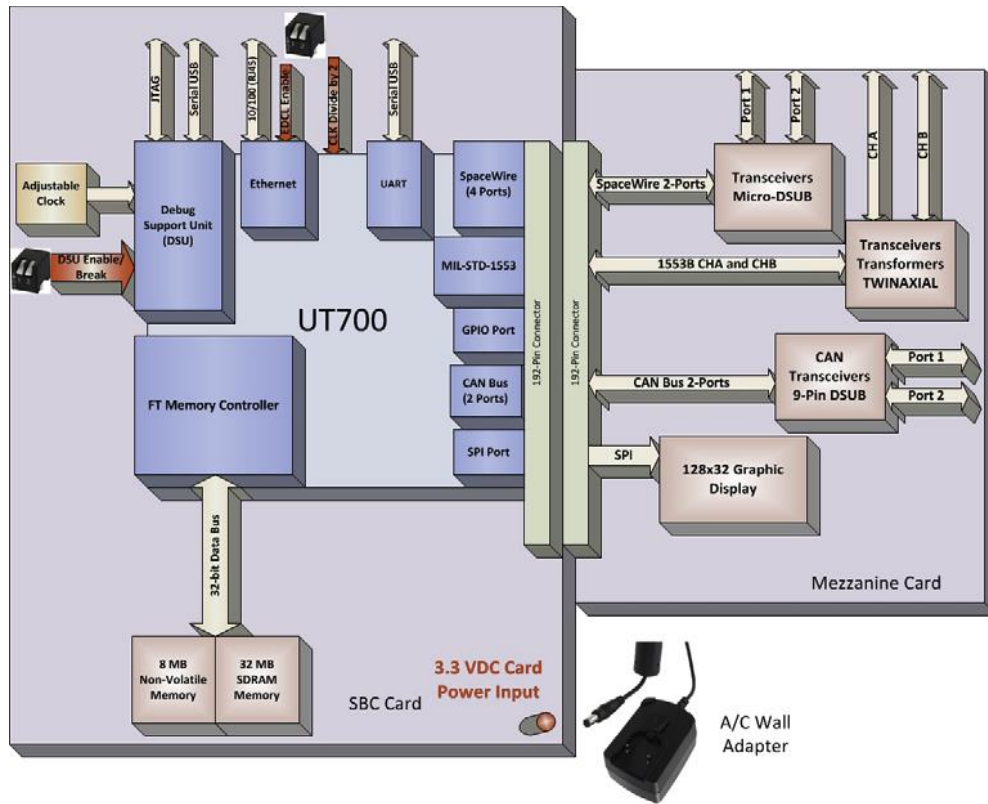


Figure 1.1: LEAP Top-Level Block Diagram and Assembly (AX9-4250252)

1.3 AX9-4250252-001 LEAP Board Stack Package Contents (Main SBC Only)

- One LEAP (4350271) card (main SBC & Memory)
- One A/C Wall Adapter
- One Euro-style A/C interface plate
- One US-style A/C interface plate
- One LEAP Board User Guide (CD)

1.4 AX9-4250252-000 LEAP Board Stack Package Contents (Main SBC and Mezzanine)

- One LEAP (4350271) card (main SBC & Memory)
- One LEAP (4350274) card (mezzanine with SPI display, two CAN bus, two SpaceWire and a dual-redundant CHA/ CHB 1553B ports)
- One A/C Wall Adapter
- One Euro-style A/C interface plate
- One US-style A/C interface plate
- One LEAP Board User Guide (CD)

1.5 CAES LEAP Board Power Input

The CAES LEAP utilizes one AC wall-adapter to provide the 3.3VDC used to operate the board. CAES includes the A/C wall adapter in the delivery package to enable “out-of-the-box” software operation and development. Connect the A/C wall adapter DC port-side cable to the LEAP card as shown in [Figure 1.2](#), the wall-adapter side should be connected to a compatible A/C wall outlet.

A/C Wall Adapter (included) DC Power Side Input Jack/Port

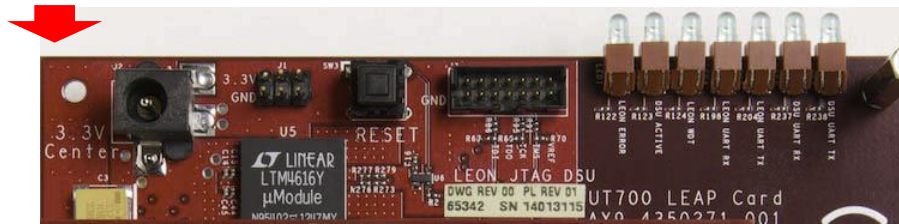


Figure 1.2: LEAP Power Input

Chapter 2: Architecture

2.1 UT700 Brief Overview

The CAES LEAP design consists of the UT700 Leon 3FT processor and a set of IP cores connected through the AMBA (Advanced Microcontroller Bus Architecture) AHB/APB (Advanced High-Performance Bus/Advanced Peripheral Bus) buses.

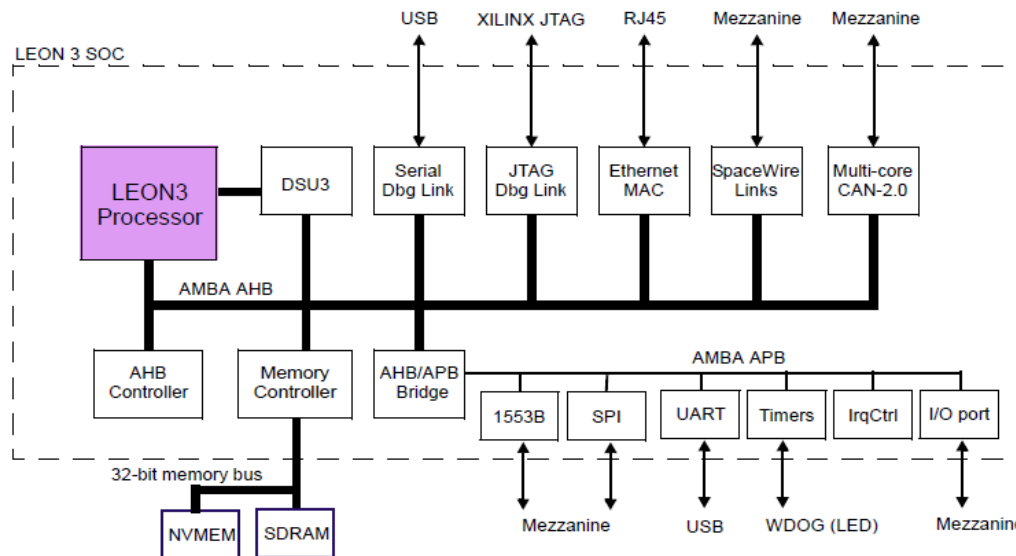


Figure 2.1: UT700 LEON 3FT Block Diagram

The design is centered on the AMBA AHB bus, to which the LEON 3 processor, AMBA APB, and other high-bandwidth devices are connected. External memory is accessed through a combined PROM/SDRAM memory controller. The on-chip peripheral devices include four SpaceWire links, one Ethernet 10/100 Mbit MAC, dual CAN-2.0 interfaces, one SPI interface, a dual channel 1553B interface, serial and JTAG debug interfaces, a UART, interrupt controller, timers and a 16-bit general purpose I/O port.

2.2 UT700 LEON 3FT SPARC V8 Processor

The LEAP's UT700 design is based the LEON 3 SPARC V8 processor. The processor core is configured with a cache system consisting of 4Kbyte, 4-way set associative Instruction and 4Kbyte, 4-way Data cache. The LEON3 debug support unit (DSU3) is a user port for downloading and debugging of programs through the serial or JTAG ports.

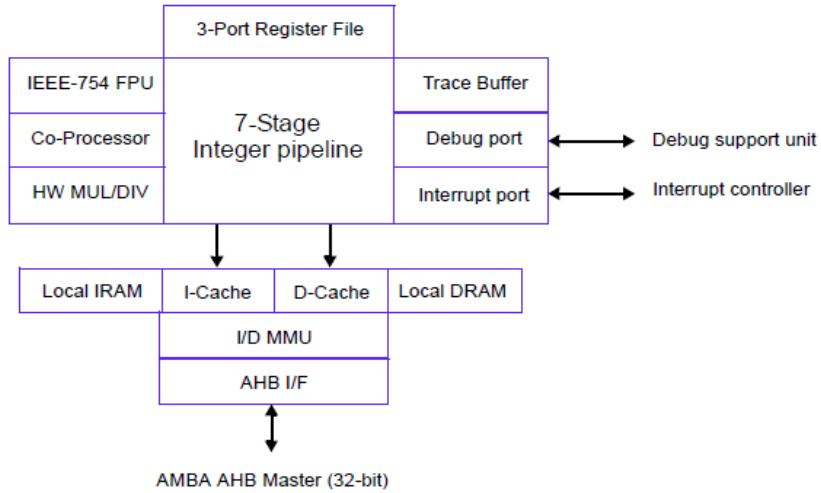


Figure 2.2: UT700 LEON 3FT SPARC V8 Processor Core Block Diagram

2.3 Memory Interfaces

The external memory is interfaced through a combined PROM/SDRAM memory controller core (UT700). The CAES LEAP card provides 8 MB of Non-Volatile memory and 32 MB of SDRAM for system program/data memory.

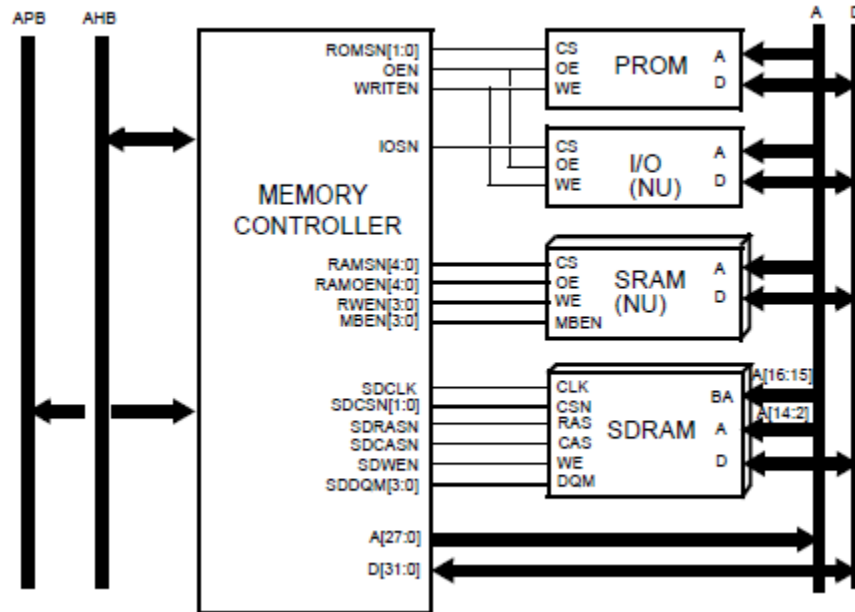


Figure 2.3: PROM/SDRAM Memory Interfaces

2.4 SpaceWire Links

The LEAP design is configured with four SpaceWire links. Each link is controlled separately through the APB bus, and transfers received and transmitted data through DMA transfer on AHB. All four of the SpaceWire links can be configured with RMAP support and each of the four SpaceWire Ports are routed to the mezzanine connector.

Note: SPWCLK is connected to LEONCLKX

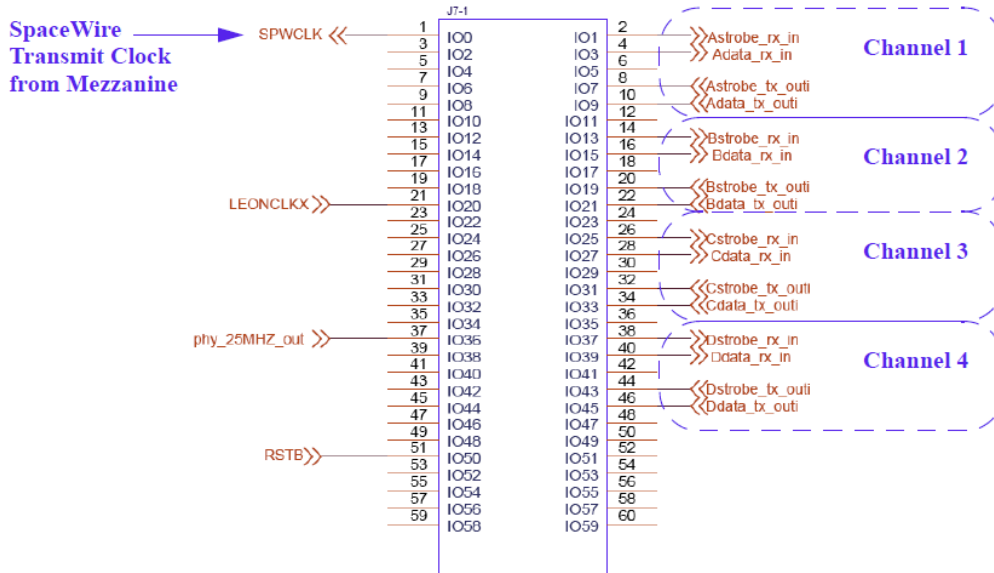


Figure 2.4: SpaceWire Mezzanine Interface Pinout

2.5 UT700 Timer Unit

The timer unit consists of a common scaler and up to four individual timers. The timers can work in periodical or one-shot mode. If configured to do so, Timer 4 also operates as a watchdog timer, driving the watchdog output signal (WDOG) when expired. The WDOG output signal is connected to a LED (see Figure 2.5) and will illuminate if Timer 4 is configured as the WDOG timer and expires.

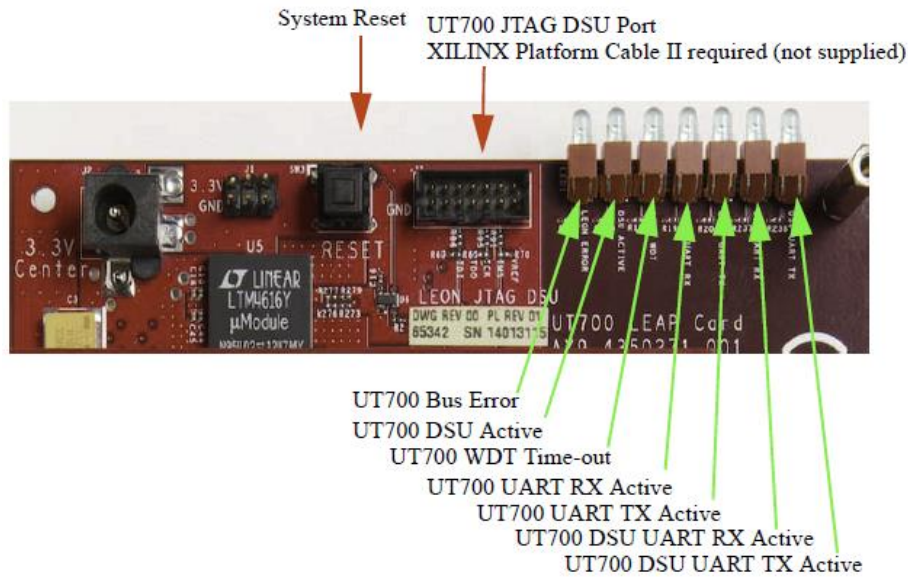


Figure 2.5: LEAP Reset and LED Indicators

2.6 System Reset Button

A method to generate a manual system-level reset to the LEAP card is provided through a momentary push-button switch (see [Figure 2.5](#)). The user can issue a system reset to the LEAP card through a simple press-and-release action on the System Reset switch

2.7 System Status LED

Several LEDs are provided as a visual confirmation/detection of different LEAP card board status conditions (see [Figure 2.5](#))

- The most basic of the indicators are the DSU and UART TX/RX LED's. The user should see these LEDs pulse on and off when transfers are occurring on the serial DSU or serial UART USB ports.
- The DSU Active LED is connected to the UT700's DSU Active output pin and will illuminate when the UT700 detects the DSU Enable switch is in the active (down position) and confirms it has enabled the internal DSU.

