
***Tensilica[®] Prototyping User's
Guide for the
Xilinx ML605 (XT-ML605) Board***

For Xtensa[®] Cores

Tensilica, Inc.
3255-6 Scott Blvd.
Santa Clara, CA 95054
(408) 986-8000
fax (408) 986-8919
www.tensilica.com

© 2011 Tensilica, Inc.
Printed in the United States of America
All Rights Reserved

This publication is provided "AS IS." Tensilica, Inc. (hereafter "Tensilica") does not make any warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Information in this document is provided solely to enable system and software developers to use Tensilica processors. Unless specifically set forth herein, there are no express or implied patent, copyright or any other intellectual property rights or licenses granted hereunder to design or fabricate Tensilica integrated circuits or integrated circuits based on the information in this document. Tensilica does not warrant that the contents of this publication, whether individually or as one or more groups, meets your requirements or that the publication is error-free. This publication could include technical inaccuracies or typographical errors. Changes may be made to the information herein, and these changes may be incorporated in new editions of this publication.

Tensilica and Xtensa are registered trademarks of Tensilica, Inc. The following terms are trademarks of Tensilica, Inc.: FLIX, OSKit, Sea of Processors, TurboXim, Vectra, Xenergy, Xplorer, and XPRES. All other trademarks and registered trademarks are the property of their respective companies.

Issue Date: 10/2011
RD-2011.3
PD-11-4524-10-00

Tensilica, Inc.
3255-6 Scott Blvd.
Santa Clara, CA 95054
(408) 986-8000
fax (408) 986-8919
www.tensilica.com

Contents

1. Overview	1
1.1 Required Tools	2
1.2 Features	2
2. Hardware Description	5
2.1 Operational Overview	5
2.2 XT-ML605 Board Diagram	6
2.3 Clock Frequencies	6
2.4 Reset Button	6
2.5 DDR3 SDRAM	7
2.6 Flash Memory	7
2.7 Boot Select Switch	7
2.8 UART	8
2.9 Ethernet	8
2.10 USB Interface	8
2.11 OCD and JTAG	8
2.12 Stereo Audio DAC/ADC	9
2.13 HDMI Interface	9
2.14 Clock synthesizer	9
3. Getting Started	11
3.1 Loading the XT-ML605 Board with a Tensilica Processor	11
3.1.1 Required Tools	12
3.1.2 Programming the Configuration onto the XT-ML605 Board	12
3.1.3 Core Frequency	15
3.2 Connecting the OnCE JTAG Probe to the XT-ML605-D Daughter Board	15
3.3 Installing the Xtensa OCD Daemon	15
3.4 Running a Program Using OCD and GDB on the Command Line	16
4. XT-ML605 Board Programmer's Guide	19
4.1 Memory Map	19
4.2 System I/O Block	20
4.2.1 Flash	21
4.2.2 UART	21
4.2.3 DIP Switches	22
4.2.4 Ethernet	22
4.2.5 LCD Display	22
4.2.6 Board Specific FPGA Registers	22
4.2.7 USB	23
4.3 Audio Specific Interfaces	23
4.3.1 Audio I2S Out	23
4.3.2 I2S Transmitter Config RATIO Bits	25
4.3.3 Audio I2S In	25
4.3.4 I2C Master	26
4.3.5 SPI Interface	27
4.4 Booting from RAM or Flash	27

- 4.5 Developing Software for the XT-ML605 Board..... 27
 - 4.5.1 Using the Xtensa Board-Independent API (`xtbsp.h`)..... 28
 - 4.5.2 Using the XT-ML605 Board-Specific Header (`xtml605.h`) 28
 - 4.5.3 Linking Code for the XT-ML605 Board 29
 - Example 1 30
 - Example 2 30
- 4.6 Debugging Code on the XT-ML605 Board 31
- 4.7 Programming the Flash 31
- 4.8 Using the UART..... 33
- 5. XT-ML605-D Daughter Board 35**
 - 5.1 Feature Set 35
 - 5.2 Hardware Description 35
 - 5.2.1 Headphone (HP)..... 37
 - 5.2.2 Microphone (MIC) 37
 - 5.2.3 Line-In (LI) 37
 - 5.2.4 Line-Out (LO) 37
 - 5.2.5 HDMI Port..... 38
 - 5.2.6 USB Interface..... 38
 - 5.2.7 Flash Memory 38
 - 5.2.8 FMC Connectors 38
 - 5.2.9 OnCE JTAG Probe Header 39
 - 5.2.10 Stereo Audio Codec (AIC)..... 40
 - 5.2.11 HDMI Transmitter..... 40
 - 5.2.12 Clock Synthesizer 41
 - 5.3 Jumper Settings..... 41
 - 5.4 Running a Sample Audio Playout Program 42
 - 5.5 Running a Sample Audio Recording Program 43
 - 5.6 Writing Device Drivers for AIC and HDMI..... 45
 - 5.6.1 Sample C Functions..... 45
 - 5.6.2 Writing Data to and From Memory Through OCD 46

List of Figures

Figure 1–1.	XT-ML605 Emulation Board	3
Figure 2–2.	XT-ML605 Board Diagram	6
Figure 3–3.	Programming Jumpers and Header on the XT-ML605 Board	12
Figure 3–4.	Uploading the Bit-Stream Through Boundary Scan	13
Figure 3–5.	Platform Flash Selection in iMPACT	14
Figure 3–6.	iMPACT After Selecting Platform Flash.....	14
Figure 5–7.	Top View of the XT-ML605-D Daughter Board.....	36
Figure 5–8.	Bottom View of the XT-ML605-D Daughter Board.....	37
Figure 5–9.	Connection of XT-ML605-D Daughter Board and XT-ML605 Board	39
Figure 5–10.	JTAG Probe Header J19.....	39

List of Tables

Table 4–1.	Memory Map of Diamond Cores (Without MMU)	19
Table 4–2.	Default Virtual Memory Map of Diamond DC_232L Core (With MMU).....	20
Table 4–3.	System I/O Block Layout	21
Table 4–4.	Board Specific FPGA Register Layout	23
Table 4–5.	I2S Output Register Mapping	24
Table 4–6.	I2S Input Register Mapping.....	26
Table 4–7.	SPI Register Mapping	27
Table 5–8.	J19 JTAG Probe Header Pin Number and Function	40

Preface

This document is written for Tensilica customers who are experienced in the programming and debugging of software.

Notation

- *italic_name* indicates a program or file name, document title, or term being defined.
- \$ represents your shell prompt, in user-session examples.
- **literal_input** indicates literal command-line input.
- *variable* indicates a user parameter.
- `literal_keyword` (in text paragraphs) indicates a literal command keyword.
- `literal_output` indicates literal program output.
- `... output ...` indicates unspecified program output.
- `[optional-variable]` indicates an optional parameter.
- `[variable]` indicates a parameter within literal square-braces.
- `{variable}` indicates a parameter within literal curly-braces.
- `(variable)` indicates a parameter within literal parentheses.
- | means *OR*.
- `(var1 | var2)` indicates a required choice between one of multiple parameters.
- `[var1 | var2]` indicates an optional choice between one of multiple parameters.
- `var1 [, varn]*` indicates a list of 1 or more parameters (0 or more repetitions).
- `4'b0010` is a 4-bit value specified in binary.
- `12'o7016` is a 12-bit value specified in octal.
- `10'd4839` is a 10-bit value specified in decimal.
- `32'hff2a` or `32'HFF2A` is a 32-bit value specified in hexadecimal.

Terms

- *Ox* at the beginning of a value indicates a hexadecimal value.
- *b* means bit.
- *B* means byte.
- *flush* is deprecated due to potential ambiguity (it may mean *write-back* or *discard*).
- *Mb* means megabit.
- *MB* means megabyte.
- *PC* means program counter.
- *word* means 4 bytes.

Related Tensilica Documents

Customizable DPU Data Books

- *Xtensa® LX4 Microprocessor Data Book*
- *Xtensa® 9 Microprocessor Data Book*
- *Xtensa® TX Dataplane Processor Data Book*
- *Xtensa® Instruction Set Architecture (ISA) Reference Manual*

DPU User Guides and Reference Manuals

ConnX Communications DPUs

- *ConnX D2 DSP Engine User's Guide*
- *ConnX Vectra™ LX DSP Engine Guide*
- *ConnX BBE16 DSP Engine User's Guide*

HiFi Audio DSPs

- *HiFi 2 Audio Engine Codecs Programmer's Guides*
- *HiFi 2/EP Audio Engine Instruction Set Architecture Reference Manual*

Hardware Design Guides

SOC Implementation

- *Xtensa Hardware User's Guide*
- *Xtensa® System Designer's Guide*
- *Tensilica Bus Designer's Toolkit Guide*
- *Tensilica Processors Bus Bridges Guide*
- *Xtensa® Processor Interface Protocol Reference Manual*
- *Tensilica Avnet LX200 (XT-AV200) Board User's Guide*
- *Tensilica Avnet LX110 (XT-AV110) Board User's Guide*
- *Tensilica Avnet LX600 (XT-AV60) Board User's Guide*

TIE Creation and Processor Optimization

- *Tensilica Instruction Extension (TIE) Language Reference Manual*
- *Tensilica Instruction Extension (TIE) Language User's Guide*
- *Xtensa® Processor Extensions Synthesis (XPRES™) Compiler User's Guide*
- *Xtensa® Upgrade Guide*

Software Development and Simulation Tools Guides

- *Xtensa® Software Development Toolkit User's Guide*
- *Xtensa® Development Tools Installation Guide*

- *Tensilica C Application Programmer's Guide*
- *Xtensa® C and C++ Compiler User's Guide*
- *Xtensa® Energy Estimator (Xenergy) User's Guide*
- *Xtensa® Linker Support Packages (LSPs) Reference Manual*
- *Xtensa® OSKit™ Guide*
- *Xtensa® Microprocessor Programmer's Guide*
- *Xtensa® System Software Reference Manual*

System Simulation

- *Xtensa® Instruction Set Simulator (ISS) User's Guide*
- *Xtensa® Modeling Protocol (XTMP) User's Guide*
- *Xtensa® SystemC® (XTSC) Reference Manual*
- *Xtensa® SystemC® (XTSC) User's Guide*
- *Xtensa® SystemC® (XTSC) for Carbon SoC Designer User's Guide*

Utilities and Libraries

- *GNU Assembler User's Guide*
- *GNU Binary Utilities User's Guide*
- *GNU Debugger User's Guide*
- *GNU Linker User's Guide*
- *GNU Profiler User's Guide*
- *Red Hat newlib C Library Reference Manual*
- *Red Hat newlib C Math Library Reference Manual*
- *Xtensa C Library User's Guide*

Debug and Trace Product Guides

- *Tensilica On-Chip Debugging Guide*
- *Tensilica Trace Solutions User's Guide*

Diamond Standard Controllers and Standard DSPs Guides

- *Diamond Series Hardware User's Guide*
- *Diamond Series Upgrade Guide*
- *Diamond Standard Controllers Data Book*
- *330HiFi Standard DSP Processor Data Book*
- *545CK Standard DSP Processor Data Book*

Other Related Documents

The XT-ML605 board contains integrated circuits from a variety of manufacturers. Tensilica does not provide copies of documentation for those devices. Advanced users of the XT-ML605 board may want to obtain data sheets or user's guides from the device manufacturer to support complex usage of the XT-ML605 board. Here is a list of references that may be useful.

- *Xilinx® Virtex®-6 LX Evaluation Kit User Guide*, can be obtained from the Xilinx support website.
- Ethernet PHY data sheet - <http://www.national.com/ds/PC/PC16550D.pdf>
- LCD display data sheet - <http://www.mytechcorp.com/cfdata/productFile/File1/MOC-16216B-B-A0A04.pdf>
- Cypress CY7C67300 USB controller data sheet - <http://www.cypress.com/?rID=14164>
- Stereo audio codec data sheet - <http://focus.ti.com/lit/ds/symlink/tlv320aic23.pdf>
- HDMI transmitter data sheet - http://www.analog.com/static/imported-files/data_sheets/AD9889B.pdf
- Clock synthesizer data sheet - <http://focus.ti.com/lit/ds/symlink/tlv320aic23.pdf>
- www.opencores.org
- www.ecoscentric.com/ecos/redboot.shtml

1. Overview

The Xilinx ML605 Virtex-6 evaluation board can be easily configured with a Tensilica processor system to offer a complete emulation, evaluation, and development platform. A designer can evaluate each Tensilica processor configuration by developing and debugging software early in the design cycle on a dedicated FPGA platform.

In this guide, *XT-ML605 board* refers to the Xilinx ML605 evaluation board configured with a Tensilica core using FPGA configuration files provided by Tensilica.

You can find purchasing information about the Xilinx ML605 board from the Xilinx website at:

<http://www.xilinx.com/products/boards-and-kits/EK-V6-ML605-G.htm>

The Xilinx ML605 board can be bought from Avnet and other distributors. The Avnet part number is EK-V6-ML605-G-J. When purchasing the board, make sure that the board contains a Xilinx Virtex-6 LX240T device. You also need the XT-ML605-D daughter board; contact Tensilica support to obtain the board. Note that Tensilica may have only a limited stock of XT-ML605-D daughter boards.

The XT-ML605 board uses a Xilinx Field Programmable Gate Array (FPGA) for hardware emulation of a Tensilica processor core. Tensilica has completed the synthesis, place, and route of the core, along with appropriate peripheral controllers. Load the FPGA configuration files onto the processor FPGA from the host using the USB JTAG port on the Xilinx ML605 board and Xilinx programming software.

Tensilica provides bitstreams for the FPGA and software support consisting of a limited set of drivers, linker support packages, and examples that demonstrate how to build and run software on the board. For Diamond processors, this is provided pre-built for a specific core. For Xtensa processors, this is built and downloaded on request via the Xtensa Processor Generator (XPG).

Four I/O ports are available for board communication and control:

1. USB to UART serial port

The USB to UART serial port provides a communications link to a Linux or Windows workstation host for debugging software.

2. On-Chip Debugger (OCD) module (through XT-ML605-D daughter board)

This module provides full and precise debugger control of all Tensilica cores through a JTAG probe and the Xtensa OCD Daemon.

3. 10/100 Ethernet port.

4. Peripheral USB port (through XT-ML605-D daughter board)

This guide covers the XT-ML605 board in five chapters:

- Chapter 1 introduces the XT-ML605 board and discusses the capabilities and required tools.
- Chapter 2 describes the hardware features used on the XT-ML605 board.
- Chapter 3 describes how to install software, configure and run code on the XT-ML605 board.
- Chapter 4 provides a programmer's guide for using the various peripherals on the XT-ML605 board.
- Chapter 5 describes the features of XT-ML605-D daughter board.

1.1 Required Tools

When the board is purchased from Xilinx or Avnet, it is not preloaded with any Tensilica processors. While Tensilica provides bitstreams targeted for the XT-ML605 board, loading the configuration on the XT-ML605 board requires the following tools to program the FPGA configuration ROMs:

- Xilinx iMPACT Programming Software version 13.1 or greater

To debug the processor through OCD, a JTAG wiggler must be purchased from a third party, and a XT-ML605-D daughter board must be obtained from Tensilica. See Section 3.2 “Connecting the OnCE JTAG Probe to the XT-ML605-D Daughter Board” and Section 3.3 “Installing the Xtensa OCD Daemon” for more information.

1.2 Features

The XT-ML605 emulation system provides the following:

- Xilinx FPGA, to which you can download a Tensilica processor
- 512 MB DDR3 SDRAM
- 16MB Flash
- 10/100 (10/100 Mbps) Ethernet
- USB to UART serial port
- 32-character, 2-line, alphanumeric LCD display
- Debug support via JTAG and the Xtensa OCD Daemon, through the XT-ML605-D daughter board
- Audio ADC and DAC, stereo headphone output, line output and stereo microphone input through the XT-ML605-D daughter board
- High Definition Multimedia Interface (HDMI) output through the XT-ML605-D daughter board.

- Peripheral USB port through the XT-ML605-D daughter board.

Figure 1–1 shows a logical block diagram of the XT-ML605 emulation system implemented on the FPGA.

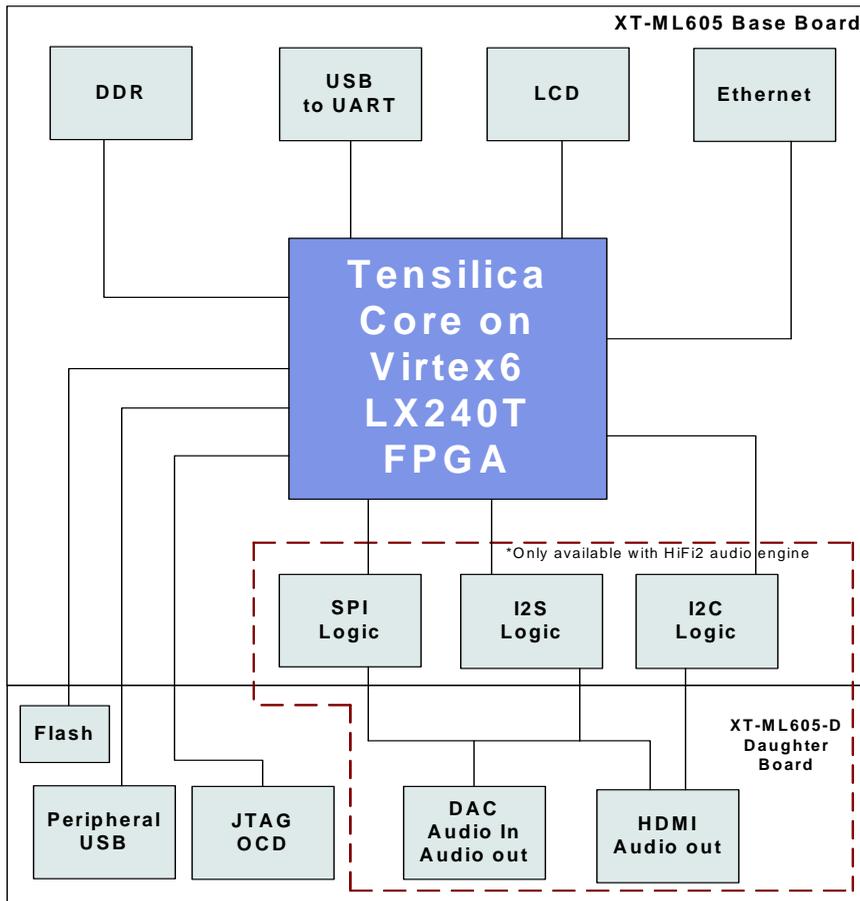


Figure 1–1. XT-ML605 Emulation Board

2. Hardware Description

The XT-ML605 board is highly programmable and has a variety of memory and input and output (I/O) devices. The hardware features on the board make it suitable for many tasks. This guide outlines all of the hardware devices used by Tensilica processors. For a comprehensive description of the board, refer to the Xilinx ML605 documentation from the Xilinx website at

http://www.xilinx.com/products/boards/ml605/reference_designs.htm

2.1 Operational Overview

The Tensilica processor runs on the Xilinx Virtex-6 XC6VLX240T-1-FF1156 FPGA on the board. Using the processor interface (PIF), the processor can access the following peripherals:

- 512MB DDR3 SDRAM
- 16MB Flash
- UART
- 10/100 Ethernet MAC
- 32-character LCD display
- USB peripheral interface
- Stereo Audio DAC on the XT-ML605-D daughter board (only for the configurations that have HiFi 2, HiFi EP, or Connx D2 configured)
- Stereo Audio ADC on the XT-ML605-D daughter board (only for the configurations that have HiFi 2, HiFi EP, or Connx D2 configured)
- HDMI transmitter on the XT-ML605-D daughter board (only for the configurations that have HiFi 2, HiFi EP, or Connx D2 configured)
- Clock Synthesizer on the XT-ML605-D daughter board (only for the configurations that have HiFi 2, HiFi EP, or Connx D2 configured)

Conversely, the processor can also be accessed using its debugger interfaces:

- On-Chip Debug (OCD) module via JTAG port on the XT-ML605-D daughter board.

2.2 XT-ML605 Board Diagram

Figure 2–2 shows the components of the XT-ML605 board.