2.8 USB General Purpose UART I/F

The internal LEON UART is connected to a standard mini-USB connector through a standard USB bus transceiver. The UART can be used as a general-purpose serial I/O port (see [Figure 2.5](#))

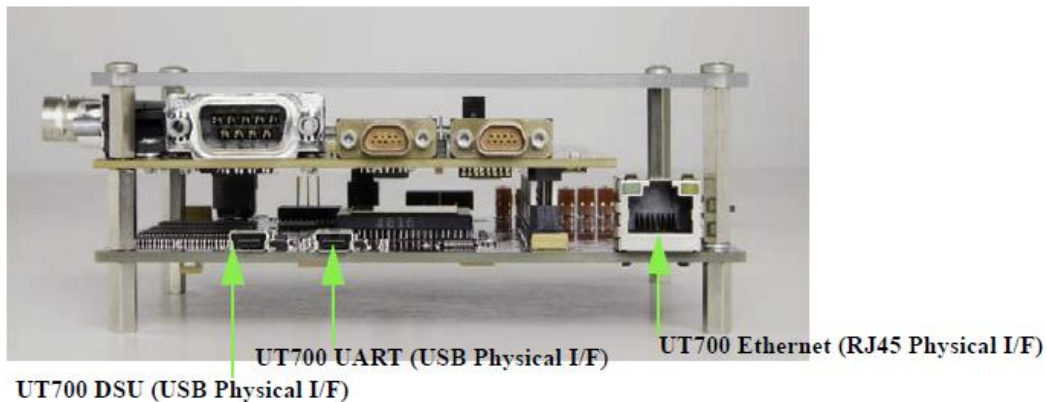


Figure 2.6: USB and Ethernet Interfaces

2.9 UT700 Ethernet Interface

The internal LEON Ethernet (10/100Mbps) I/F is connected to a standard RJ45 connector through a standard Ethernet PHY (see [Figure 2.6](#)). An additional feature of the UT700 Ethernet MAC is the built-in EDCL (Ethernet Debug Communication Link) protocol. The user can enable/disable the EDCL function by placing the piano switch as shown in [Figure 2.7](#).

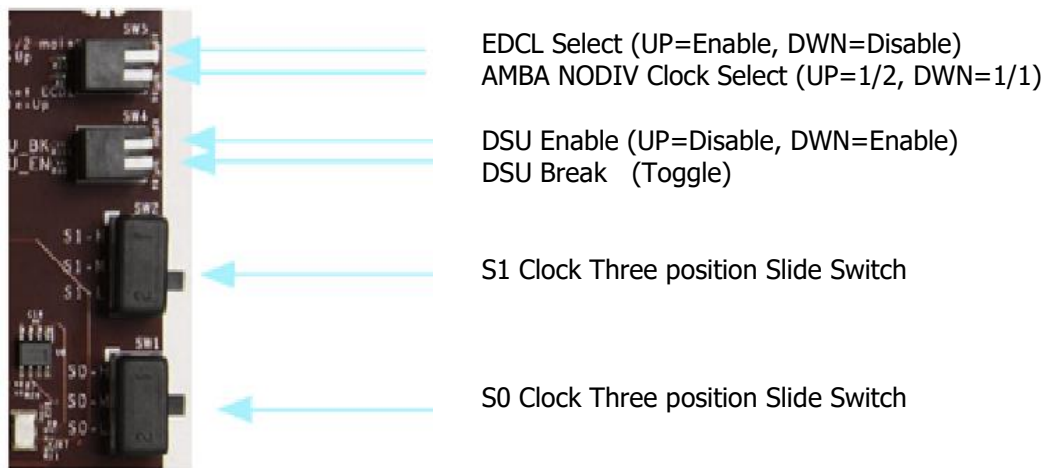


Figure 2.7: UT700 Mode and System Clock Adjust Selection

2.10 System Clock Selection

Two switches (S0 and S1) control the main clock frequency input to the UT700 (see **Figure 2.7**). For this application, the maximum operating clock that can be selected for the UT700 with the AMBA NODIV piano switch (see **Figure 2.7**) set to the down (1/1) position is 133MHz. The positions listed in **Table 2.1** outline the valid **S0** and **S1** position combinations and corresponding UT700 AMBA frequencies.

Table 2.1: System Clock Selection

S1 Position	S0 Position	NODIV	AMBA Frequency
H	M	Up/Down	37.50 MHz/75.00 MHz
M	H	Up/Down	41.67 MHz/83.33 MHz
L	H	Up/Down	62.50 MHz/125.0 MHz
L	M	Up/Down	66.65 MHz/133.3 MHz
H	L	Up/Down	75.00 MHz/Not Valid

Note: The LEAP board is designed to operate at a maximum bus frequency of 125MHz.

2.11 UT700 General Purpose I/O

A general purpose I/O port (GPIO) is provided in the design. The port is 16-bits wide and each bit can be dynamically configured as input or output. The GPIO can also generate interrupts from external devices. The 16-bit GPIO port is connected to the mezzanine connector enabling the user to define the function & interface controlling each pin. See **Figure 2.8**.

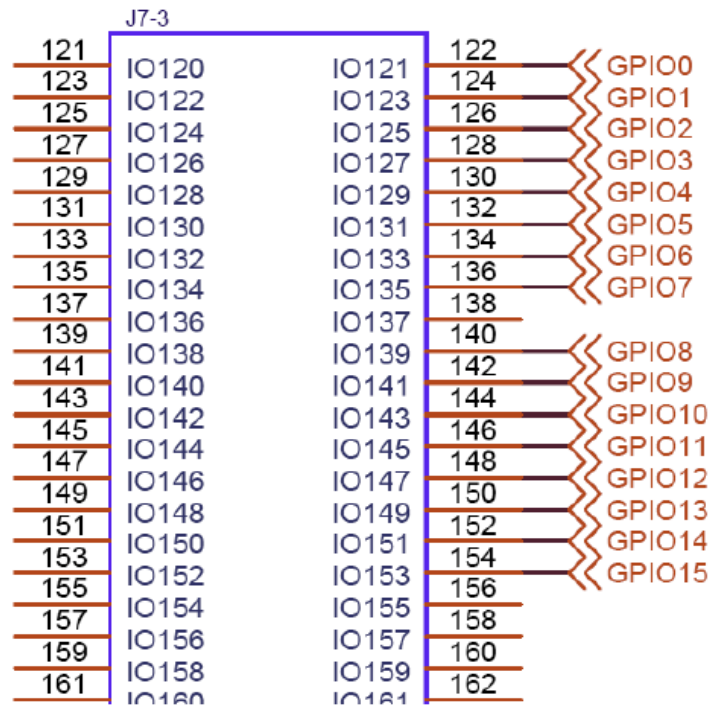


Figure 2.8: UT700 GPIO I/F to Mezzanine Connector

2.12 UT700 1553B Port to Mezzanine

The UT700's dual-redundant (Channel A and B) 1553B interface is routed to the **J7** mezzanine connector as shown in **Figure 2.9**.

Note: the 1553 Clock pin from the mezzanine is a 3.3V logic level input to the UT700 and should be a 20MHz 50% duty-cycle clock if the 1553B port is to be implemented (see UT700 datasheet for details).

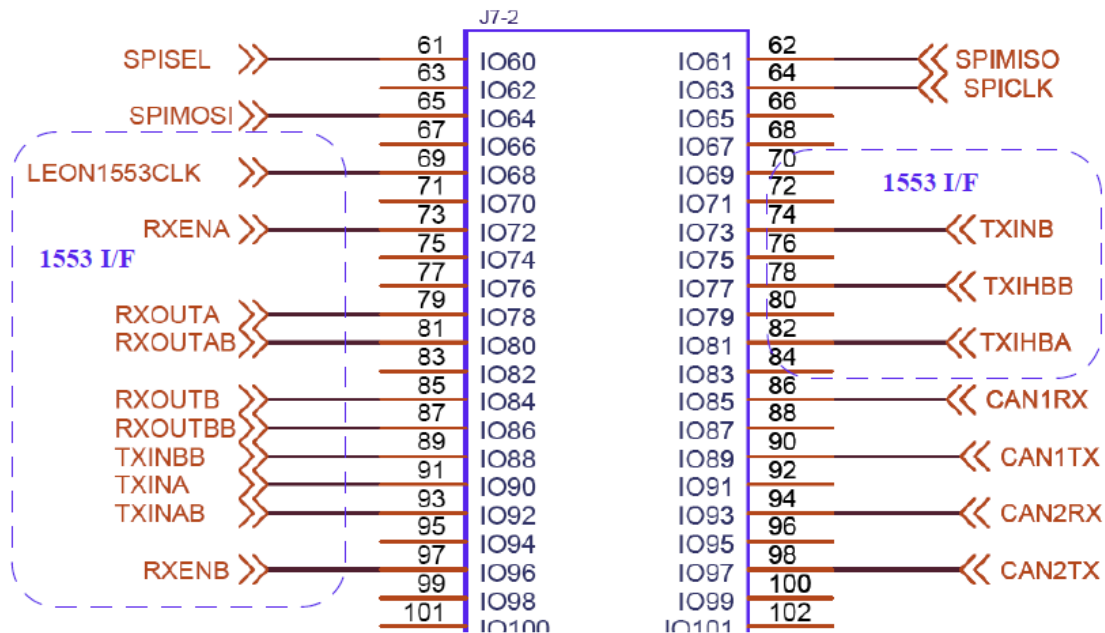


Figure 2.9: UT700 1553B I/F to Mezzanine Connector

2.13 UT700 CAN Bus Interface

Two CAN-2.0 interfaces are available through the mezzanine interface connector **J7**. The voltage levels are standard 3.3VDC single-ended signal type. See **Figure 2.10**.

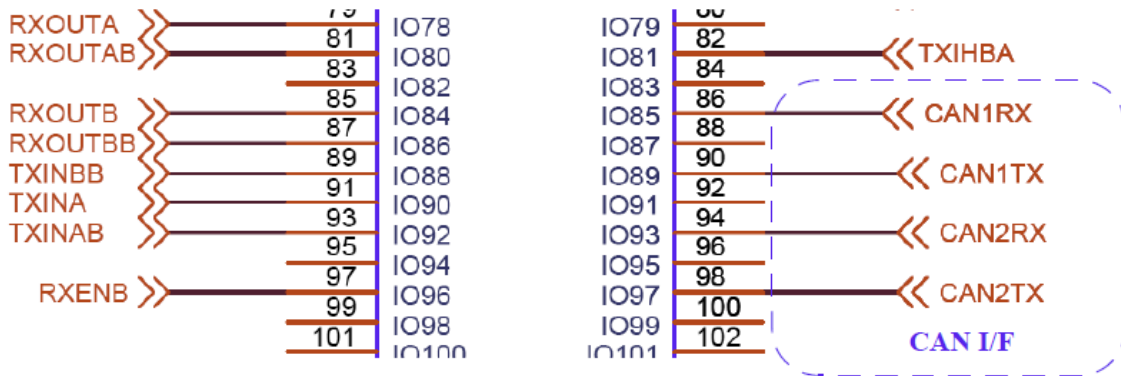


Figure 2.10: UT700 CAN Bus I/F to Mezzanine Connector

2.14 UT700 SPI Bus Interface

The UT700 SPI bus is routed to the mezzanine connector **J7** as shown in **Figure 2.11**. The voltage levels used by the SPI interface are standard 3.3VDC single-ended signal type.

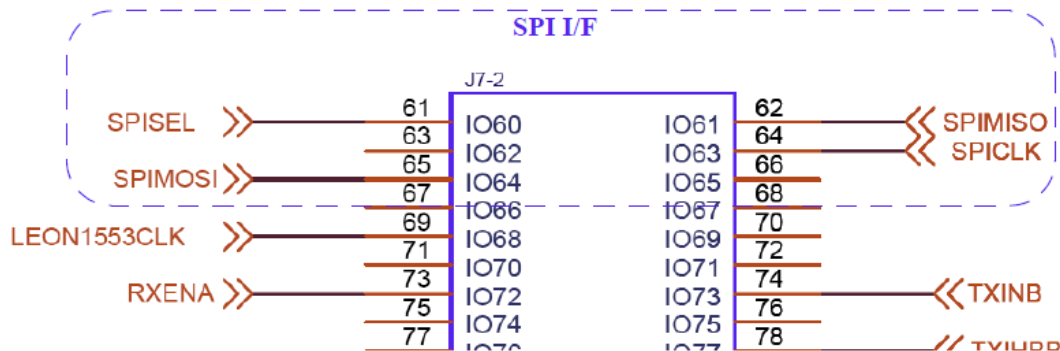


Figure 2.11: UT700 SPI Bus I/F to Mezzanine Connector

2.15 LEAP Core

The CAES UT700 SBC design is comprised of various core elements. **Figure 2.12** the initial screen-shot and core listing from a GRMON2 debug session invoked on the LEAP card through the JTAG DSU.

```

Command Prompt (2) - grmon -xilusb
0 File(s)          0 bytes
4 Dir(s)  259,869,982,720 bytes free

C:\grmon2\win32>cd bin
C:\grmon2\win32\bin>grmon -xilusb

GRMON2 LEON debug monitor v2.0.39 pro 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

Xilusb: Cable type/rev : 0x3
JTAG chain (1): UT699A
Detected system:      UT699E/UT700
Detected frequency:  133 MHz

Component                               Vendor
LEON3-FT SPARC V8 Processor             Aeroflex Gaisler
AHB Debug UART                          Aeroflex Gaisler
JTAG Debug Link                         Aeroflex Gaisler
Fast 32-bit PCI Bridge                  Aeroflex Gaisler
PCI/AHB DMA controller                 Aeroflex Gaisler
GR Ethernet MAC                        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
MIL-STD-1553B Interface                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 UART                           Aeroflex Gaisler
Multi-processor Interrupt Ctrl.        Aeroflex Gaisler
Modular Timer Unit                     Aeroflex Gaisler
Clock gating unit                      Aeroflex Gaisler
PCI Arbiter                            European Space Agency
General Purpose I/O port                Aeroflex Gaisler
AHB Status Register                    Aeroflex Gaisler
SPI Controller                         Aeroflex Gaisler
General Purpose Register                Aeroflex Gaisler

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

grmon2>