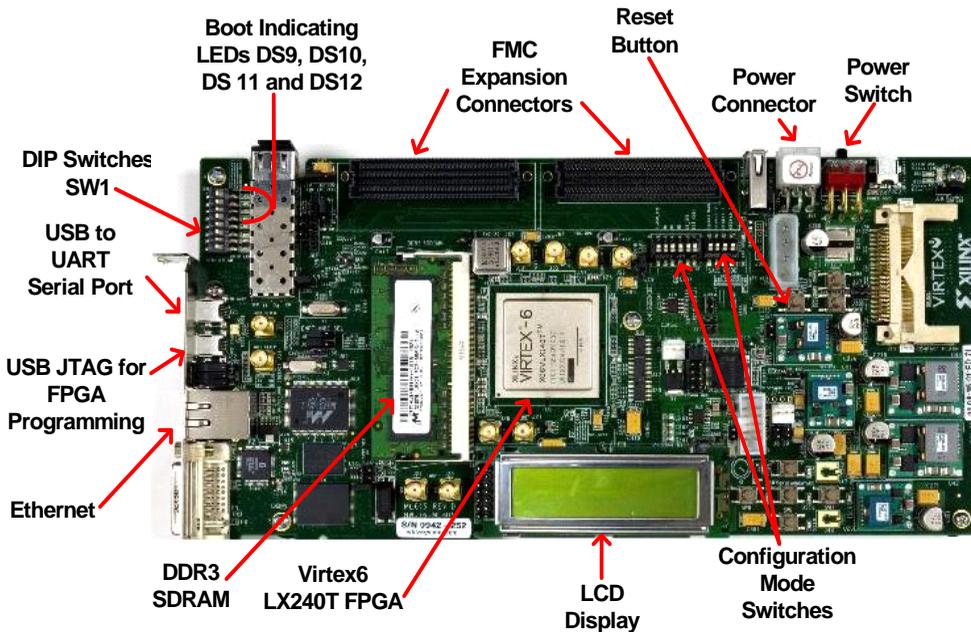


Figure 2–2. XT-ML605 Board Diagram

2.3 Clock Frequencies

The XT-ML605 board uses one 200 MHz differential clock source. This clock is divided using the Mixed Mode Clock Managers (MMCMs) on the FPGA to provide clocks to the processor and DDR3 SDRAM. The frequency at which the Tensilica processor can run is dependent on the configuration. See Section 3.1.3 “Core Frequency” on page 15 for details.

The DDR3 SDRAM always runs at 333MHz, asynchronous to the Tensilica processor.

2.4 Reset Button

The XT-ML605 board uses one main reset button, SW10, to reset all of the devices within the FPGA. When pressed, this button resets the core and all the peripherals.

2.5 DDR3 SDRAM

After an Xtensa processor bitstream is programmed onto the board, the XT-ML605 board uses 512MB of DDR3 SDRAM. The DDR3 SDRAM is always memory-mapped to the configuration's system RAM and system ROM locations (see Section 4.1 "Memory Map" on page 19). 256MB is allocated to system ROM, and 256MB is allocated to system RAM.

2.6 Flash Memory

The XT-ML605-D daughter board contains 16MB of 16-bit wide flash memory. You can boot the processor using this flash device by mapping it to the ROM area using the boot select switch described in Section 2.7 "Boot Select Switch". The flash is also mapped in the system I/O block as described in Section 4.2 on page 20.

Note: The Tensilica XT-ML605 support package provides a simple example program, flasher, to burn an image into the flash.

2.7 Boot Select Switch

The system ROM area (see Section 4.1 "Memory Map" on page 19) maps to either the SDRAM or the flash device. The XT-ML605 board's SW1 DIP switch number 8 selects between the two. Because the ROM area includes the processor's reset vector, this switch effectively selects whether the core boots from the SDRAM or the flash device. LEDs DS9, DS10, DS11 and DS12 on the XT-ML605 board indicate the boot selection:

- If ON, LEDs DS9, DS 10, DS11 and DS12 are lit and the processor boots from the flash.
- If OFF, LEDs DS9, DS10,DS11 and DS12 are not lit and the processor boots from the SDRAM.

Note that if you have enabled Relocatable Vectors and have selected the **Alternate Static Vector Base Address** option, you will not be able to boot from the Flash. With the alternate static vector base address, only booting from SDRAM is possible and SW1 DIP switch number 8 should be OFF.

2.8 UART

The UART is implemented as a standard 16550 UART, using the external interrupt 0 input of the processor. This maps to the processor's lowest numbered external interrupt (on Diamond processors this is interrupt number zero). The UART is clocked at the core's clock frequency (it is not derived from a separate crystal). The RTS and CTS signals are not implemented.

The UART is also mapped in the system I/O block as described in Section 4.2.2 "UART".

Note: The UART uses the USB to UART port J21 on the base board. You must install CP210x VCP drivers on the host PC first in order to use UART. For more details on these drivers refer to the Xilinx documentation, specifically look for user guide titled "Getting Started with the Xilinx Virtex-6 FPGA ML605 Evaluation Kit".

2.9 Ethernet

The Ethernet MAC is memory-mapped and uses the external interrupt 1 input of the processor. This maps to the processor's second lowest numbered external interrupt (on Diamond processors this is interrupt number one).

2.10 USB Interface

The XT-ML605-D daughter board provides peripheral USB interface. The USB interface is memory mapped and uses external interrupt 5 of the processor. This maps to the processor's fifth lowest numbered external interrupt (on Diamond processors this is interrupt number 5). Note that the host USB port on the XT-ML605 (base board) is not supported. The XT-ML605-D daughter board provides only the peripheral USB interface. There is an additional host USB port on the XT-ML605-D daughter board, but it is currently not supported.

2.11 OCD and JTAG

The On-Chip Debugger (OCD) module, based on the IEEE 1149.1 specification, provides access to the processor core through a JTAG Test Access Port (TAP) controller. The JTAG TAP controller uses the standard 5-signal JTAG port. The OCD module and JTAG are implemented through the XT-ML605-D daughter board. See Chapter 5, "XT-ML605-D Daughter Board" on page 35 for details on the daughter board, OCD module and JTAG. For additional information about OCD, refer to the *Tensilica On-Chip Debugging Guide*.

2.12 Stereo Audio DAC/ADC

Analog audio functionality is implemented via a TI 320AIC23 stereo audio codec on the XT-ML605-D daughter board. Audio playout through headphones and line out, and input through the microphone and line-in jacks, are supported through the daughter board; however, these hardware features can be used only when HiFi 2, HiFi EP, or ConnX D2 is configured in the processor using XPG instruction options. In that case, the audio interface uses external interrupts 2 and 3 of the processor for the I²S transmitter FIFO and I²S receiver FIFO interrupts, respectively. See Chapter 5, “Stereo Audio Codec (AIC)” on page 40, for details on the daughter board and the audio DAC.

2.13 HDMI Interface

The XT-ML605-D daughter board provides HDMI transmitter interface using Analog Devices AD9889B for playout of multichannel digital audio. This hardware feature is available only when HiFi 2, HiFi EP, or ConnX D2 is configured. It uses external interrupt pin 2 for I²S transmitter FIFO interrupts. See Chapter 5, “HDMI Transmitter” on page 40, for details on the daughter board and the HDMI interface.

2.14 Clock synthesizer

The XT-ML605-D daughter board provides a TI CDCE706 clock synthesizer device. It is used to generate the bus clock for I²S transmitter and I²S receiver interfaces. The use of the clock synthesizer makes it possible to support multiple audio sampling frequencies. For details, refer to Chapter 5, “Clock Synthesizer” on page 41.

3. Getting Started

The XT-ML605 board is shipped from Xilinx/Avnet without any Tensilica processors on the FPGA. Debugging via JTAG requires the XT-ML605-D daughter board from Tensilica. This chapter describes the hardware tools and power up requirements for the XT-ML605 board, how to connect the OnCE JTAG probe, and how to build and run software programs on the board. Specifically, it describes these steps:

- Loading a Tensilica configurable processor on the XT-ML605 board FPGA
- Connecting the OnCE JTAG probe to the XT-ML605-D daughter board
- Running a pre-built example program on the board using command-line tools

This document assumes that the Tensilica software tools, core packages and the OCD Daemon have already been installed and the JTAG probe has been set up per the instructions provided with the tools (see the *Xtensa Development Tools Installation Guide*). It also assumes that a processor core configuration has been generated on XPG for the targeted XT-ML605 board. This document refers to the installed locations of the software tools and core specific packages as follows:

`<xtensa_tools_root>` Root of Xtensa Tools installation

`<xtensa_root>` Root of core-specific package installation

You can obtain these paths by typing the following commands at a command shell prompt:

```
<xtensa_tools_root>:
```

```
$ xt-run --show-config=xttools
```

```
<xtensa_root>:
```

```
$ xt-run --show-config=config
```

3.1 Loading the XT-ML605 Board with a Tensilica Processor

The XT-ML605 board allows different Tensilica processors to be loaded onto the board. This section describes the steps required to load a new processor configuration onto the XT-ML605 board.

Tensilica provides the necessary set of FPGA configuration PROM files (bitstreams). For Diamond processors, this is provided pre-built. For Xtensa processors, this is built and downloaded on request via the Xtensa Processor Generator (XPG).

3.1.1 Required Tools

- iMPACT FPGA Programming Software (Xilinx WebPACK) version 13.1 or greater
- USB Type-A to mini-B 5-pin cable (Included in the Xilinx ML605 evaluation kit)

3.1.2 Programming the Configuration onto the XT-ML605 Board

The FPGA configuration data is stored in Xilinx Platform Flash XL device XCF128X-FTG64C on the XT-ML605 board. During power up, the configuration flash device is used to configure the Virtex-6 FPGA on the board. Before continuing, ensure that you are using Xilinx ISE, Version 13.1 or greater, or the following steps may not work.

Figure 3–3 illustrates the placement of different programming jumpers and headers. Make sure that the placement on your board is the same as this before you program it.