```

Figure 2.12: LEAP UT700 Core List

2.16 AHB Core Mapping

Once GRMON is online and the screen above displayed in the DOS window, typing the "info sys" command provides the AHB map and associated interrupt assignments of the UT700's core elements. An example system information screen is shown in [Figure 2.13](#).

```
grlib> info sys
00.01:053 Gaisler Research LEON3FT SEARC V8 Processor (ver 0x0)
  ahb master 0
01.01:007 Gaisler Research AHB Debug UART (ver 0x0)
  ahb master 1
  apb: 80000700 - 80000800
  baud rate 921600, ahb frequency 74.00
02.01:01c Gaisler Research AHB Debug JTAG TAP (ver 0x1)
  ahb master 2
03.01:014 Gaisler Research Fast 32-bit PCI Bridge (ver 0x0)
  ahb master 3, irq 3
  ahb: c0000000 - fff00000
  ahb: fff00000 - fff20000
  apb: 80000400 - 80000500
04.01:016 Gaisler Research PCI/AHB DMA controller (ver 0x0)
  ahb master 4
  apb: 80000500 - 80000600
05.01:01d Gaisler Research GR Ethernet MAC (ver 0x0)
  ahb master 5, irq 14
  apb: 80000e00 - 80000f00
  Device index: dev0
  edcl ip 192.168.0.64, buffer 2 kbyte
06.01:029 Gaisler Research GRSPW2 Spacewire Link (ver 0x0)
  ahb master 6, irq 10
  apb: 80000a00 - 80000b00
  Number of ports: 1
07.01:029 Gaisler Research GRSPW2 Spacewire Link (ver 0x0)
  ahb master 7, irq 11
  apb: 80000b00 - 80000c00
  Number of ports: 1
08.01:029 Gaisler Research GRSPW2 Spacewire Link (ver 0x0)
  ahb master 8, irq 12
  apb: 80000c00 - 80000d00
  Number of ports: 1
...
06.01:02c Gaisler Research Clock gating unit (ver 0x0)
  apb: 80000600 - 80000700
  GRMON did NOT enable clocks during initialisation
08.04:010 European Space Agency PCI Arbiter (ver 0x0)
  apb: 80000800 - 80000900
09.01:01a Gaisler Research General purpose I/O port (ver 0x1)
  apb: 80000900 - 80000a00
0f.01:052 Gaisler Research AHB status register (ver 0x0)
  irq 1
  apb: 80000f00 - 80001000
01.01:02d Gaisler Research SFI Controller (ver 0x5)
  irq 18
  apb: 80100100 - 80100200
  FIFO depth: 16, 1 slave select signals
  Maximum word length: 32 bits
02.01:087 Gaisler Research General purpose registers (ver 0x0)
  apb: 80100200 - 80100300
grlib>
```

Figure 2.13: LEAP UT700 Core Information List

2.17 Non-Volatile Memory Mapping

The LEAP card offers the user access to 8Mbytes of contiguous Non-Volatile (NV) memory storage. The base address for the 32-bit wide PROM space is 0x0000_0000. The user has the ability to write data into the NV memory by setting the PROM write enable bit in the UT700 memory configuration register (see UT700 Datasheet for details).

Note: MRAM 4x MR4A16B

2.18 SDRAM Memory Mapping

The LEAP card offers the user access to 32Mbytes of contiguous volatile memory storage. The base address for the 32-bit wide SDRAM space is 0x4000_0000; all programs targeting the LEAP card should compile programs using this base address.

Note: SDRAM 1x IS42S32800D

Chapter 3: Mezzanine Power Interface

3.1 Mezzanine Connector Power

The LEAP card supplies **3.3 VDC** (0.75A max) and **2.5 VDC** (0.5A max) to the mezzanine connector; each supply shares common ground (**GND**) pins. The mezzanine power connector map/pin-out is shown in **Figure 3.1**.

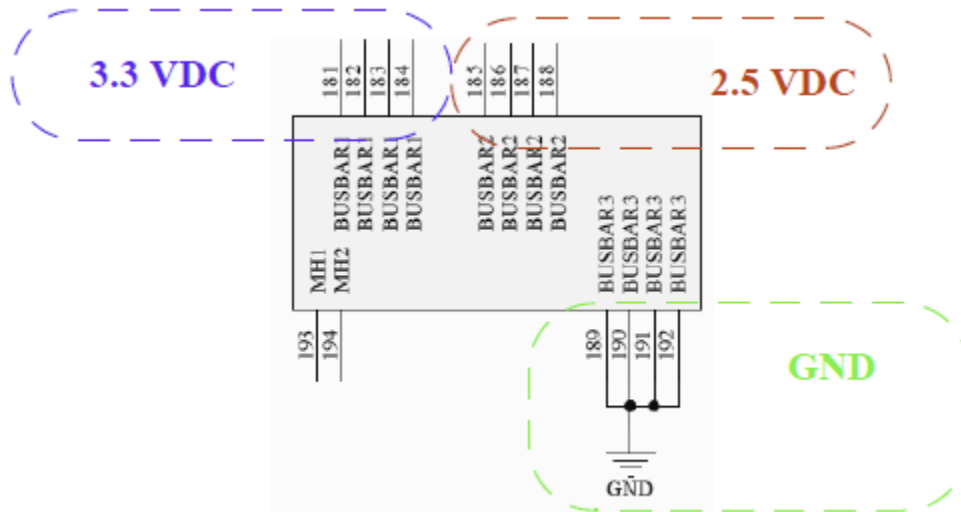


Figure 3.1: LEAP Power I/F to Mezzanine

Chapter 4: Optional LEAP with Mezzanine Card

- Base PN: 4250252-000

4.1 Mezzanine Card SPI Interface

The LEAP SBC card routes the UT700 SPI interface signals (**Figure 4.1**) to the mezzanine connector. The mezzanine card incorporates a 128x32 LCD Display (New Haven Display PN NHD-C12832A1Z-NSW-BBW-3V3. Since the LCD display does not have a SPI output port (input mode only), a jumper (P9) on the mezzanine can be used to loop-back the data sent by the UT700 SPI port to the display. When the **P9** Jumper is installed, the SPI data sent will be seen in the SPI data RX register.

Additionally, the LCD display requires two additional inputs for reset and A0 (command/data). The UT700 GPIO[15] is connected to the LCD A0 input; the UT700 GPIO[14] is connected to the LCD reset input.

Note: both GPIO[15:14] are pulled-low via 10K ohm resistors on the SBC card to hold the display in reset and A0 to a known state after power is applied to the system.

The New Haven Display Data Sheet contains additional information for the initialization and use of the graphic display.

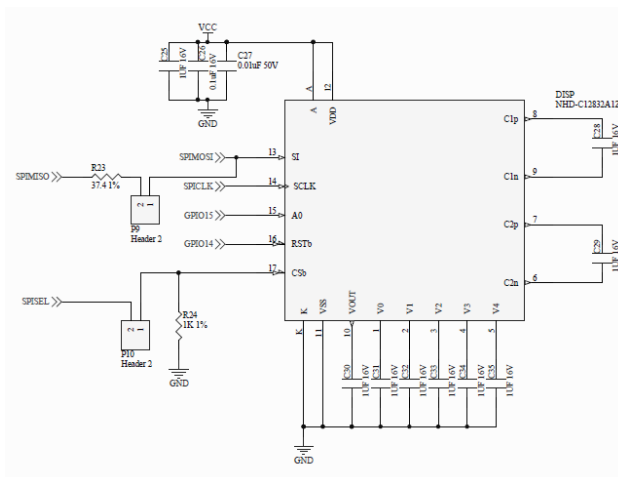
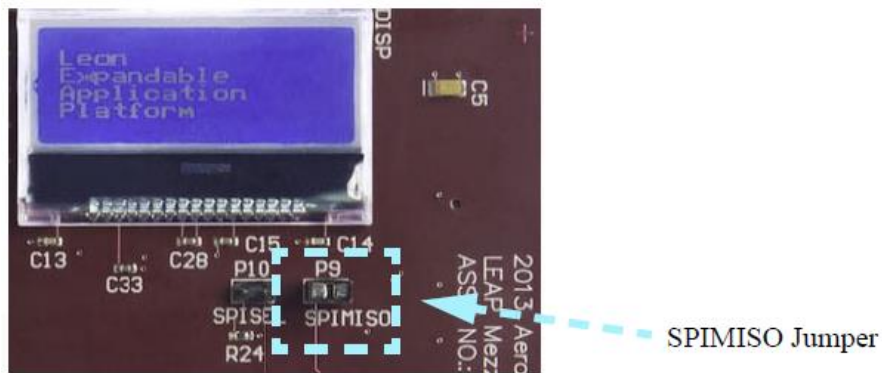


Figure 4.1: LEAP SPI I/F to Mezzanine

4.2 Mezzanine Card CAN Bus 1 Interface

The LEAP SBC card routes the UT700 CAN bus 1 and 2 interface signals to the mezzanine connector. The mezzanine card incorporates the CAN transceivers and standard 9-pin DSUB connectors along with the 120-ohm end-bus termination resistor. For CAN Bus 1, refer to **Figure 4.2**. The 120-ohm termination resistor can be installed (jumper on **P3**) or open (jumper on **P3** not installed). The optional ground connection to pin 5 of the DSUB connector can be selected by installing a block jumper on **P4** as shown in **Figure 4.2**.

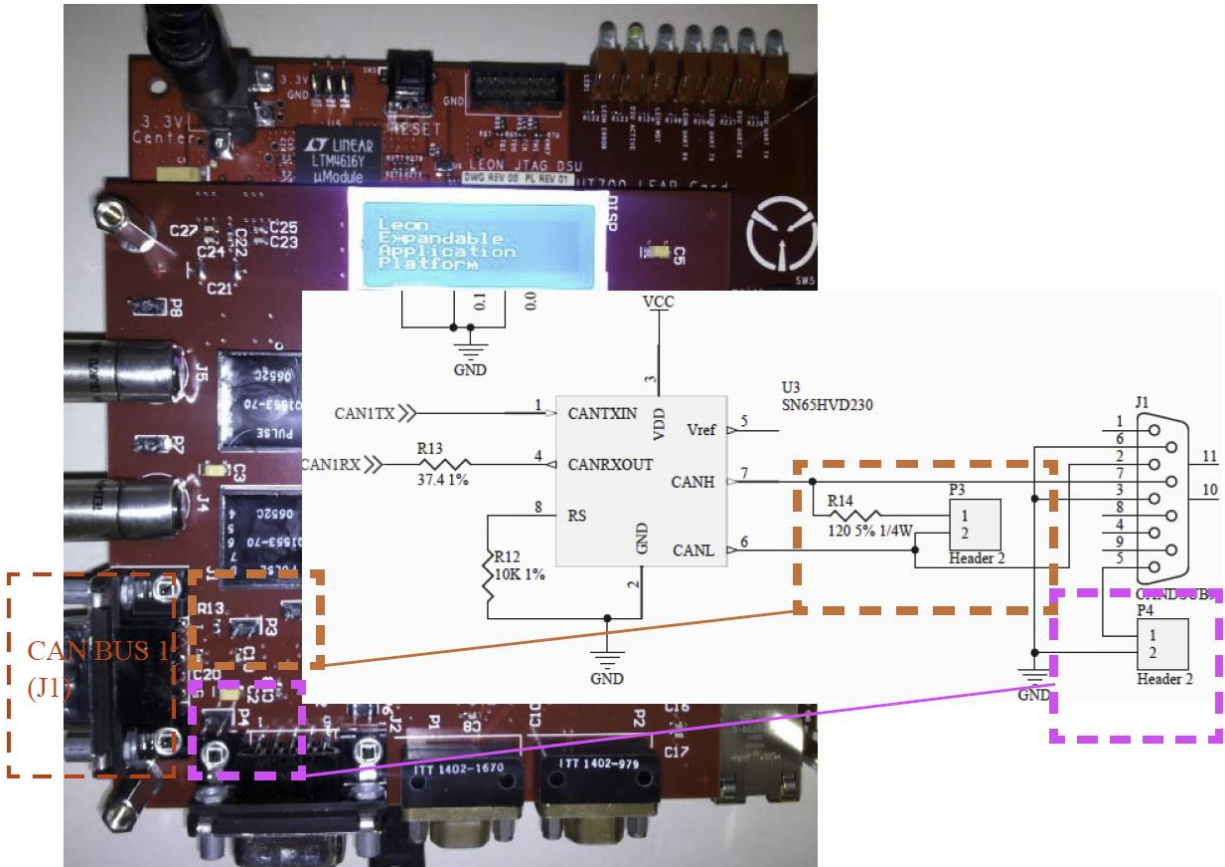


Figure 4.2: CAN Bus 1 Interface

4.3 Mezzanine Card CAN Bus 2 Interface

The LEAP SBC card routes the UT700 CAN bus 1 and 2 interface signals to the mezzanine connector. The mezzanine card incorporates the CAN transceivers and standard 9-pin DSUB connectors along with the 120-ohm end-bus termination resistor. For CAN Bus 2, refer to **Figure 4.3**. The 120-ohm termination resistor can be installed (jumper on **P5**) or open (jumper on **P5** not installed). The optional ground connection to pin 5 of the DSUB connector can be selected by installing a block jumper on **P6** as shown in **Figure 4.3**.

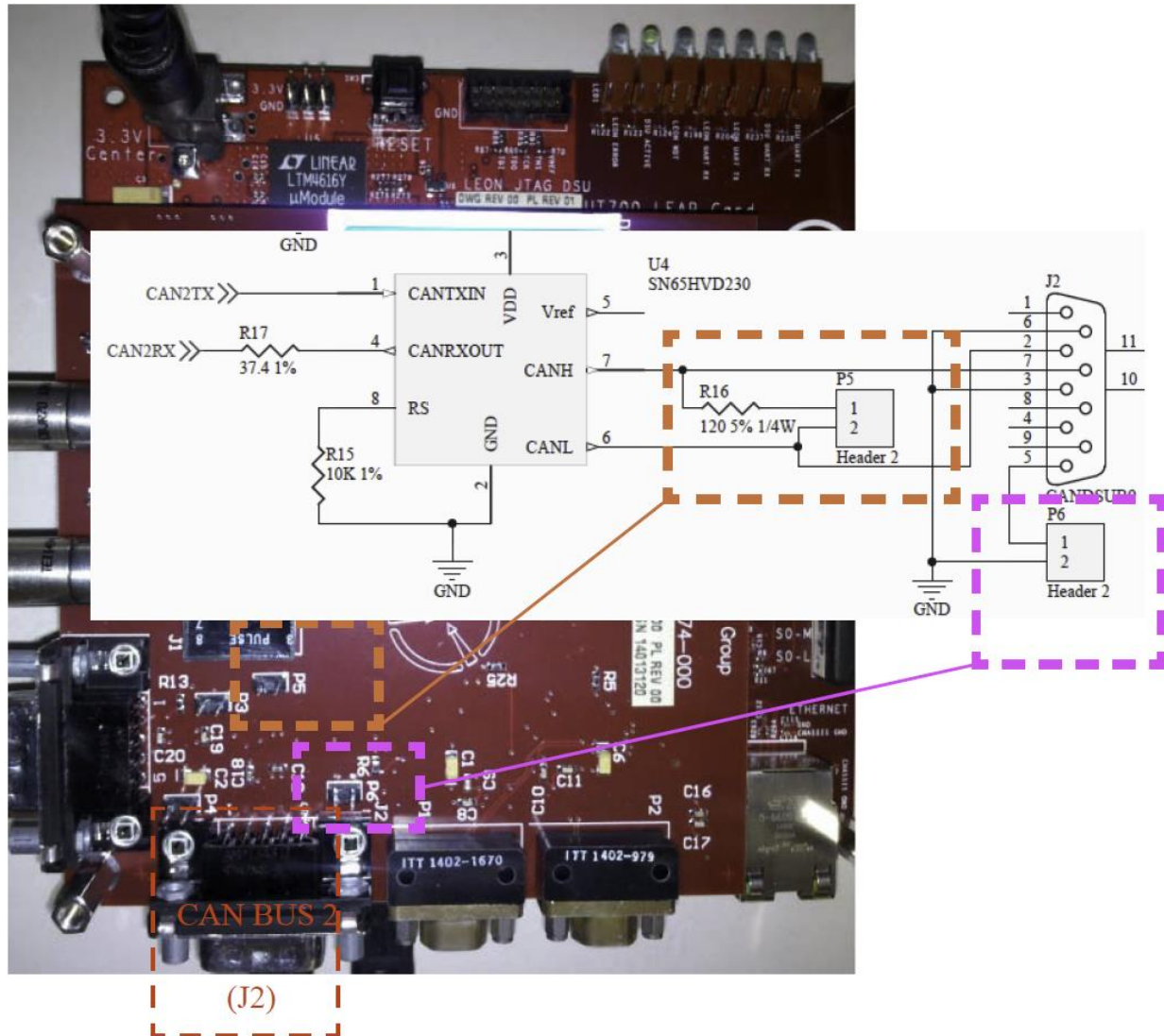


Figure 4.3: CAN Bus 2 Interface

4.4 Mezzanine Card 1553B Interface

The LEAP SBC card routes the UT700 1553B interface signals to the mezzanine connector. The mezzanine card incorporates a standard set of 1553B bus transceiver and transformers. For 1553B bus connector orientation, refer to **Figure 4.4**. The optional ground connection to the shield of the 1553B bus connector can be selected by installing a block jumper on **P7** for **CHA** and **P8** for **CHB** as shown in **Figure 4.4**.