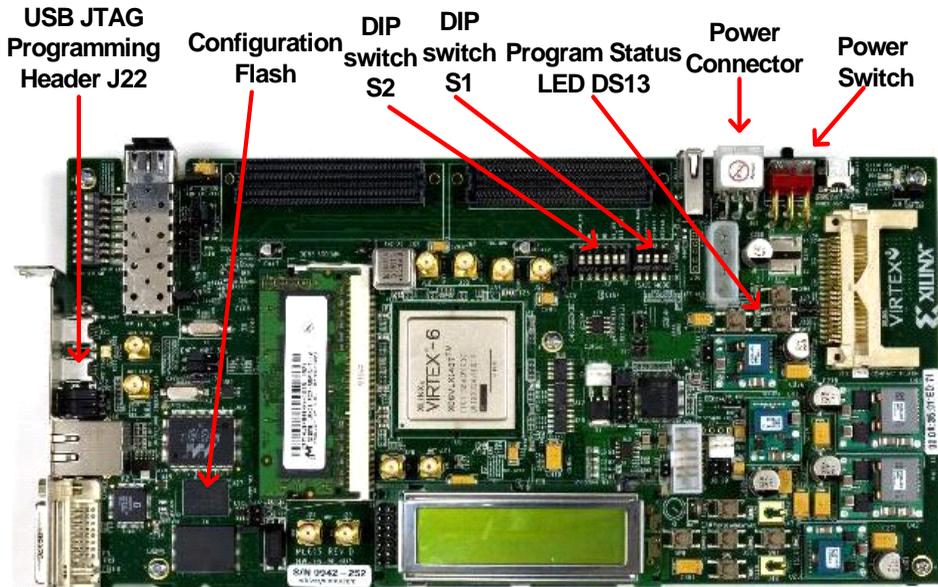


Figure 3–3. Programming Jumpers and Header on the XT-ML605 Board

Program the Platform Flash with the configuration files¹ as follows:

1. Set the FPGA configuration mode to SlaveSelectMap, to do this follow the steps below.

1. Located in the FPGA package supplied by Tensilica or downloaded from the Xtensa Processor Generator.

2. DIP switch S2, switch number 5 and 4 should be ON, switch number 3 should be OFF.
3. DIP switch S2, switch number 6 should be OFF.
4. DIP switch S2, switch number 1 should be ON and switch number 2 should be OFF. This ensures that FPGA configuration uses Platform Flash XL.
5. DIP switch S1 switch number 4 should be OFF preventing configuration from compact flash card.
6. Ensure that the USB Type-A to mini-B cable connects the host PC to USB JTAG port J22 on the XT-ML605 board.
7. Power up the XT-ML605 board.
8. Start the Xilinx iMPACT Configuration tool.
9. When the message asking if you want to load the most recent project, or create a new project displays, click **Create a new project** to proceed to the next step. From the dialog box, select **Configure devices using Boundary-Scan (JTAG)**. The dialog box opens to assign new configuration files. Click **Cancel**.
10. As shown in Figure 3–4, the right section of the window shows the boundary scan, which consists of the Xilinx FPGA, xc6vlx240T device. Right click on the FPGA and select “Add SPI/BPI flash” option. When a window opens up, browse to the `prom0.mcs` file for your specific Tensilica processor.

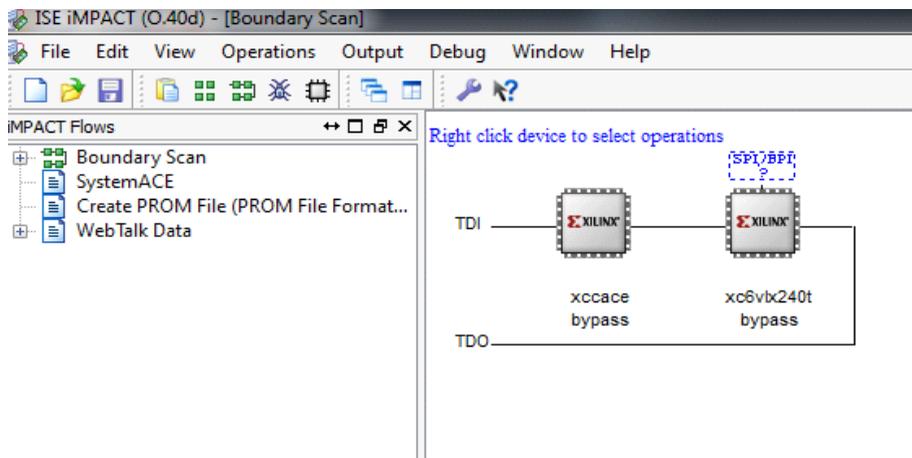


Figure 3–4. Uploading the Bit-Stream Through Boundary Scan

- 11. Another window titled “Select attached SPI/BPI” pops up. Make sure you set the drop down options in this window as shown in Figure 3-5 below and press OK.

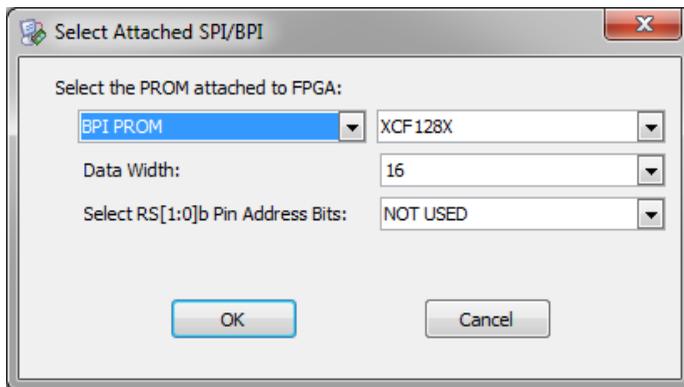


Figure 3-5. Platform Flash Selection in iMPACT

- 12. As shown in Figure 3-6, the FLASH icon should appear on top of the FPGA icon in the right side of the window.

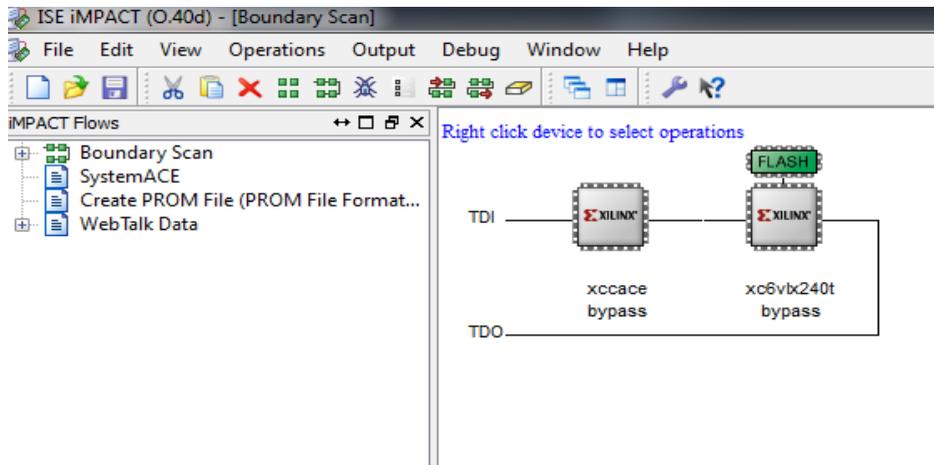


Figure 3-6. iMPACT After Selecting Platform Flash

- 13. Right click on the FLASH icon and select “Program”. Click "OK" to start programming.
- 14. Now iMPACT should start programming. The programming will take a few minutes. When the program is done successfully, a Program Succeeded message displays at the end of the programming. If a failure message displays, verify that the appropriate programming options were selected.

- Turn off the XT-ML605 board's power; wait for few moments and then turn the power on again. The FPGA will be programmed from the flash device upon power-up. If the FPGA is programmed successfully, FPGA Done LED DS13 will turn on.

3.1.3 Core Frequency

Some processors may run faster or slower depending on the configuration and size of the processor. The configuration's timing information is located along with the FPGA configuration (bitstream) files in the `Xm_<core>FPGAMain.par` file. For Xtensa customers, the `Xm_<core>FPGAMain.par` file is located under the `<core>FPGAFILES` directory as part of your bitstream download. Diamond customers can obtain the bitstream from customer support. The constraints report inside this file indicates if the design met desired timing. Search for the phrase "`TS_divclk_dcm`" inside this file, the entry in the Constraint report table looks something like this:

Constraint	Check	Worst Case Slack	Best Case Achievable

<code>TS_divclk_dcm = PERIOD TIMEGRP</code>	<code>SETUP</code>	<code>0.357ns</code>	<code>19.643ns</code>
<code>"divclk_dcm" TS_CLKP / 0.16 67 HIGH 50%</code>	<code>HOLD</code>	<code>0.021ns</code>	

Negative Worst Case Slack indicates that the design did not meet timing, positive slack as shown in the sample above, indicates the design met timing. The case above targets and meets 50MHz.

Note: The speed of the DDR3 SDRAM is always fixed at 333MHz.

3.2 Connecting the OnCE JTAG Probe to the XT-ML605-D Daughter Board

A daughter card, XT-ML605-D, is required from Tensilica to support on-chip debugging using the JTAG probe. Please refer to Chapter 5, "XT-ML605-D Daughter Board" on page 35 for details. Contact Tensilica Support for the availability of the XT-ML605-D daughter board.

3.3 Installing the Xtensa OCD Daemon

The Xtensa OCD Daemon (XOCD) auto-installer is downloaded from Xplorer's XPG view. For details about installing the Xtensa OCD Daemon, refer to the *Xtensa Development Tools Installation Guide*. For details about setting up the installed Xtensa OCD Daemon, refer to the *Tensilica On-Chip Debugging Guide*. Refer to the *GNU Debugger User's Guide* (for GDB) or the Xplorer online help for instructions to establish a connection to XOCD from `xt-gdb` or Xplorer, respectively.

The following duplicates a small subset of the installation instructions found in the *Tensilica On-Chip Debugging Guide*, adding more specific topology file entries. For more details, including additional driver installation steps required for parallel port probes, refer to the *Tensilica On-Chip Debugging Guide*.

As of this writing, the Xtensa OCD Daemon supports the following JTAG probes on Windows XP hosts:

- usbWiggler from Macraigor (www.macraigor.com)
- USB port USB2Demon™ from Macraigor (www.macraigor.com)
- Parallel port Wiggler™ from Macraigor (www.macraigor.com)
- Ethernet port Catapult™ from ByteTools (www.byte-tools.com)

After the Xtensa OCD Daemon is installed, edit its `topology.xml` file (located in the Daemon's install directory) according to the JTAG probe used. In most cases, all that is needed is to replace the `<controller .../>` line as follows.