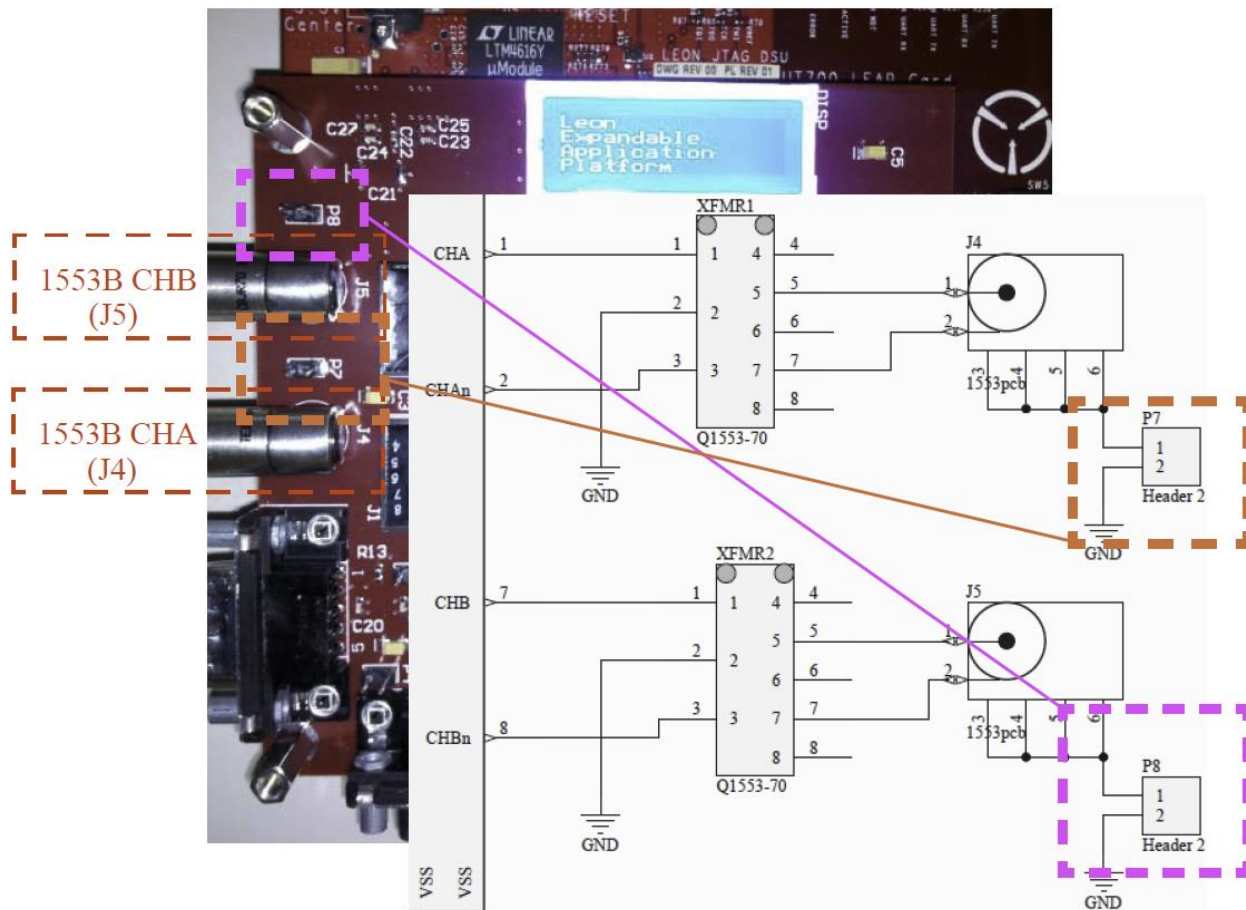


Figure 4.4: 1553B Bus Interface

4.5 Mezzanine Card SpaceWire Interface

The LEAP SBC card routes the UT700 SpaceWire interface signals to the mezzanine connector. The mezzanine card incorporates a standard set of SpaceWire transmitter/receiver interface devices ported to the 9-pin micro-D SpaceWire connectors. The SpaceWire bus connector orientation is shown in **Figure 4.5**. Only two of the four SpaceWire channels are provided on the mezzanine card. These channels are Port 0 (**P1**) and Port 1 (**P2**) of the UT700.

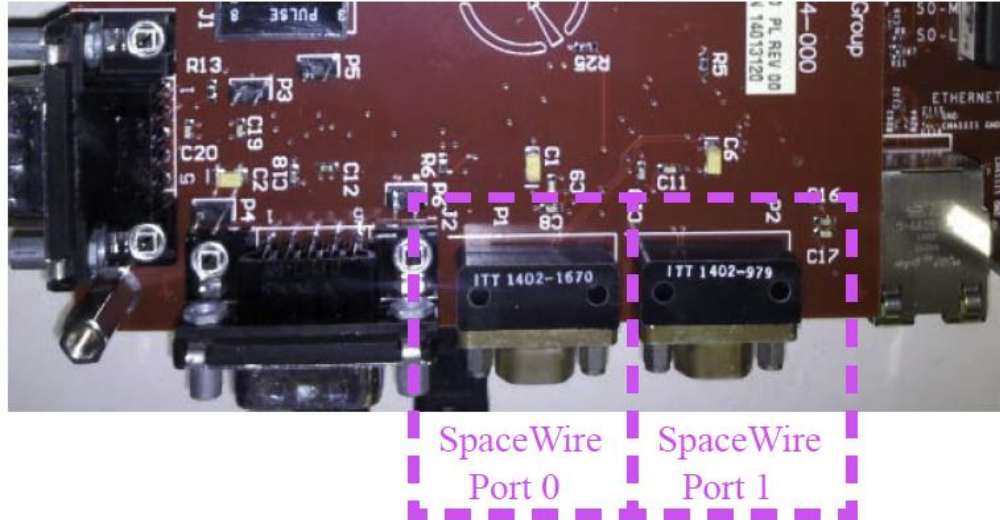


Figure 4.5: SpaceWire Bus Interface

Chapter 5: Software Development

5.1 “Out of the Box” Experience

The LEAP board comes preloaded with a “hello world” program, and after applying power to the board, it will display the “hello world” message on the Terminal, see [Figure 5.5](#). To display messages on the Terminal, the LEAP board connects over a USB serial connection (UT700 UART), see [Figure 2.6](#), with the following information.

Port number

Window: COM<port number>

Linux: /dev/ttyUSB<port number>

Connection Speed: Default **baud** for the LEAP board is **38,400**

Other options

Data: **8**

Stop bits: **1**

Parity: **none**

Flow control: **No**

5.2 Tool Chains

The LEON3 processor is supported by several software tool chains:

- Bare-C cross-compiler system (BCC)
- RTEMS cross-compiler system (RCC)
- Linux
- eCos real-time kernel
- VxWorks
- ThreadX

All these tool chains and associated documentation can be obtained from www.gaisler.com.

5.3 Downloading Software to the Target System

LEON3 has an on-chip debug support unit (DSU) which greatly simplifies the debugging of software on a target system. The DSU provides full access to all processor registers and system memory and also includes instruction and data trace buffers. Downloading and debugging of software is done using the GRMON debug monitor, a tool that runs on the host computer and communicates with the target through either serial or JTAG interfaces.

Please refer to the GRMON User’s Manual for a description of the GRMON operations at www.gaisler.com.

5.4 Bare-C cross-compiler system

The Bare-C (BCC) cross toolchain is a set of software tools for building computer software. The software is built on a host computer but builds programs to run on LEON3 processors, i.e., when developing applications for embedded systems. Besides the binary utilities, BCC includes the GDB source-level symbolic debugger.

BCC consists of the following packages:

- GNU GCC C/C++ compiler
- LLVM (Clang) C/C++ compiler
- GNU Binutils (assembler, linker, ...)
- Newlib embedded C library
- Low-level I/O routines for LEON3, including interrupt support
- GDB debugger
- Linux and Windows hosts

5.4.1 Installation

BCC needs a working native toolchain and build tools on the host PC.

To work with BCC on a host Linux computer, the following are system requirements:

Linux: Linux-2.6.x, glibc-2.11 (or higher)

The latest BCC cross toolchain is available at: www.gaisler.com/index.php/downloads/compilers

BCC is provided as a bziped tar-file. The compressed archive files should be unpacked in the /opt directory of the host.

```
$ mkdir /opt
$ cd /opt
$ tar xf ~/Downloads/sparc-elf-<version-number>.tar.bz2
```

After the installation, set the environment variable by adding /opt/sparc-elf-<version-number>/bin to the PATH variable.

```
$ export PATH=/opt/sparc-elf-<version-number>/bin:$PATH
```

5.4.2 Application development for BCC

BCC provides all the different phases that all programs go through: compiler, assembler and linker. The compiler transforms the C program into an assembly language program. The assembler transforms the assembly language into an object file, which contains the file header, text segment, data segment, relocation information and debugging information. Finally, the linker places code and data modules symbolically in memory; determines the addresses of data and instruction labels; and patches both the internal and external references.

BCC useful options for compiling applications are:

-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, -O3 or -Os	optimize code for maximum performance or minimal code size
-qsvt	use the single-vector trap model
-mtune=ut699	set UT699 specific parameters (gcc-3.4.4 and gcc-4.4.2)
-mfix-b2bst	enable workarounds for LEON3FT store-store errata (GRLIB-TN-0009)

- mv8 generate SPARC V8 mul/div instructions - needs hardware multiply and divide
- mflat do not use register windows (i.e. no save/restore instructions)

To use BCC cross toolchain, every command needs to prepend the target name, i.e., sparc-elf-gcc, when compiling applications with gcc, as the following line shows.

```
$ sparc-elf-gcc -O2 -msoft-float -mv8 -o helloworld helloworld.c
```

To run the application "hello world" on the target's RAM, attach grmon2, see 5.3, to the target board using the serial debug link (DSU), see Figure 2.6, Figure 5.1 and Figure 5.2.

```
$ grmon -uart /dev/ttyUSB0

grmon2> load helloworld
40000000 .text          23.6kB / 23.6kB [=====>] 100%
40005E70 .data          2.7kB / 2.7kB [=====>] 100%
Total size: 26.29kB (104.80kbit/s)
Entry point 0x40000000
Image /home/jaguas/projects/sparc-elf/apps/hello/helloworld loaded

grmon2> run
Program exited normally.

grmon2> load helloworld
40000000 .text          23.6kB / 23.6kB [=====>] 100%
40005E70 .data          2.7kB / 2.7kB [=====>] 100%
Total size: 26.29kB (104.80kbit/s)
Entry point 0x40000000
Image helloworld loaded

grmon2> run
Program exited normally.
```

Figure 5.1: Loading the application using grmon2

From Terminal Window:

```
Hello, World!
```

Figure 5.2: Application output from Terminal

5.4.3 Booting with mkprom2

Once the application is compiled with BCC, mkprom2 can create a boot-image to run from PROM on the LEON3 processor. Mkprom2 utility is freely available at:

www.gaisler.com.

To generate a boot-prom from the above compiled "hello world" application, use the following options as in

Figure 5.3.

```
mkprom2 -freq 50 -nosram -sdram 32 -memcfg1 0x1803caff
-memcfg2 0x89a06005 -memcfg3 0x08184000
-baud 38400 helloworld -o helloworld.prom

LEON2/3/ERC32 MKPROM prom builder for BCC, ECOS, RTEMS and ThreadX v2.0.60
Copyright Gaisler Research 2004-2007, all rights reserved.

creating LEON3 boot prom: helloworld.prom
Searching for compiler to use (sparc-elf, sparc-rtems or sparc-linux):
sparc-elf-gcc (BCC 4.4.2 release 1.0.44) 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.
```

Figure 5.3: Making the boot-prom image

To run the hello world app on the target’s PROM, attach grmon2, see [5.3](#), to the target board using the serial debug link (DSU), see [Figure 2.6](#), [Figure 5.4](#) and [Figure 5.5](#).

```
$ grmon -uart /dev/ttyUSB0

grmon2> mcfg1
mcfg1: 0x1803c2ff

grmon2> mcfg1 0x1803caff
mcfg1: 0x1803caff

grmon2> load helloworld.prom
00000000 .text          20.0kB/ 20.0kB [=====>] 100%
Total size: 19.97kB (111.59kbit/s)
Entry point 0x0
Image helloworld.prom loaded
```

Figure 5.4: Loading the boot-prom image

From Terminal Window:

```
MkProm2 boot loader v2.0
Copyright Gaisler Research - all rights reserved

system clock : 50.0 MHz
baud rate    : 38343 baud
prom         : 512 K, (2/2) ws (r/w)
sdram        : 32768 M, 1 bank(s), 9-bit column
sdram        : cas: 2, trp: 40 ns, trfc: 80 ns, refresh 7.8 us

decompressing .text to 0x40000000
decompressing .data to 0x40005e70

starting helloworld

Hello, World!
```

Figure 5.5: Output from Terminal

5.5 RTEMS

RCC is a multi-platform development system based on the GNU family of freely available tools with additional tools developed by Cygnus, OAR and CAES. RCC consists of the following packages:

- GCC C/C++ compiler
- GNU Binutils (assembler, linker, ...)
- RTEMS C/C++ real-time kernel with LEON support
- Newlib embedded C library
- GDB debugger
- Linux and Windows hosts

For RTEMS installation, see documentation available at

www.gaisler.com/anonftp/rcc/doc/rcc-1.2.pdf

5.5.1 RTEMS using Eclipse

Eclipse is an integrated development environment (IDE). It integrates cross toolchains and other tools, debuggers and plug-ins for managing software projects. For the LEAP board, Gaisler provides gdb on Eclipse for fully remote debugging applications running on the LEON3 processor. C/C++ development is supported on Eclipse using the C/C++ Development Tooling (CDT) extension.

For Eclipse installation, see the documentation available at

www.gaisler.com/index.php/products/compilers/51-section-software-category-ide/148-leon-cc-ide-for-eclipse

5.5.2 Configuring Eclipse for Cross Compilation

Open **MSYS**, change directory to **GRTools**, launch **Eclipse.exe** and on Workspace Launcher click OK (**Figure 5.6**).

```
peter@UXCLASS4 ~  
$ cd "C:\Program Files (x86)\GRTools"  
peter@UXCLASS4 /c/Program Files (x86)/GRTools  
$ eclipse.exe
```

Figure 5.6: MSYS Terminal

To create a new project, right-click on the **File** menu and select **New** -> **C project**, as pictures shows and click **Finish**, see **Figure 5.7**.