For the Macraigor USB2Demon:

```
<controller id='Controller0' module='macraigor' cable='usb2demon' speed='2' />
```

For the Macraigor Wiggler (see the XOCD README or guide for `ntpar` driver installation):

```
<controller id='Controller0' module='macraigor' cable='wiggler' speed='1' />
```

For the ByteTools Catapult (where `xx.xx.xx.xx` is the Catapult's IP address):

```
<controller id='Controller0' module='catapult' ipaddr='xx.xx.xx.xx' speed='16000000' />
```

On Linux hosts, only the Macraigor USB2Demon™ and usbWiggler™ are supported, and only on RedHat Enterprise 4 (see the *Tensilica On-Chip Debugging Guide*).

3.4 Running a Program Using OCD and GDB on the Command Line

Once all installation and set up steps described earlier in this chapter are completed (this includes loading a Tensilica core on the XT-ML605 board FPGA, installing the Xtensa OCD Daemon and the Xtensa Tools and core package), you are ready to run a simple program on the XT-ML605 board. In this example, the pre-built `count` program is run on the board using your default core configuration.

1. Launch the Xtensa OCD Daemon.
2. Connect the JTAG probe to the XT-ML605-D board and to the OCD host machine (or Ethernet network) and then power up the board.

- To select the boot memory of the processor, set SW1 DIP switch 8 as described in Section 2.7 “Boot Select Switch” on page 7. When the LEDs are lit, the processor boots from flash. When the LEDs are *not* lit, the processor boots from the SDRAM. For this example, the processor boots from the SDRAM, so the LEDs should *not* be lit.
- Connect to the target’s core from the debugger and start the debugger (`xt-gdb`) as you normally would. It is usually simpler to provide the name of the image you want to download and run. For this example, locate the `count` program provided as an example in the Tensilica core package, and start `xt-gdb`. Type:

```
$ xt-gdb <xtensa_root>/xtensa-elf/bin/xtml605/count
```

where `<xtensa_root>` is the root of your Tensilica core specific package as described on page 11.

- To connect to the target, at the GDB prompt execute the following:


```
target remote [<ip_address>]:20000 <target_memory_address>
```

 where `<ip_address>` is the IP address (or DNS name) of the PC running the Xtensa OCD Daemon (this may be omitted if that is localhost, the same machine running `xt-gdb`), 20000 is the default port number, and `<target_memory_address>` is any 16 byte aligned address at which there is RAM. If you have trouble connecting to the target, the Xtensa OCD Daemon window may display an informative error message on the lines that start with asterisks.
- Load the image into RAM using the `load` command. Note that this also sets the program counter `$pc` to the reset vector. This step is not necessary if the image you are debugging is already burned in read-only memory or already loaded in RAM by some other mechanism (in that case, you might still have to set `$pc` to some appropriate value) by using the `set $pc = <start address>`. At the GDB prompt, type:


```
load
```
- Start the execution using the `continue` command (usually abbreviated to `c` at the GDB prompt).

Output is printed continuously on the LCD to demonstrate that the program is running on the board. You should see a continuously changing up count number sequence on the LCD.

For more information about using the debugger and debugging with the on-chip debug mode, refer to the *Tensilica On-Chip Debugging Guide* and *GNU Debugger User’s Guide*.

4. XT-ML605 Board Programmer's Guide

The XT-ML605 board, along with Tensilica's software development tools provide all the resources necessary to perform both deeply embedded software development and more fully functioned software development. The peripherals available on the XT-ML605 board allow development of software systems as complex as full-featured desktop operating systems such as Linux, while the basic CPU emulation function of the board allows development of low-level, CPU-specific code.

4.1 Memory Map

The address space of the processor is mapped onto memories and I/O devices on the XT-ML605 board in a manner that depends on the Tensilica processor configuration in the FPGA. The board memory map for a processor configuration is provided to your program via a board specific header `xtml605.h` located in the core specific tools package. Section 4.5.2 on page 28 describes where it is located and how to use it.

Xtensa processor users determine the memory map when designing a configuration with the Xtensa Processor Generator (XPG) interface in Xplorer. If XT-ML605 board is targeted, the local memory sizes are constrained to fit on the board.

For Diamond processors the memory map is predetermined. Most Diamond cores share the same memory map, except the MMU equipped DC_233L. Table 4–1 shows the XT-ML605 board's memory map for most Diamond cores (except DC_233L). Table 4–2 shows the XT-ML605 board's memory map for the DC_233L core. These memory maps outline the locations of system RAM, system ROM, local memories, as well as the 224 MB system I/O block that maps all peripherals (see Section 4.2)

Table 4–1. Memory Map of Diamond Cores (Without MMU)

Memory Map Contents	Address	Size	Reserved Size	Default Cache Mode
<i>Reserved</i>	0x00000000	512 MB	512 MB	Illegal
<i>Reserved</i>	0x20000000	511.75 MB	511.75 MB	Bypass cache
Data RAM 1 (108Mini, 330HiFi or 545CK) -or- Data Port (212GP, 570T)	0x3FFC0000	128 KB	128 KB	(n/a)
Data RAM 0	0x3FFE0000	128 KB	128 KB	(n/a)
Instruction RAM	0x40000000	128 KB	256 MB	(n/a)
System ROM	0x50000000	256 MB [†]	256 MB	Cached [†]
System RAM	0x60000000	256 MB	256 MB	Cached [†]

Table 4–1. Memory Map of Diamond Cores (Without MMU) (continued)

Memory Map Contents	Address	Size	Reserved Size	Default Cache Mode
System I/O Block (cached [†])	0x70000000	224 MB	256 MB	Cached [†]
<i>Reserved</i>	0x80000000	256 MB	256 MB	Bypass cache
System I/O Block (uncached)	0x90000000	224 MB	256 MB	Bypass cache
<i>Reserved</i>	0xA0000000	1536 MB	1536 MB	Bypass cache

†. As selected by the Boot Select Switch described in Section 2.7 on page 7 the system ROM region will map to either 16MB of flash or DDR3 SDRAM.

‡. On the Diamond 106Micro, 108Mini, and 545Ck Standard cores, accesses to this region are never cached because there are no caches.

Table 4–2. Default Virtual Memory Map of Diamond DC_232L Core (With MMU)

Memory Map Contents	Virtual Address	Physical Address	Size	Reserved Size	Default Cache Mode
<i>Unmapped</i>	0x00000000	n/a	3.25 GB	3.25 GB	Illegal
System RAM (cached)	0xD0000000	0x00000000	128 MB	128 MB	Cached
System RAM (uncached)	0xD8000000	0x00000000	128 MB	128 MB	Bypass cache
System I/O Block (cached)	0xE0000000	0xF0000000	224 MB	224 MB	Cached
System ROM (cached)	0xEE000000	0xFE000000	16 MB [†]	32 MB	Cached
System I/O Block (uncached)	0xF0000000	0xF0000000	224 MB	224 MB	Bypass cache
System ROM (uncached)	0xFE000000	0xFE000000	16 MB	32 MB	Bypass cache

†. As selected by the Boot Select Switch described in Section 2.7 on page 7, the system ROM region will map to either 16MB of flash or DDR3 SDRAM.

4.2 System I/O Block

The system I/O block is a 224 MB-sized, 256 MB-aligned area of physical memory. The base address of the system I/O block in memory is based upon the processor configuration, as shown in Section 4.1. Within the system I/O block, all offsets to particular device registers and memories are constant.

Device driver writers may want to make both cached and uncached references to the system I/O block. Therefore, the system I/O block is mirrored in the address map. With the default attribute settings in the TLBs, one copy of the system I/O block appears in a cached address range and one copy appears in an uncached address range.

Table 4–3 shows the layout of the system I/O block. Offsets must be added to the appropriate (cached or uncached) system I/O block address to form a full 32-bit address. Address ranges not shown are reserved.

Table 4–3. System I/O Block Layout

Address Offset	Current Size	Reserved Size	Device
0x08000000	16 MB	64 MB	Flash
0x0d020000	20 bytes	64 KB	FPGA registers
0x0d030000	16 KB	16 KB	Ethernet registers
0x0d034000	1 KB	1 KB	Ethernet buffer descriptors
0x0d050000	32 bytes	64 KB	UART
0x0d080000	64 bytes	32 KB	Audio I ² S Out
0x0d088000	64 bytes	32 KB	Audio I ² S In
0x0d090000	32 bytes	64 KB	I ² C Master
0x0d0a0000	64 bytes	64 KB	SPI
0x0d0c0000	8 bytes	64 KB	LCD display controller
0x0d0d0000	16 bytes	64 KB	USB controller
0x0d800000	16 KB	64 KB	Ethernet buffer memory

Some of the interfaces in the System I/O block are only available when HiFi 2, HiFi EP, or ConnX D2 is configured. Section 4.3 “Audio Specific Interfaces” on page 23 describes these audio specific interfaces. All other devices and interfaces are described below.

4.2.1 Flash

The flash is implemented with a 16MB, 16-bit wide flash chip. Note that the flash physically resides on the XT-ML605-D daughter board. This memory is mapped into the system I/O block, and is optionally also mapped into the system ROM space as described in Section 2.7 “Boot Select Switch” on page 7, making the flash bootable.

4.2.2 UART

The UART is implemented as a standard 16550 UART. Although the registers in the UART are only 1-byte wide, they are aligned to 4-byte addresses. The registers are described in the PC16550D data sheet, at

<http://www.national.com/ds/PC/PC16550D.pdf>.

Details about how the UART is connected in the XT-ML605 board are described in Section 2.8 “UART” on page 8. The UART uses the interrupt 0 pin.

Note: The access to UART is through USB to UART port J21 only. There is no RS232 connector on the XT-ML605 base board or the XT-ML605-D daughter board. The host PC must have CP210x VCP drivers installed prior to accessing the UART. These drivers can be obtained from the website www.silabs.com, alternatively they are also available on the USB flash drive shipped with the Xilinx ML605 board.

4.2.3 DIP Switches

The status of the eight DIP switches located at SW1 on the XT-ML605 board can be read by software at the location specified in Table 4–4 on page 23. The switches are numbered 1 through 8, where switch 1 is the least significant bit when read by software. A switch in the "on" position reads as a 1 and a switch in the "off" position reads as a 0. Switch 8 is the boot select switch (Section 2.7 on page 7) and switch 7 is reserved by Tensilica and must be in the "off" (0) position.

4.2.4 Ethernet

The Ethernet MAC on the XT-ML605 board is essentially software compatible with the OpenCore WISHBONE Ethernet controller (refer to www.opencores.org).