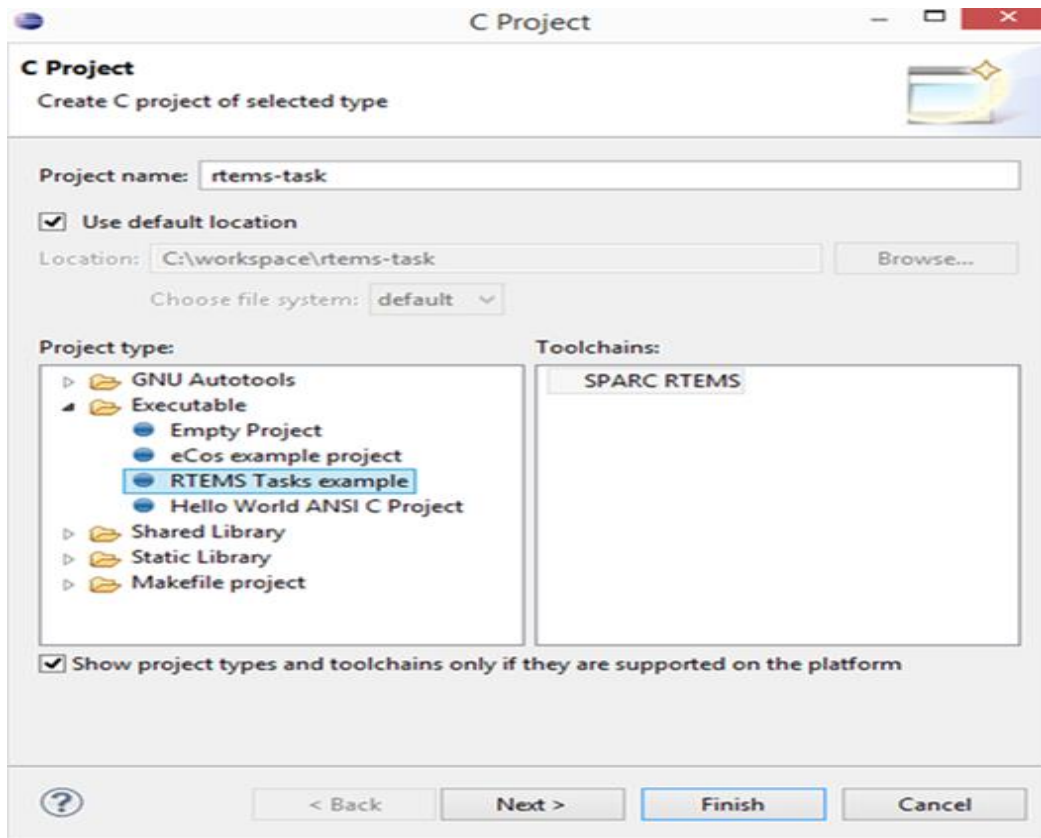


Figure 5.7: RTEMS project

Configure Eclipse to include RTEMS and CPU=v8 by selecting **Add** -> **File system...**, then browsing to include

the file shown in [Figure 5.5](#) and choose the settings shown in [Figure 5.8](#) and [Figure 5.9](#).

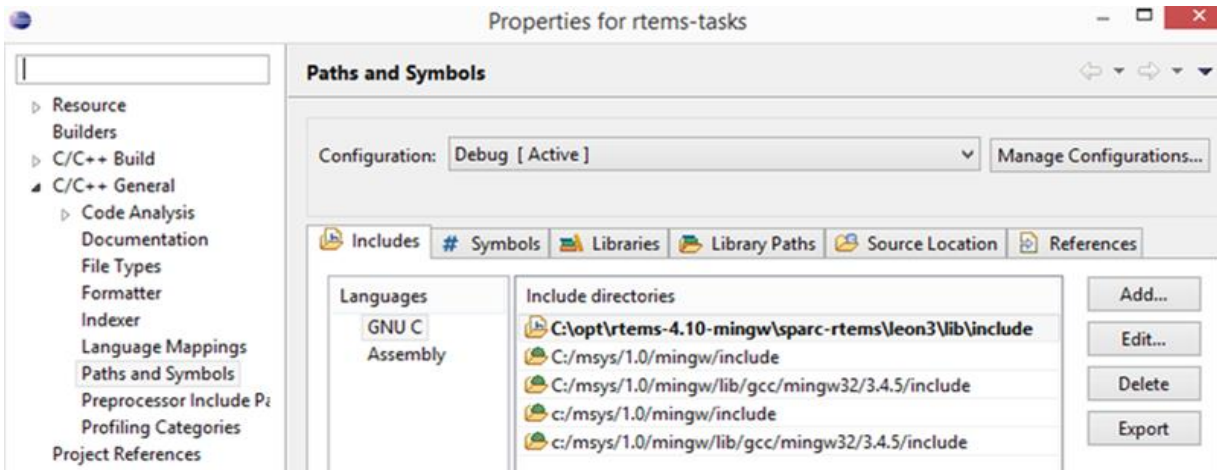


Figure 5.8: RTEMS include file

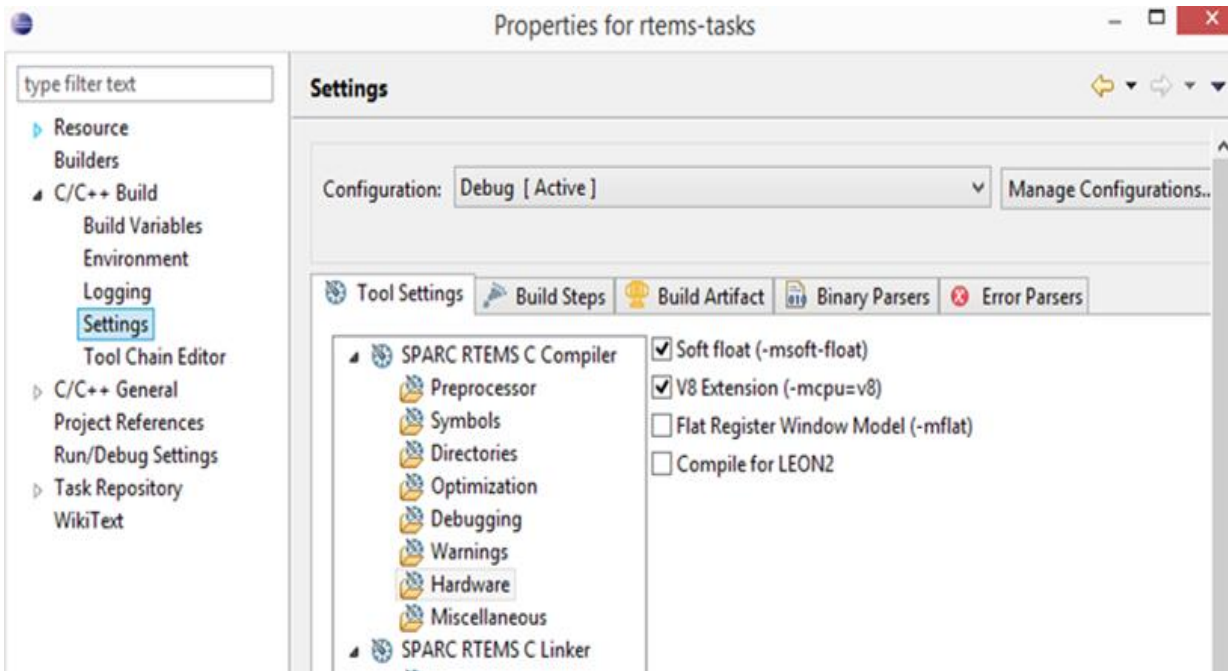


Figure 5.9: RTEMS SPARC V8

Once Eclipse is configured for cross compilation, right-click on **rtems-tasks** project and select **Build All**, see **Figure 5.10**.

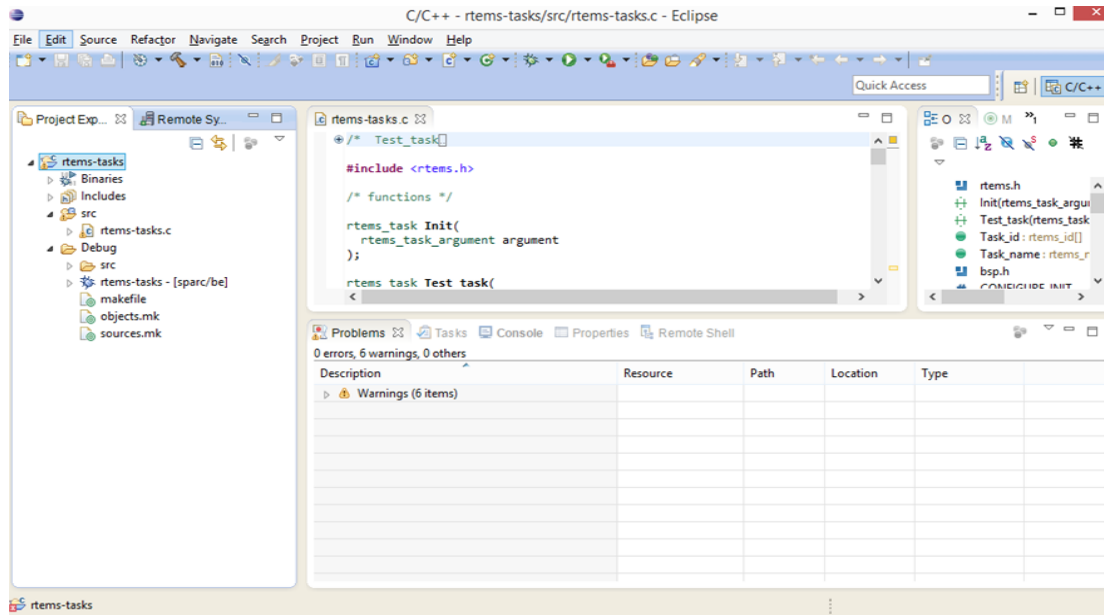


Figure 5.10: RTEMS project build

5.5.3 Remote Debugging with GDB

GDB makes it possible to remote debug with Eclipse by enabling control on execution steps and viewing debug messages and memory values directly from within Eclipse on the desktop.

Right-click on the **rtems-tasks** project and select **Run->External Tools ->External Tools Configurations...**, see **Figure 5.11**.

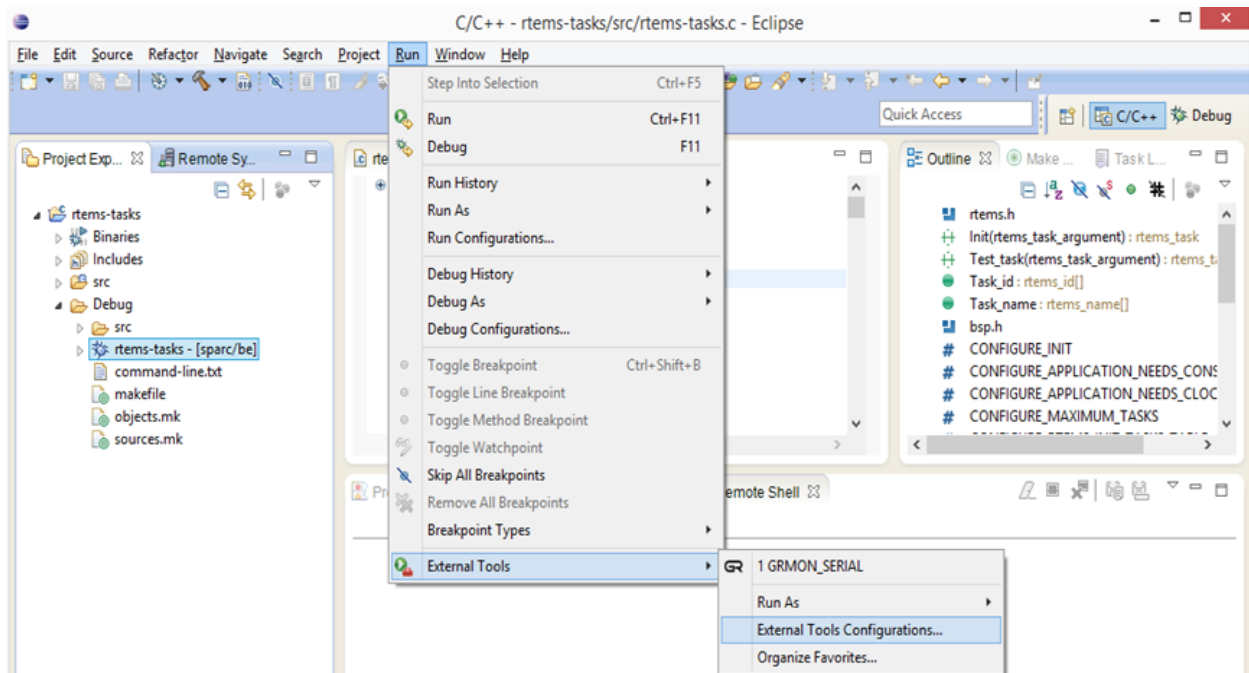


Figure 5.11: External Configurations

Specify the right com port < com3 > under **Device** and click **Run**, see **Figure 5.12**.

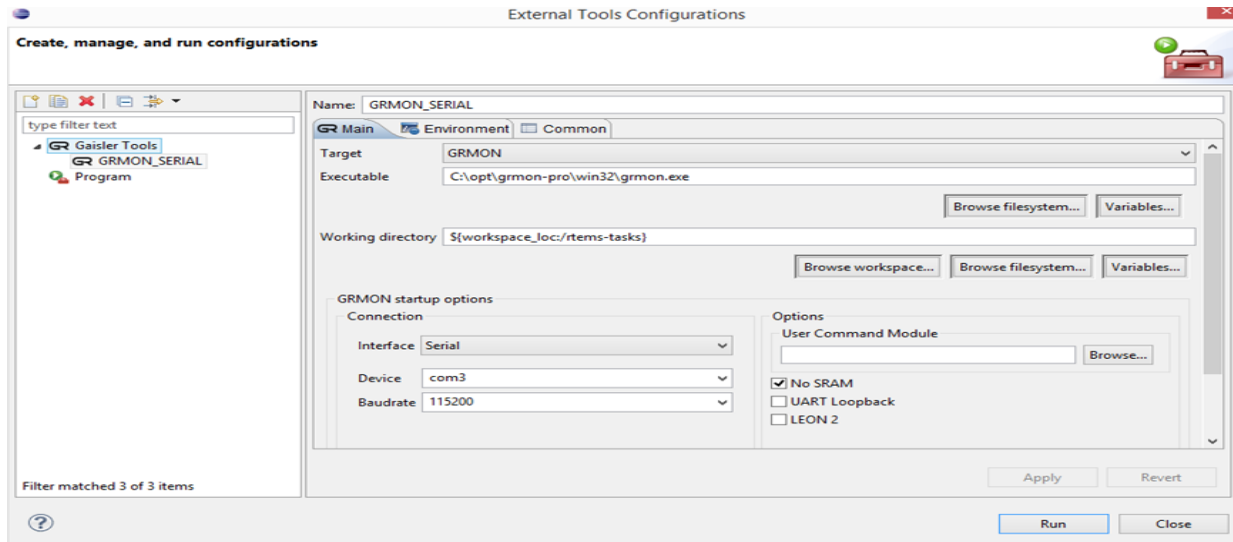


Figure 5.12: Grmon Terminal Settings

Remember the interface port number < using port 2226 >, see **Figure 5.13**.