If the Ethernet MAC is used, by convention the DIP switches described in Section 4.2.3 "DIP Switches" select the lower 6 bits of the MAC address, allowing up to 64 of these boards on the same network segment. The upper 34 bits of the MAC address are constant, and are generally given the value allocated by Tensilica: `00:50:c2:13:6f:xx`. Switches 1-2 set the two least significant bits of MAC address. Thus, the following range of MAC addresses can be used:

```
00:50:C2:13:6F:00 to 00:50:C2:13:6F:3F
```

The Ethernet MAC uses the external interrupt 1 pin.

4.2.5 LCD Display

The XT-ML605 board has a 2X16 character Mytech MOC-16216B-B LCD display. The LCD is operated in 4-bit mode and acts as a write-only device. For more information about the LCD, consult the data sheet, which can be accessed from:

```
http://www.mytechcorp.com/cfdata/productFile/File1/MOC-16216B-B-A0A04.pdf
```

4.2.6 Board Specific FPGA Registers

A number of board specific registers appear in the I/O block at offset 0x0d020000. These are organized as shown in Table 4–4:

Table 4–4. Board Specific FPGA Register Layout

I/O Block Offset	Size	Access	Device
0x0d020000	4 bytes	read-only	Build date code (BCD coded as mmddyyyy)
0x0d020004	4 bytes	read-only	Processor clock frequency in Hz
0x0d020008	4 bytes	read/write	Board LEDs (bits 0 to 3 corresponding to LEDs DS15, DS14, DS22 and DS21 respectively, off=0, on=1)
0x0d02000c	4 bytes	read-only	DIP switch (SW1) status: bits 0 .. 7 correspond to switches 1 to 8(left to right, off=0, on=1)
0x0d020010	4 bytes	write-only	Board reset (when writing 0x0000DEAD)
0x0d020014	65516 bytes	(reserved)	(reserved)

4.2.7 USB

The XT-ML605-D daughter board supports a peripheral USB port. The USB controller uses external interrupt number 5 of the processor. The device used is Cypress Semiconductor CY7C67300. Consult the device data sheet for more details at

<http://www.cypress.com/?rID=14164>. The same device can also be used to control the host USB port on the daughter card, but it is currently not supported.

4.3 Audio Specific Interfaces

This section describes the audio specific interfaces which are only available when a processor containing HiFi 2, HiFi EP, or ConnX D2 ISA options is configured. These interfaces are used to control analog and digital audio devices on the XT-ML605-D daughter board. They are also used to move audio data between the processor and audio devices. A detailed description for the actual devices on the daughter board can be found in Section 5.2.10 “Stereo Audio Codec (AIC)”, Section 5.2.11 “HDMI Transmitter” and Section 5.2.12 “Clock Synthesizer” on page 41.

4.3.1 Audio I²S Out

The audio I²S output interface is used by the processor to write audio data to stereo audio codec (AIC) and High Definition Multimedia Interface transmitter (HDMI) devices. The controller can support up to four I²S channels. The data bitwidth is programmable to 16 or 32 bits. It uses external interrupt pin 2 of the processor for FIFO interrupts. Two

types of FIFO interrupts are possible, FIFO underrun interrupt and FIFO level interrupt. The FIFO underrun interrupt occurs when audio output stream completely runs out of data, whereas FIFO level interrupt is triggered when the FIFO level falls below a preset value. This value is software programmable and can be read and written via Config register, refer to Table 4–5 for details. The controller is memory mapped at offset 0x0d080000 of system I/O block. For example, if the system I/O block base address is 0x90000000, then the base address of the I²S output registers is 0x9d080000. Table 4–5 explains the register mapping and functionality.

Table 4–5. I²S Output Register Mapping

Register Name	Offset	Read/Write	Reset Value	Description
Version	0x0	Read Only	32'h0000100	Version number
Config	0x4	Read and Write	32'b0	<ul style="list-style-type: none"> ■ Configuration Register ■ Bit[31:28] Enable controls for I²S channels. Writing 1 enables the channel and writing 0 disables it. Bits 28,29,30 and 31 enable/disable I²S channels 0,1,2 and 3 respectively. ■ Bit[27:24] Sets the fifo level in which an interrupt will be taken based on the fifo size(8192 entries). The value is set by fifo size >> Bit[27:24]. For example, a value of 3 indicates that an interrupt will be flagged when the fifo falls below1024 entries ■ Bit[21:16] The resolution of the audio stream, currently only 32-bit resolution is supported. ■ Bit[15:8] RATIO bits - These bits must be set based on the transmit data rate, see Section 4.3.2 on page 25 ■ Bit[2] ■ 0: Left channel stored on even addresses ■ 1: Left channel stored on odd addresses ■ Bit[1] Interrupt Enable ■ Bit[0] Transmitter Enable
Interrupt Mask	0x8	Read and Write	2'd0	<ul style="list-style-type: none"> ■ Bit 0 when '0' masks the fifo underrun interrupt ■ Bit 1 when '0' masks the fifo level interrupt
Interrupt Status	0xc	Read and Write	2'd0	<ul style="list-style-type: none"> ■ Bit 0 when '1' indicates a fifo underrun interrupt has occurred ■ Bit 1 when '1' indicates a fifo level interrupt has occurred ■ *Interrupts can be cleared by writing a '1' to the status register at the appropriate interrupt bit.
I ² S Chan 0 Data	0x10	Write Only	32'b0	Data to be written to channel 0 of I ² S interface.

Table 4–5. I²S Output Register Mapping (continued)

Register Name	Offset	Read/Write	Reset Value	Description
I ² S Chan 1 Data	0x14	Write Only	32'b0	Data to be written to channel 1 of I ² S interface.
I ² S Chan 2 Data	0x18	Write Only	32'b0	Data to be written to channel 2 of I ² S interface.
I ² S Chan 3 Data	0x1C	Write Only	32'b0	Data to be written to channel 3 of I ² S interface.

4.3.2 I²S Transmitter Config RATIO Bits

The RATIO bits in the I²S Config register depend on the transmit data rate, use the following equation to determine the RATIO value:

$$\text{RATIO} = (\text{I}^2\text{S_bus_clock} - (\text{sampling_frequency} * \text{bit_resolution} * 8)) / (\text{sampling_frequency} * \text{bit_resolution} * 4)$$

I²S_bus_clock is generated by the clock synthesizer device on the XT-ML605-D daughter board. It is possible to program the clock synthesizer and set the ratio bits accordingly to support multiple audio sampling frequencies. See Section 5.2.12 on page 41 for details related to the clock synthesizer device.

4.3.3 Audio I²S In

The Audio I²S input interface brings the recorded audio data from AIC to the processor. It uses a single I²S channel. The data bitwidth is programmable to 16 or 32 bits. External interrupt pin 3 on the processor is used for I²S input interrupts. Again, two types of FIFO interrupts are possible, FIFO overrun interrupt and FIFO level interrupt. The FIFO overrun interrupt occurs when FIFO input data completely fills the FIFO and it overflows, whereas FIFO level interrupt is triggered when the FIFO level is greater than a preset value. This value is software programmable and can be read and written via the Config register, refer to Table 4–6 for details. The controller is memory mapped at offset 0x0d088000 of system I/O block. For example, if the system I/O block base address is 0x90000000, then the base address of the I²S input registers is 0x9d088000. The I²S input controller register mapping and functionality is described in Table 4–6.

Table 4–6. I²S Input Register Mapping

Register Name	Offset	Read/Write	Reset Value	Description
Version	0x0	Read Only	32'h0000100	Version number
Config	0x4	Read and Write	32'b0	<ul style="list-style-type: none"> ■ Configuration Register ■ Bit[31:28] Reserved ■ Bit[27:24] Sets the fifo level in which an interrupt will be taken based on the fifo size(8192 entries). The value is set by $\text{fifo size} \gg \text{Bit}[27:24]$. For example, a value of 3 indicates that an interrupt will be flagged when the fifo level is greater than 1024 entries ■ Bit[21:16] The resolution of the audio stream, currently only 32-bit resolution is supported. ■ Bit[15:8] Reserved ■ Bit[2] ■ 0: Left channel stored on even addresses ■ 1: Left channel stored on odd addresses ■ Bit[1] Interrupt Enable ■ Bit[0] Receiver Enabled
Interrupt Mask	0x8	Read and Write	2'd0	<ul style="list-style-type: none"> ■ Bit 0 when '0' masks the fifo overrun interrupt ■ Bit 1 when '0' masks the fifo level interrupt
Interrupt Status	0xc	Read and Write	2'd0	<ul style="list-style-type: none"> ■ Bit 0 when '1' indicates a fifo overrun interrupt has occurred ■ Bit 1 when '1' indicates a fifo level interrupt has occurred ■ *Interrupts can be cleared by writing a '1' to the status register at the appropriate interrupt bit.
I ² S Data	0x10	Read Only	32'b0	Data read from the I2S interface.

4.3.4 I²C Master

The I²C master interface is provided to control AIC, HDMI and clock synthesizer devices on the XT-ML605-D daughter board. The I²C interface uses external interrupt number 4 of the processor. The controller is software compatible with the OpenCore WISHBONE I²C master. Refer to the specification document at www.opencores.org for details. The control registers are mapped to the base address location of `<io_bypass_base_address> + 0x0d090000`. For example, if the system I/O block base address is `0x90000000`, then the base address of the I²C master registers is `0x9d090000`. The details of registers mapping and their descriptions are located in the specification document.

4.3.5 SPI Interface

The SPI interface can be used to control the AIC device on the xtml605-D daughter board. SPI registers are memory mapped to the base address location of `<io_bypass_base_address> + 0x0d0a0000`. For example, if the system I/O block base address is `0x90000000`, then the base address of the SPI registers is `0x9d0a0000`. SPI registers are explained in Table 4–7.

Table 4–7. SPI Register Mapping

Register Name	Offset	Read/Write	Reset Value	Description
SPI_START	0x0	Read and Write	1'b0	Begins serial transfer on the SPI interface.
SPI_BUSY	0x4	Read Only	1'b0	Indicates that the SPI interface is busy.
SPI_DATA	0x8	Read and Write	32'b0	Data to be written to the SPI interface.

4.4 Booting from RAM or Flash

Use the SW1 DIP switch number 8 on the XT-ML605 board to map either the SDRAM or the Flash memory to the address space that contains the reset vector. See Section 2.7 on page 7 for details.