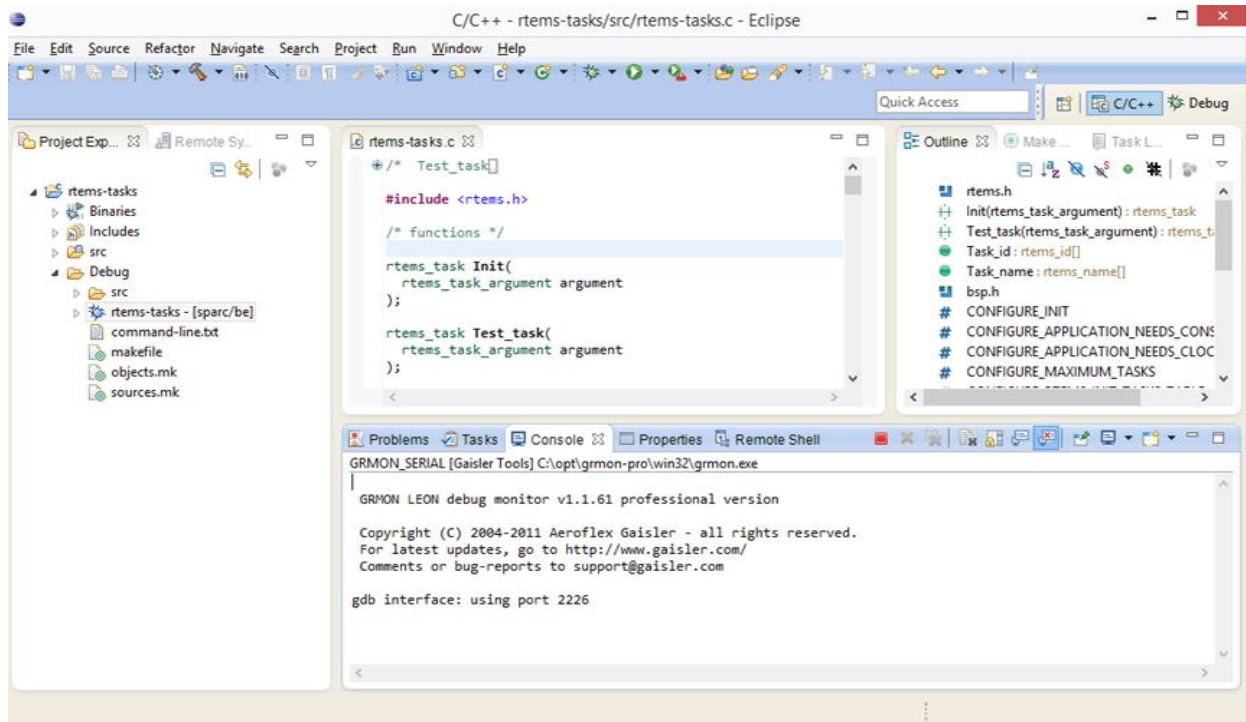


Figure 5.13: GDB Terminal display

Right-click on rtems-tasks project and select Debug As ->Debug Configurations..., see [Figure 5.14](#).

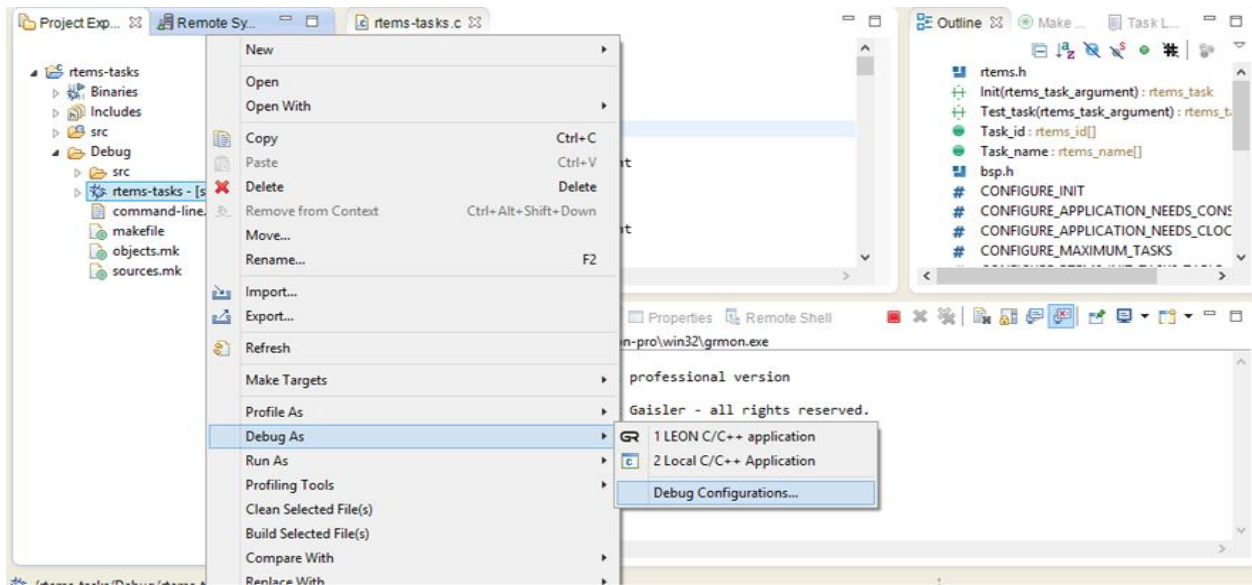


Figure 5.14: Debug Configurations

Port < 2226 > should match the target device's port. After the configuration has been set, click **Debug**, see [Figure 5.15](#).

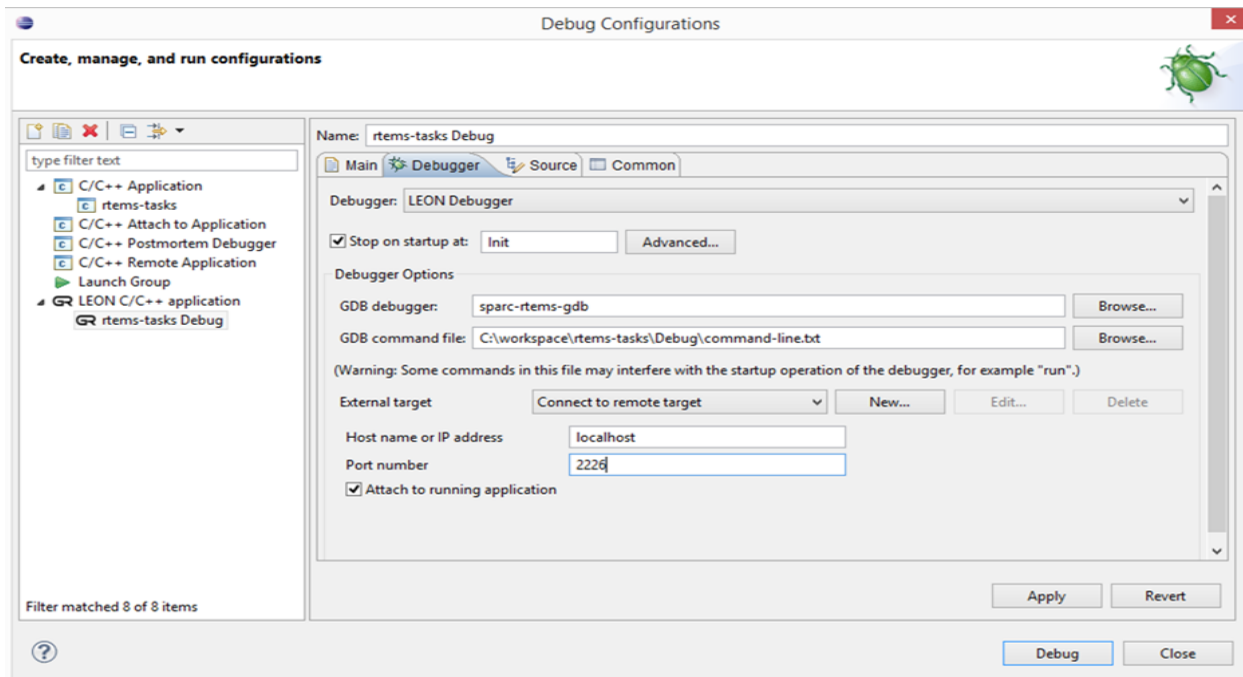


Figure 5.15: GDB Host Settings

Once **GDB** for the target and host have been configured, all debugging commands should work. Single **Step Over** a few lines, see [Figure 5.16](#).

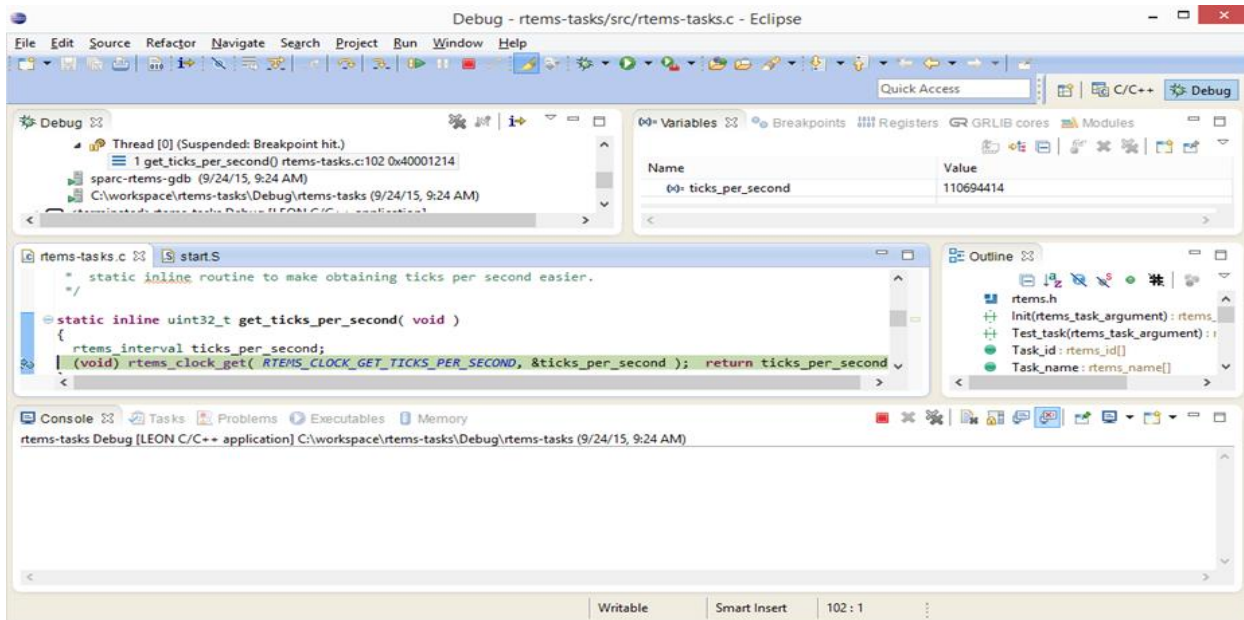


Figure 5.16: Debugger Perspective

5.5.4 Running RTEMS applications on the LEAP board

From the **Window** menu, select **Show View -> Other...**, see **Figure 5.17**.

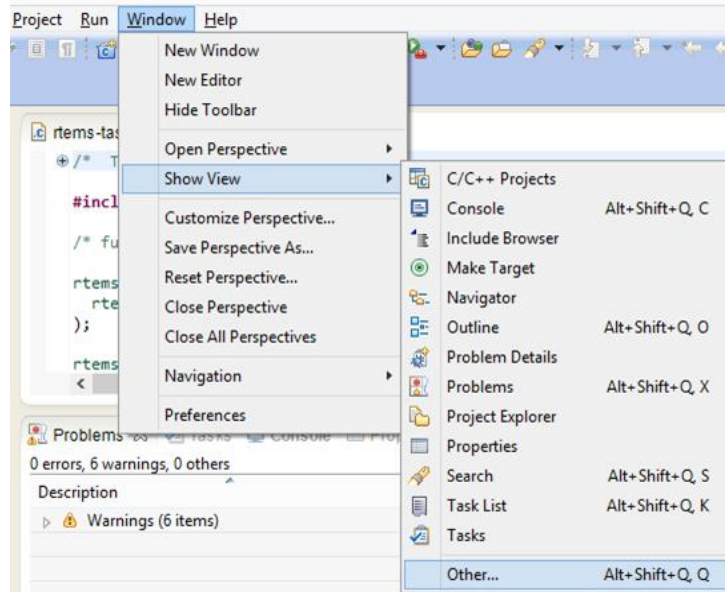


Figure 5.17: Selecting Remote Window

Choose **Remote Systems**, drag and drop it to **Project Explorer**, see **Figure 5.18** and **Figure 5.19**.

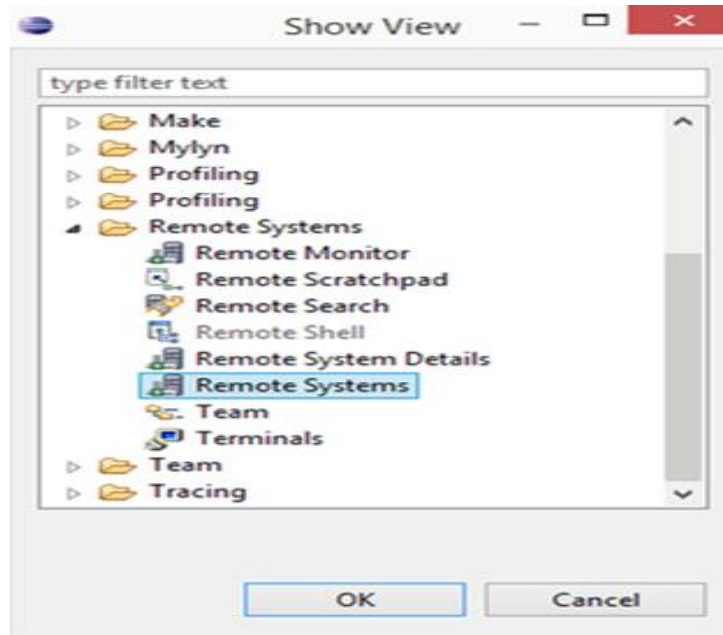


Figure 5.18: Remote System

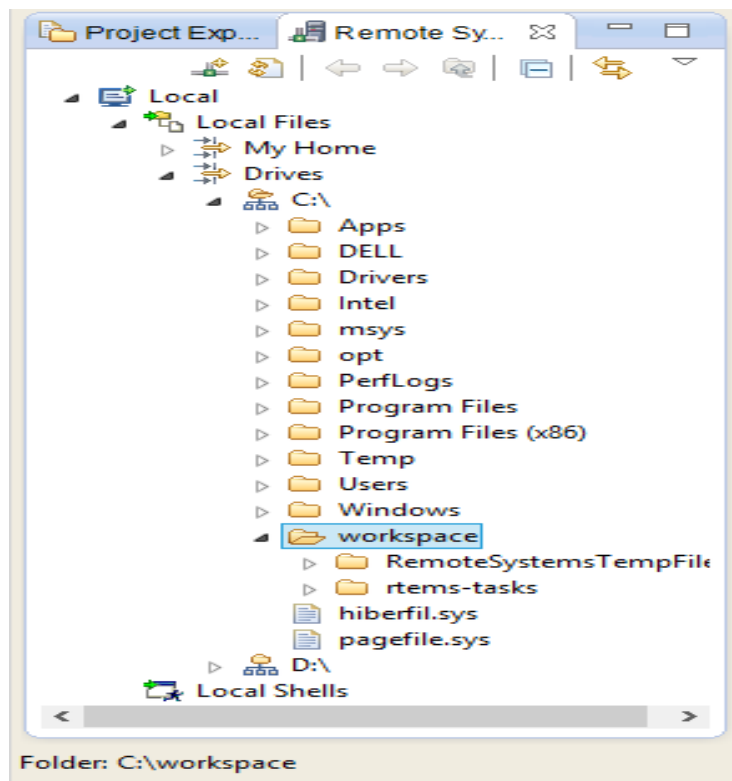


Figure 5.19: Local Project

Right-click on **workspace** and select **Launch Shell**, see [Figure 5.20](#).