This mapping can be useful for a software developer who is developing a complete run-time system and wants to easily iterate between different versions of code, or to burn a completed image onto the board.

Booting from SDRAM allows the software developer to download complete code, including a reset vector, to the XT-ML605 board using the Xtensa OCD Daemon. This avoids the overhead of programming Flash during initial code development and allows the debugger to set software breakpoints in the boot code.

After the application has been fully developed and debugged it can be burned into the Flash memory. The Tensilica core specific tools package provides a simple program, *flasher*, to burn an image into the flash. It is written specifically for the flash on XT-ML605 board.(see Section 4.7 “Programming the Flash”).

4.5 Developing Software for the XT-ML605 Board

To develop code for the XT-ML605 board, the Tensilica tools package includes header files, a library of simple drivers, a set of Linker Support Packages (LSPs), and example source code. This section describes locations of files relative to `<xtensa_tools_root>` and `<xtensa_root>` defined in Chapter 3, “Getting Started” on page 11.

4.5.1 Using the Xtensa Board-Independent API (*xtbsp.h*)

The XT-ML605 support package provides a board-independent application programming interface (API) that allows basic board information and functionality to be accessed by a program that knows nothing about which board it is running on, and therefore may easily be compiled and linked for any supported board. The board-independent API is defined in the `xtbsp.h` header located at:

```
<xtensa_tools_root>/xtensa-elf/include/xtensa/xtbsp.h
```

The API functions are clearly described in the header itself.

To use `xtbsp.h`, add the following to your program:

```
#include <xtensa/xtbsp.h>
```

and link with one of the XT-ML605 specific LSPs (see Section 4.5.3 “Linking Code for the XT-ML605 Board”). These LSPs automatically link your program to the board specific library `libxtml605.a` which contains the API implementation for this board.

The `xtbsp` example programs use this API to achieve board-independence. Their source code is found in this directory:

```
<xtensa_tools_root>/xtensa-elf/src/boards/examples/xtbsp
```

Only the most basic functionality available on most boards is supported, such as reporting the processor clock frequency or setting the baud rate and character I/O over the UART (polled mode only). See `xtbsp.h` for details.

4.5.2 Using the XT-ML605 Board-Specific Header (*xtml605.h*)

For features specific to the XT-ML605 board, it is helpful to include this board-specific header:

```
<xtensa_tools_root>/xtensa-elf/include/xtensa/xtml605/xtensa/xtml605.h
```

It can be included by adding the following to your program:

```
#include <xtensa/xtml605.h>
```

and compiling with:

```
-I<xtensa_tools_root>/xtensa-elf/include/xtensa/xtml605
```

The above header and the manufacturers' documentation for each peripheral device should make it easy to develop drivers specific to your operating system or application.

The source code for the board specific library can be helpful in understanding the board and provides examples of how to write drivers and applications that directly access the boards registers and devices. The source code is provided with the Tensilica tools in these locations:

```
<xtensa_tools_root>/xtensa-elf/src/boards/  
<xtensa_tools_root>/xtensa-elf/src/boards/xtml605/
```

4.5.3 Linking Code for the XT-ML605 Board

Board-specific Linker Support Packages (LSPs) make it easier to develop code for use with the XT-ML605 board. These LSPs direct the linker according to the board's memory map. An LSP is specified with the `-mlsp=<LSP>` command line option to `xt-xcc` (see Example 1 and 2 following).

A set of standard LSPs are supplied with the core specific tools package. Included are board specific run-time LSPs (`xtml605-rt`, `xtml605-rt-rom`) that link a board support library (`libxtml605.a`) in addition to the Tensilica default runtime libraries. The library provides information about the board, initialization, and simple drivers for some devices such as the UART. This includes a complete implementation of the board independent API discussed in Section 4.5.1 on page 28.

The `-rom` LSP variants generate images that can be stored in Flash (reset code unpacks code and data to RAM); whereas the others are typically used for programs downloaded and debugged using OCD (with the XT-ML605 board boot mode select switch set for booting from SDRAM).

Standard LSPs that you may consider using with XT-ML605 board include:

- `xtml605-rt` — For use with OCD. Includes the default runtime. Standard input and output are directed to the serial port.
- `xtml605-rt-rom` — For Flash images (also debuggable using OCD). Otherwise, this is identical to the `xtml605-rt` LSP.
- `min-rt` — For use with OCD. Includes the default runtime, but no board-specific peripheral support. Standard output goes to a small circular buffer in memory.
- `min-rt-rom` — For Flash images (also debuggable using OCD). Otherwise, this LSP is identical to `min-rt`.
- `nort` — For use with OCD. This LSP does not include any runtime or vector code. Use this LSP if you intend to write or provide everything including vector routines.
- `nort-rom` — For Flash images (also debuggable using OCD). Otherwise, this LSP is identical to `nort`. Your reset code must unpack code and data to RAM.

- `gdbio` — For use with OCD and GDB (which may be under control of Xplorer). This LSP includes the default runtime. Standard output goes to the GDB console. File access using the standard C library is also available; such requests are serviced by GDB on behalf of the target. GDB must be connected to the target using OCD for standard output and file access to work.
- `gdbio-rom` — For Flash images. Otherwise this LSP is identical to the `gdbio` LSP.

For more detailed information, refer to the *Xtensa Linker Support Packages (LSPs) Reference Manual*.

The following command line examples illustrate how to use the LSPs. Source code for this and other example programs can be found in the Tensilica tools package located in:

```
<xtensa_tools_root>/xtensa-elf/src/boards/examples/xtbsp/
```

Example 1

To compile and link the `announce` program for use with OCD, type the following:

```
$ xt-xcc -g -o announce -mlsp=xtml605-rt announce.c
```

This creates an executable that can be downloaded and run on the XT-ML605 board as described in Section 4.5 “Developing Software for the XT-ML605 Board” on page 27.

Note: Using this LSP requires that the Boot Mode Select switch be set for SDRAM (Section 2.7 “Boot Select Switch” on page 7). This ensures that the reset vector code can be loaded and executed.

Example 2

To compile and link the `announce` program and generate a binary image for Flash, type the following:

```
$ xt-xcc -g -o announce -mlsp=xtml605-rt-rom announce.c
$ xt-objcopy -O binary announce announce.bin
```

This creates a binary image `announce.bin` that can be burned into Flash according to instructions in Section 4.7 “Programming the Flash” on page 31. When booting from flash (see Section 4.4 on page 27) the program will run automatically after reset.

4.6 Debugging Code on the XT-ML605 Board

Tensilica's debugging environment is composed of several components that work together to provide the designer with useful program development tools, including:

- **Xtensa Xplorer** — The integrated development environment provides a source linked GUI debugging environment which can be used with an Xplorer project or as a stand-alone debugger. It operates as a front-end to the `xt-gdb` debugger.
- **`xt-gdb` Debugger** — Implements the command structure for controlling the user program on the emulation board. This can be used directly via command line, or indirectly via the Xplorer GUI debugger interface.
- **XOCD** — A combination of hardware (OnCE JTAG probe) and software (Xtensa OCD Daemon) tools providing a communications link for downloading programs, examining and modifying memory and registers, and other functions used by the debugger.

You can debug via command-line as described in Section 3.4 “Running a Program Using OCD and GDB on the Command Line” on page 16. If you prefer you can use the Xplorer IDE (see *Xtensa Software Development Toolkit User's Guide* and Xplorer's on-line help).

To run the example discussed in Section 4.5.3 “Linking Code for the XT-ML605 Board” on page 29, `xt-gdb` can be invoked and a program loaded and started, as follows:

1. Type `$ xt-gdb announce`.
2. Type `target remote [<ip_address>]:20001 0` (see Section 3.4 on page 16).
3. Type `load`.
4. Type `continue` or `c`.

4.7 Programming the Flash

The XT-ML605 package contains an example program named `flasher` which, when run via `xt-gdb` under OCD control, allows you to program a binary image into the Flash. The code to be burned into the Flash must have been linked with a romming LSP as described in Example 2 of Section 4.5.3 on page 29, be in binary form, and must not exceed 16MB in size. It will be loaded into available memory via `xt-gdb`.

The `flasher` program is provided as a pre-built executable at:

```
<xtensa_root>/xtensa-elf/bin/xtml605/flasher
```

The source code is provided at:

```
<xtensa_tools_root>/xtensa-elf/src/boards/examples/xtml605/flasher.c
```

The Flasher program uses the UART device to display the status of programming the Flash (programming can take place without the UART, but you will need to use GDB in conjunction with the flasher source code to determine if programming was successful). Connect the USB to UART port J21 on the XT-ML605 board to an RS-232 terminal or PC running a terminal emulator (for example, *HyperTerminal* on Windows, or *minicom* on Linux) using USB to mini USB cable. Make sure that the CP210x VCP drivers are installed on the host PC, as discussed in section Section 4.2.2 "UART". Set the terminal or emulator for 38400 baud, 8 data bits, 1 stop bit, no parity.

Following are instructions for programming the Flash:

1. Type: `$ xt-gdb flasher`.
2. Type: `target remote [<ip_address>]:20000 0` (see Section 3.4 on page 16).
3. Type: `reset`.
4. Type: `load`.
5. (Optional) Type: `set image_base = <address for image>`
The default address is 0x61000000 (0xD1000000 on the Diamond Core DC_232L), at which there is enough unused RAM to contain the entire Flash (16 MB). Select any unused area of RAM big enough to contain the entire image. If there is < 16 MB unused RAM at the selected address, you must also set `image_size` (see step 7).
6. Type: `restore <binary image filename> binary image_base`
Note: The binary file must not exceed 16 MB in size.
7. (Optional) Type: `set image_size = <size of image>`
Default is 0x100000 to program the entire 16 MB Flash. Providing the actual size speeds up programming. The restore command reports the address range of the image as "`<base> to <top>`". Subtract `<base>` from `<top>` to obtain the size of the image in bytes. You can have `xt-gdb` do the math by specifying `<size of image>` as "`<top> - <base>`" in this command. **Note: When you are using the flasher example for the very first time, do not set `image_size` and leave it to default 0x100000. This is required to get the flash state machine to a known state. After you have run it once, for all subsequent runs you can safely set the image size to the actual size for speed up.**
8. Type: `continue` or, `c`
9. Watch the terminal for the status, such as when programming is finished. It will take up to a few minutes, depending on the size of the image.