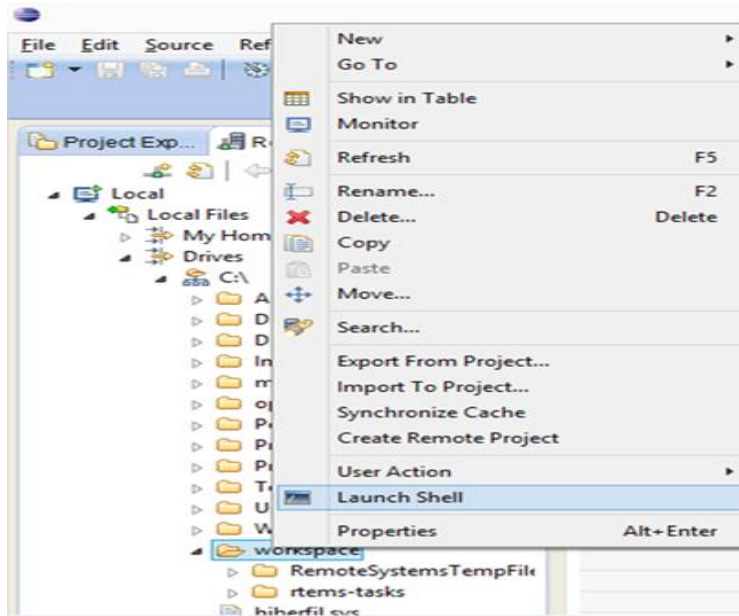


Figure 5.20: Shell

Change directories to <project>/Debug, see [Figure 5.21](#).

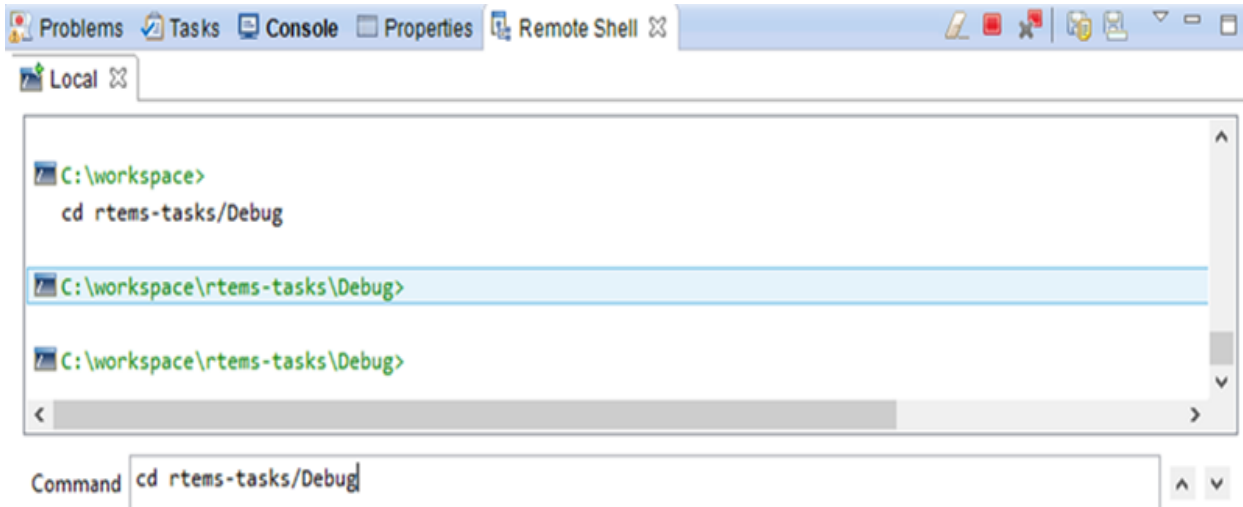


Figure 5.21: Project Debug Directory

Run **grmon**, load and run the program. The COM port is under **Device Manager**; see **Figure 5.22** to **Figure 5.24**.

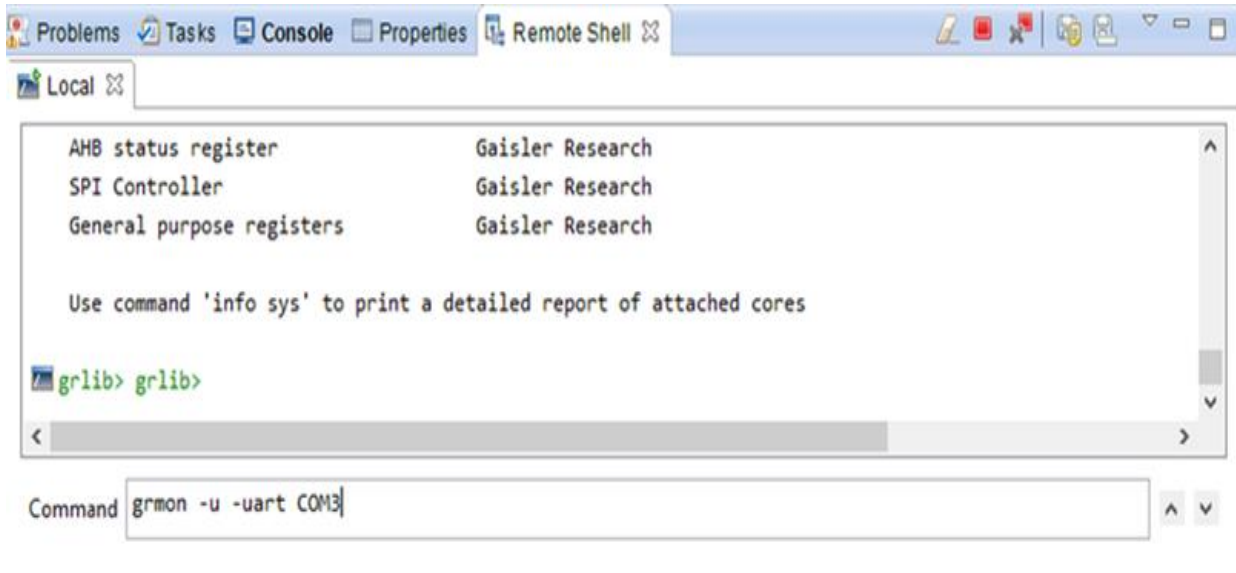


Figure 5.22: Grmon Connecting to Remote Target

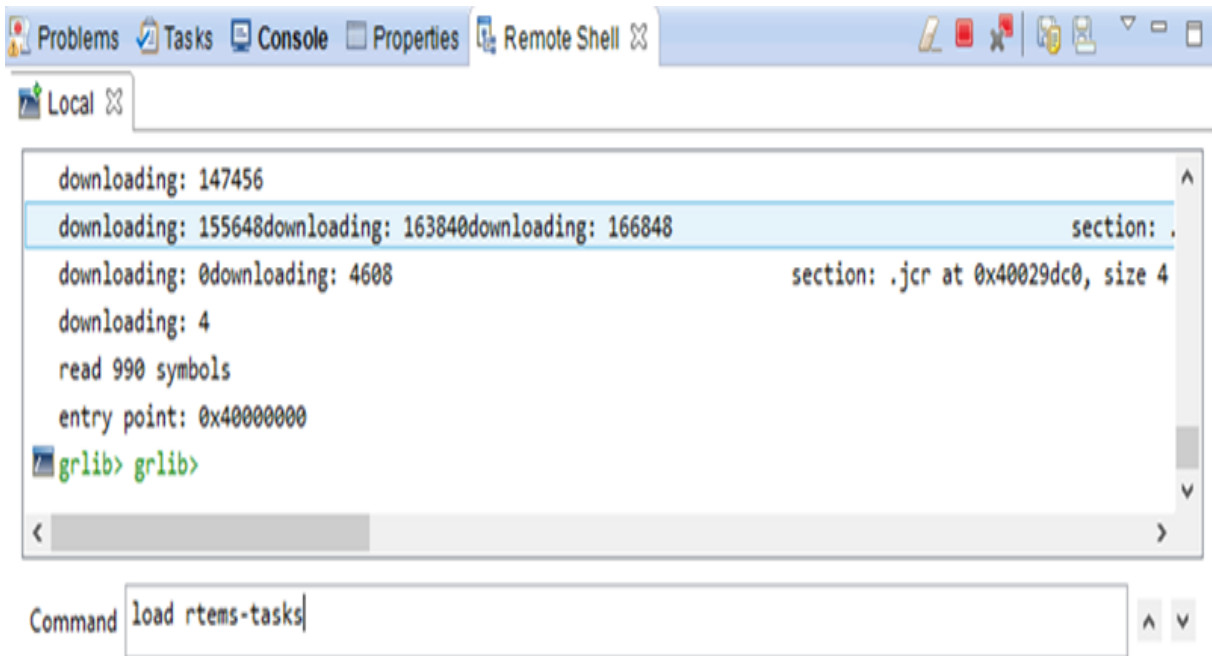


Figure 5.23: Loading Executable

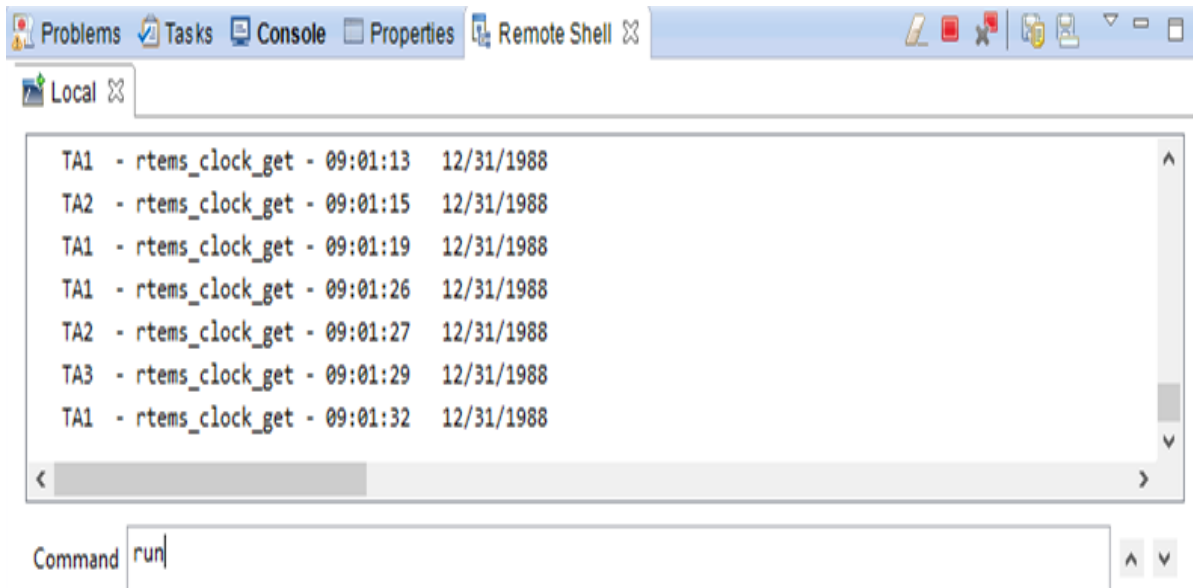


Figure 5.24: Running application

5.6 VxWorks

The VxWorks BSP contains a set of drivers for the SpaceWire core in the LEON3 SOC design. The operation of the driver is described in the VxWorks-drivers manual. The supported hardware is summarized in the list below. For documentation about a specific core's driver please see the LEON VxWorks Driver Manual.

CAES provides a SPARC/LEON architectural port for VxWorks and a UT700 BSP. The BSP contains drivers for all UT700 peripherals. Refer to www.gaisler.com for more information.

5.6.1 VxWorks Kernel

VxWorks kernel is configured by default with the target server, auto-memory configurations and some basic features. To add more components, use the project tool in Workbench.

To Start **Workbench**, double-click on the **Workbench** icon on the **desktop**. From the **File** menu select **New** -> **VxWorks Image Project**. Specify the project name as **LEAP_VxWorks_Kernel** and click **Next**.

Setup the project:

Based on **a board support package**

BSP: **gr_ut700leap**

Tool chain: **gnuv8**

Click **Finish**.

Include the following **VxWorks** components:

Spy, greth, ifconfig, external symbols, isr show, shell banner and ping.

In the **Project Explorer** view, right-click on **LEAP_VxWorks_Kernel** and select **Build Project**.

The build produces the **WxWorks** image, **vxWorks**, for the target system.

5.6.2 VxWorks Kernel Module

Create a new VxWorks Downloadable Kernel Module project. From the **File** menu select **New** -> **VxWorks Downloadable Kernel Module**. **LEAP_VxWorks_Events** for project name.

Active build spec: **SPARCgnuv8**

Click **Finish**.

Import **events.c** from **/where/downloaded/events.c** into **LEAP_VxWorks_Events** project.

- Right-click on the **LEAP_VxWorks_Events** project and select **Import....**
- Expand **General** and select **File System** from the **Import** dialog box.
- Click **Next**.
- Click **Browse...** on the **From Directory** field.
- Browse to **events.c** dir and click **OK**.
- Select **events.c** and click **Finish**.
- In the **Project Explorer** view, right-click on **LEAP_VxWorks_Events** and select **Build Project**.

The build produces the DKM module: **LEAP_VxWorks_Events.out**.

5.6.3 Booting with Bootrom

To build the **bootrom**, reference Gaisler's LEAP VxWorks BSP manual:

www.gaisler.com/doc/vxworks-ut700leap-bsp.pdf

To run the **bootrom** image on the target's PROM, connect the **LEAP board's DSU** serial port cable

To the host serial port, or DSU connection of preference and load **bootrom**, see **Figure 5.25**.

Windows: >grmon.exe -uart <com3>

Linux: \$ grmon -uart /dev/ttyUSB0

```
$ grmon -uart /dev/ttyUSB0

grmon2> mcfg1
mcfg1: 0x1803c2ff

grmon2> mcfg1 0x1803caff
mcfg1: 0x1803caff

grmon2> load bootrom.prom
00000000 .text 21.4kB / 21.4kB [=====>] 100%
00005590 .data 382.8kB / 382.8kB [=====>] 100%
Total size: 404.13kB (91.57kbit/s)Entry point 0x0
Image bootrom.prom loaded
```

Figure 5.25: Loading the boot-prom image

In Workbench, open the Terminal view, Window -> Show View -> Terminal, set the setting values as show in **Table 5.1**.

Table 5.1: Terminal Settings

Settings	Value
Baud Rate	38400
Data bits	8
Stop bits	1
Parity	None
Flow Control	None

After resetting the target, the **bootrom** application tries to load **VxWorks** image. To stop auto boot, press any key, see **Figure 5.26**.

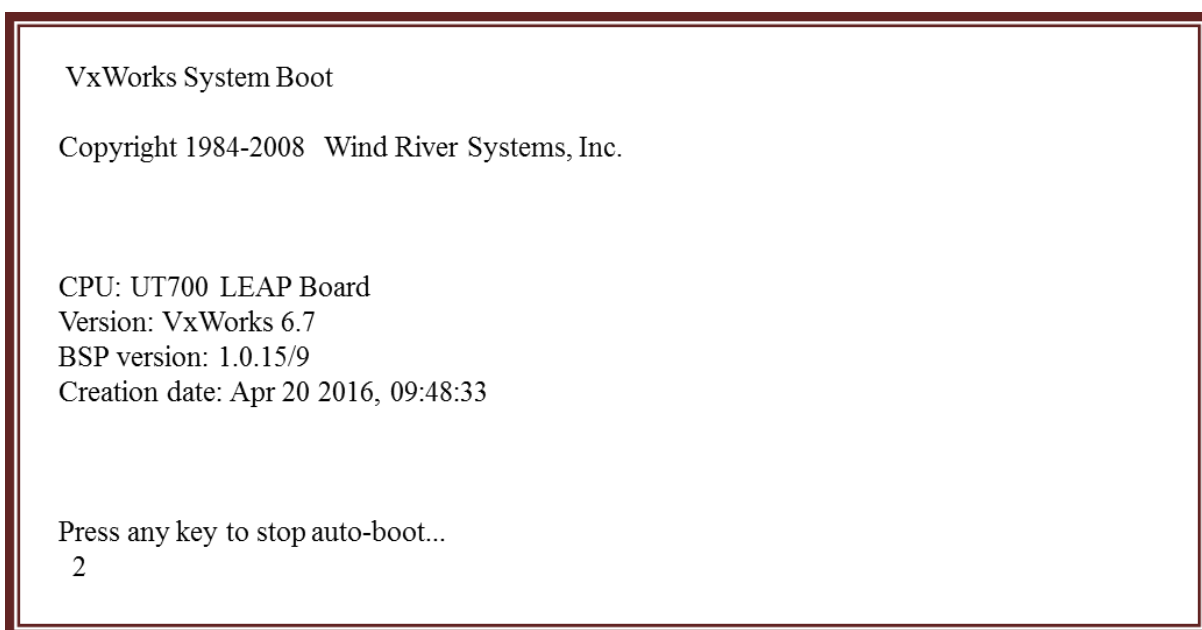


Figure 5.26: Loading the boot-prom image

In the **Terminal** view, type **p** to view the boot parameters, see [Figure 5.27](#).

```
[VxWorks Boot]: p

boot device           : greth
unit number          : 0
processor number     : 0
host name            : 10.5.233.136
file name            : vxWorks
inet on ethernet (e) : 10.5.233.215
host inet (h)        : 10.5.233.136
gateway inet (g)     : 10.5.233.1
user (u)             : anonymous
ftp password (pw)    : user
flags (f)            : 0x80
target name (tn)     : boottest
```

Figure 5.27: Boot parameters

To change the boot parameters to your particular values, type **c** at the boot prompt, see [Table 5.2](#).

Table 5.2: Useful keys

Key	Description
Backspace	Change last character typed
.	Clear current entry
-	Go back to last parameter
Ctrl+D	Return to boot parameter prompt
Ctrl-U	Erase current line and retain existing parameter

After setting the boot parameters, to boot the **VxWorks** system, type **@** at the boot prompt, see [Figure 5.28](#).

```
[VxWorks Boot]: @
boot device           : greth
unit number          : 0
processor number      : 0
host name             : 10.5.233.136
file name             : vxWorks
inet on ethernet (e) : 10.5.233.215:FFFFFF00
host inet (h)         : 10.5.233.136
gateway inet (g)      : 10.5.233.1
user (u)              : anonymous
ftp password (pw)     : user
flags (f)             : 0x80
target name (tn)      : boottest
```

```
Loading... 1358624 + 281376
Starting at 0x40003000...
```

```
Adding 5507 symbols for standalone.
```

```
CPU: UT700 LEAP Board . Processor #0.
Memory Size: 0x1fff000. BSP version 1.0.15/9.
Created: Feb 13 2017, 18:17:56
ED&R Policy Mode: Deployed
WDB Comm Type: WDB_COMM_END
WDB: Ready.
```

```
->
```

Figure 5.28: Booting the Target

5.6.4 Debugging with Workbench

Workbench is based on the Eclipse framework offering end-to-end, open standards-based suite for device software design, development, debugging, test and management.

More information on Wind River Workbench is available at www.windriver.com

5.6.5 Configuring the Target Server/Manager

With **VxWorks** running on the target, in the Remote Systems view toolbar click on **Define a connection to remote system** icon:





In the New Connection dialogue specify **VxWorks 6.x** -> Wind River VxWorks 6.x Target Server Connection as the **Remote System Type**, and click **Next**.

Backend: wdbrcp

Target name/IP address: 192.168.0.17

Change IP address <192.168.0.17> to your particular value.

Click **Finish**.

5.6.6 Debugging the Target Device

Note: there are at least 3 ways to accomplish the same task in Workbench.

In the **Project Explorer**, right-click on **LEAP_VxWorks_Events.out**, select **Download VxWorks Kernel Task**, and select your specific target to download to, i.e., **VxWorks6x_192.168.0.215**. In the **Remote Systems** view, you will see the **LEAP_VxWorks_Events.out** under you target connection.

From the **Remote Systems** view, debug a running task.

Right-click on **s12u0** and select **Debug** -> **VxWorks Kernel Task, Workbench** switches to **Device Debug** perspective and the attached task is listed in the **Debug** view, see [Figure 5.29](#).

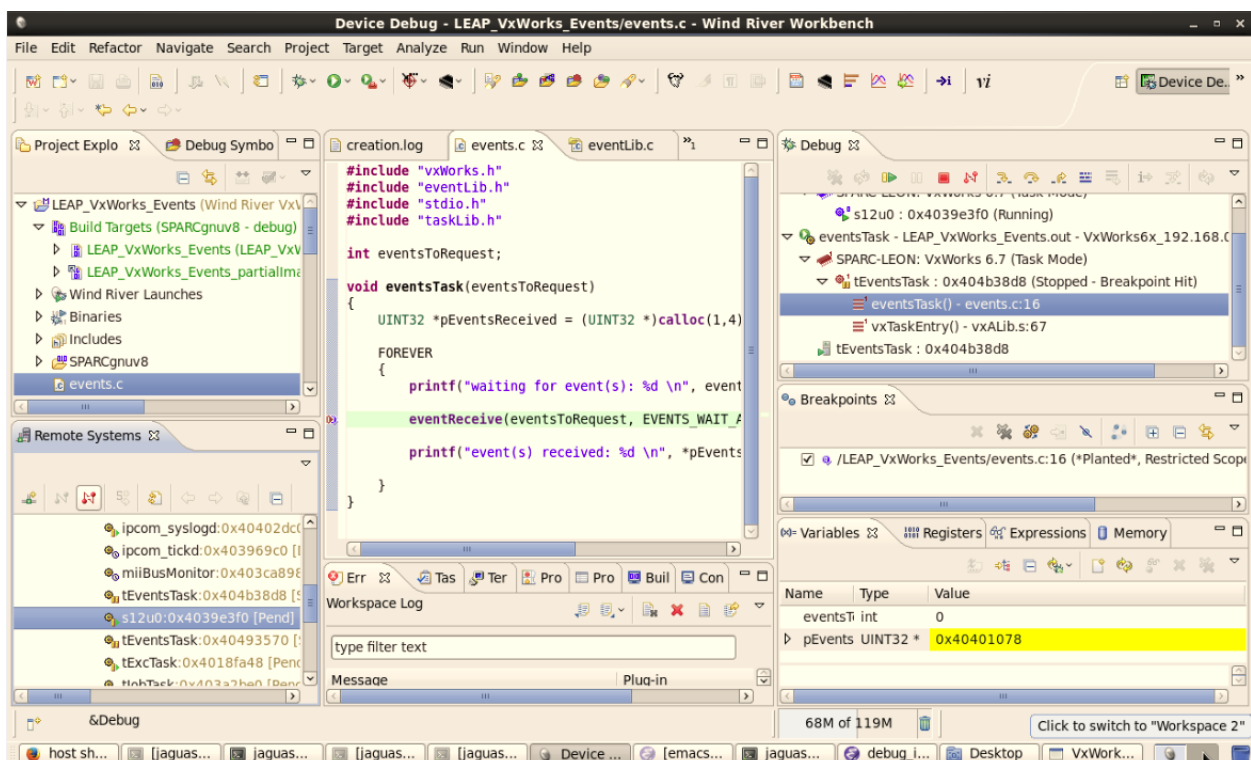



Figure 5.29: Workbench Debug Perspective

To suspend the task, select the task in the Debug view and click the Suspend button 

To resume execution, click Resume 

To set a breakpoint, first suspend as above, and double-click in the blue gutter to the left of the line.

After clicking the **Resume** button, the breakpoint is encountered immediately.

Single **Step Over** a few lines.

5.7 Linux

Linux is a full source package, containing kernel, libraries and application code for rapid development of embedded Linux systems. The LEON port of Linux on the LEAP systems supports the MMU configuration, V8 mul/div instructions and the floating-point unit (FPU). Below is a list of supported hardware:

- LEON3 with MMU, FPU, MUL/DIV
- GPTIMER System Clock Timer
- IRQMP interrupt controller
- APBUART system console
- GRSPW SpaceWire
- GRETH 10/100 and 10/100/1000 Network driver using the MDIO layer
- GRPCI Host support
- OCCAN implements the Linux socket CAN 2.0b interface
- GRGPIO supports the generic General Purpose I/O model of Linux
- SPICTRL supports SPI master interface through the spi-fsl driver

5.7.1 Linux Embedded System

CAES provides the LINUXBUILD utility as a quick way of getting started with Linux development for the LEON architecture. It ties different components together to build a complete Linux environment. Settings for standard LEON Linux configurations are available within the LINUXBUILD package and custom configurations can also be created by the user.

The LINUXBUILD utility documentation is available at www.gaisler.com/doc/linuxbuild.pdf

Table 5.3 Project directory layout

Directory	Content
Bootloader	Mkprom2
images	The binary images of the bootloader, kernel and root filesystem
Kernel	The operating system (OS) running on the target
Rootfs	The root filesystem as seen by the target's kernel at runtime
Tools	The complete cross-platform development toolchain

5.7.2 Booting Linux with mkprom2

Once the Linux image, **image.ram**, is created from LINUXBUILD, mkprom2 can create a boot-image to run from PROM on the LEON3 processor.

To generate a boot-prom from **image.ram**, use the following options as in **Figure 5.30**. Change mkprom2's options according to the particular hardware settings.

```
$ mkprom2 -freq 50 -nosram -sdram 32 -memcfg1 0x1803caff -memcfg2 0x89a06005
-memcfg3 0x08184000 -baud 38400 image.ram -o image.prom

LEON2/3/ERC32 MKPROM prom builder for BCC, ECOS, RTEMS and ThreadX v2.0.60
Copyright Gaisler Research 2004-2007, all rights reserved.

creating LEON3 boot prom: image.prom
Searching for compiler to use (sparc-elf, sparc-rtems or sparc-linux):
sh: sparc-elf-gcc: command not found
sh: sparc-rtems-gcc: command not found
sparc-linux-gcc (sparc-linux-ct-leon_multilib_basic-0.0.7) 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.
```

Figure 5.30: Mkprom2 options

To run the **image.ram** on the target's **PROM**, attach grmon2, see **5.3**, to the target board using the serial debug link (DSU), see **Figure 2.6** and **Figure 5.31-Figure 5.33**.

```
$ grmon -eth 10.5.233.215

grmon2> mcfg1 0x1803caff
    mcfg1: 0x1803caff

grmon2> load image.prom
00000000 .text          6.2MB / 6.2MB [=====>] 100%
Total size: 6.16MB (1.52Mbit/s)
Entry point 0x0 Image image.prom loaded
```

Figure 5.31: Loading Linux boot-prom image

```
MkProm2 boot loader v2.0
Copyright Gaisler Research - all rights reserved

system clock : 50.0 MHz
baud rate    : 38343 baud
prom        : 512 K, (2/2) ws (r/w)
sdram       : 32768 M, 1 bank(s), 9-bit column
sdram       : cas: 2, trp: 40 ns, trfc: 80 ns, refresh 7.8 us

decompressing .text to 0x40000000
decompressing .data to 0x400010f0
decompressing .vmlinux to 0x40004000
decompressing .startup_prom to 0x40846880

starting image.ram

PROMLIB: Sun Boot Prom Version 0 Revision 0
Linux version 3.10.58 (gcc version 4.4.2 (sparc-6
bootconsole [earlyprom0] enabled
ARCH: LEON
TYPE: Leon3 System-on-a-Chip
Ethernet address: 00:00:7c:cc:01:45
CACHE: 4-way associative cache, set size 4k
OF stdout device is: /a::a
PROM: Built device tree with 25547 bytes of memory.
Booting Linux...
PERCPU: Embedded 7 pages/cpu @f0af3000 s6272 r8192 d14208 u32768
```

Figure 5.32: Booting from mkprom2

```
Mounting /tmp /sys and /proc filesystems

Mounting /dev/pts
mkdir: can't create directory '/dev/pts': File exists
mkdir: can't create directory '/dev/shm': File exists
Starting ifplugd...
Starting telnetd...

Starting shell

leon login: root
login[88]: root login on 'ttyS0'

BusyBox v1.23.1 (2015-08-25 15:24:32 MDT) built-in shell (ash)
Enter 'help' for a list of built-in commands.

~ #
```

Figure 5.33: Linux Login

Chapter 6: Benchmarks

CAES provides a Benchmark Performance White Paper of the following benchmarks:

- Dhrystones Benchmark
- Coremark Benchmark
- Flops20 Benchmark
- Stanford Benchmark
- Whetstone Benchmark

The benchmark plots and results are shown in the white paper that can be found at:

www.cobhamaes.com/products/radiation-hardened-solutions-high-reliability-components/computing/microprocessors

REVISION HISTORY

REV	Revision Date	Description of Change	Page(s)	Author
1.0.0	12-04-2014	Initial Release		--
1.0.1	12-20-2016	New Format		Sim, Jose
1.0.2	06-07-2017	1.1,1.2 Update Company Name, 1.4 correct title, 2.10 spacing, Remove version number 5.2 &5.6, 5.4 update list, 5.5 update list, Remove feature list 5.6. Remove Nucleus (No longer supported), Remove Snapgear Linux, only support standard Linux, 5.7 remove text and update list, 5.7.1 update text, chapter 6, Dhrystones benchmark	4,24,28,39 ,47,50,51	Sim
1.0.3	10/27/2017	Add more benchmark results	49-56	MTS
1.0.4	11/28/2017	Remove benchmark results and add reference to Benchmark White Paper	49	MTS

The following United States (U.S.) Department of Commerce statement shall be applicable if these commodities, technology, or software are exported from the U.S.: These commodities, technology, or software were exported from the United States in accordance with the Export Administration Regulations. Diversion contrary to U.S. law is prohibited.

Cobham Colorado Springs Inc. d/b/a Cobham Advanced Electronic Solutions (CAES) reserves the right to make changes to any products and services described herein at any time without notice. Consult an authorized sales representative to verify that the information in this data sheet is current before using this product. The company does not assume any responsibility or liability arising out of the application or use of any product or service described herein, except as expressly agreed to in writing; nor does the purchase, lease, or use of a product or service convey a license under any patent rights, copyrights, trademark rights, or any other of the intellectual rights of the company or of third parties.