For example, to flash an mp3 file named `test.mp3` to the beginning of the flash, type the following sequence of commands

```
$ xt-gdb flasher
target remote <ip_address>:20000 0
reset
load
restore <path_to_test_file>/test.mp3 binary <image_base>
set image_size = <size_of_test.mp3>
c
```

4.8 Using the UART

As discussed in section Section 4.2.2 “UART” the host PC running the terminal emulator must have the CP210x VCP drivers installed. Refer to the Xilinx documentation for more details.

The easiest way to use the UART is via the board support library and the board independent API in `<xtbsp.h>` as described earlier in Section 4.5.1. This is demonstrated by the example programs `announce` and `memtest`. The source for `announce` can be found under

```
<xtensa_tools_root>/xtensa-elf/src/boards/examples/xtbsp
```

and the source for `memtest` can be found under

```
<xtensa_tools_root>/xtensa-elf/src/boards/examples/xtml605
```

(although `memtest` is board-specific, it accesses the UART in a board independent way).

For details about the UART device, consult the National Semiconductor 16550 data-sheet provided at <http://www.national.com/ds/PC/PC16550D.pdf>.

If you wish to bypass the API and set the baud rate explicitly, note that the divisor setting is based on the frequency at which the core is running. The frequency may vary per core and is available via the API function `xtbsp_clock_freq_hz()` or by directly reading a register on the board (see Section 4.5.2 “Using the XT-ML605 Board-Specific Header (`xtml605.h`)” on page 28). For most Diamond cores the clock frequency is 50 MHz.

Note that the RTS and CTS signals are not connected on the XT-ML605 board, therefore the XT-ML605 board does not support hardware flow control.

5. XT-ML605-D Daughter Board

The XT-ML605-D daughter board is designed to integrate audio functionality as well as provide an OnCE JTAG probe interface to the Tensilica core on the XT-ML605 board. It also provides flash memory and peripheral USB interfaces. The daughter board contains a TI320AIC Stereo Audio CODEC which allows analog audio input and output functionality to/from Tensilica's HiFi 2 Audio Engine. An Analog Devices AD9889B HDMI transmitter is used to provide multichannel digital audio output. The daughter board also contains a TI CDCE706 clock synthesizer device that provides clocks to audio interfaces. This section provides detailed information regarding the features supported on the daughter board. It also provides instructions on how to operate and exercise the functionality of this reference platform.

5.1 Feature Set

The XT-ML605-D daughter board contains the following feature set.

Audio

- Analog audio playout through Headphone and Line out
- Digital audio playout through HDMI port
- Analog audio capture through Mic and Line in
- Clock synthesizer for different sampling frequencies

System

- OCD JTAG interface for debugging
- Peripheral USB interface
- 16 MB Flash memory

5.2 Hardware Description

Figure 5–7 and Figure 5–8 show the top and the bottom view of the XT-ML605-D daughter board and all external interfaces.

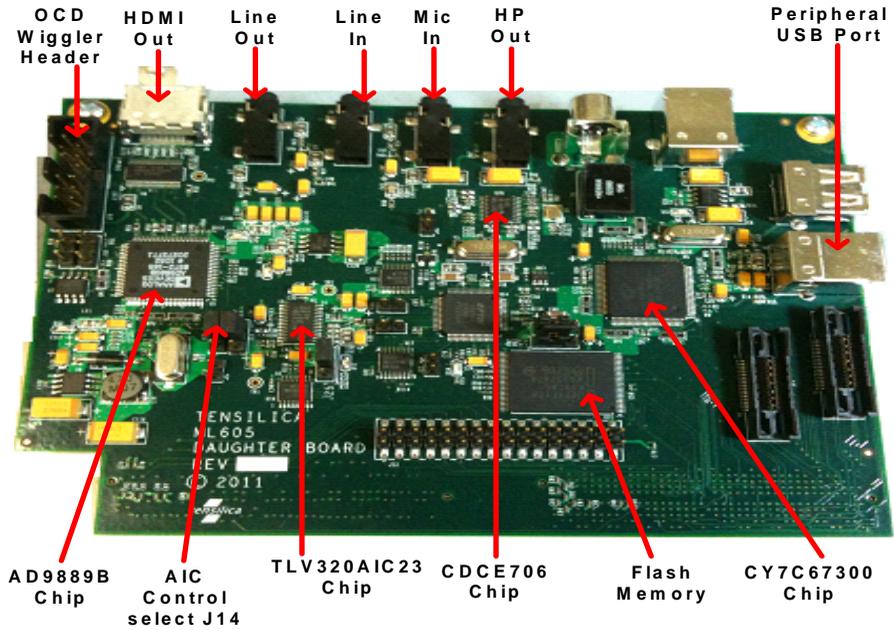


Figure 5-7. Top View of the XT-ML605-D Daughter Board

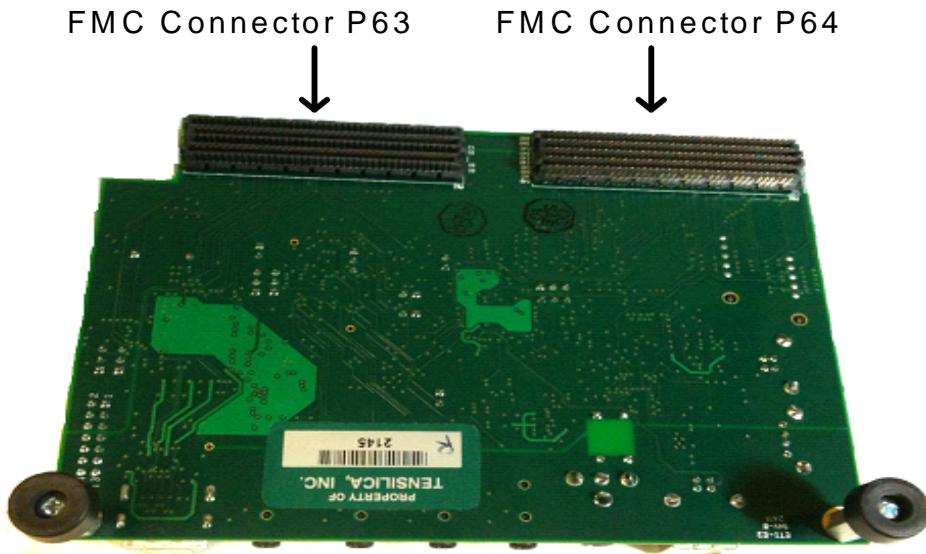


Figure 5–8. Bottom View of the XT-ML605-D Daughter Board

5.2.1 Headphone (HP)

This interface can be used to connect external headphones or speakers to the device for listening to the audio content.

5.2.2 Microphone (MIC)

Use this interface to plug in the microphone when recording audio.

5.2.3 Line-In (LI)

Use Line-In to record from an external audio source.

5.2.4 Line-Out (LO)

Use to play audio content to external amplifiers.

5.2.5 HDMI Port

Use the HDMI port to connect a digital audio receiver or HDMI capable device (for example, HDTV). This port can be used to play multichannel digital audio content.

5.2.6 USB Interface

The daughter board provides a peripheral USB interface using a CY7C67300 device. The USB controller uses external interrupt 5 of the processor as described in Section 4.2.7 “USB” on page 23. The host USB port on the daughter board is currently not supported.

Note: There are two peripheral USB ports on the XT-ML605-D daughter board, J1 and J12. Note that port J12 is for Tensilica use only and that port J1 is the user USB interface. The location of the user USB port is also shown in figure 5–7.

5.2.7 Flash Memory

The XT-ML605-D daughter board contains 16MB of 16-bit wide flash memory. The device is Numonyx Strata Flash JS28F128. You can boot the processor using this flash device by mapping it to the ROM area using the boot select switch as described in Section 2.7 “Boot Select Switch”. The flash is also mapped in the System I/O block as described in Section 4.2 “System I/O Block”.

5.2.8 FMC Connectors

The XT-ML605-D daughter board connects to the XT-ML605 base board using FMC connectors. These are located on the bottom side of the XT-ML605-D daughter board. The daughter board use both the high density FMC (J64) and low density FMC (J63) expansion slots on the XT-ML605 base board. As shown in Figure 5–9, the daughter board should be installed facing away from the XT-ML605 base board. Connector P63 on the daughter board should mate with connector J63 on the base board and connector P64 on the daughter board should mate with connector J64 on the base board. Make sure that these connectors are connected appropriately with correct pin alignment before using the XT-ML605-D daughter board.



Figure 5–9. Connection of XT-ML605-D Daughter Board and XT-ML605 Board

5.2.9 OnCE JTAG Probe Header

The Tensilica processor provides an On-Chip Debug (OCD) module. The OCD module supports a JTAG interface to a host-based debugger through a JTAG access device. The OCD probe can be connected through header J19 of XT-ML605-D daughter board. Figure 5–10 shows the OnCE JTAG probe header J19, and Table 5–8 below lists its pin assignments.

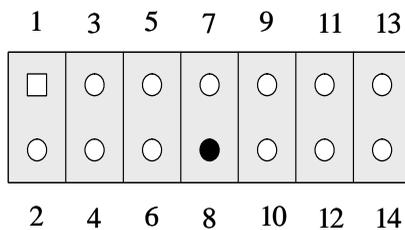


Figure 5–10. JTAG Probe Header J19

Table 5–8. J19 JTAG Probe Header Pin Number and Function

Pin Number	Function	FPGA Pin
1	JTDI	AF30
2	GND	-
3	JTDO	AG30
4	GND	-
5	JTCK	A10
6	GND	-
7	-	-
8	KEY	-
9	RESET_OUT	J30
10	JTMS	U27
11	Vcc	-
12	-	-
13	-	-
14	JTRST	K27

5.2.10 Stereo Audio Codec (AIC)

The daughter board provides a TI TLV320AIC device for analog audio input and output functionality. Headphone and line out ports are used for analog audio output; line in and microphone ports are used for analog audio input. Use of SPDIF port on the daughter board is currently not supported. The device can be controlled using the SPI interface. Output audio data is written to AIC using channel 0 of I²S transmitter. Input audio data is read using I²S receiver. SPI and I²S in/out interfaces are described in Section 4.3 “Audio Specific Interfaces” on page 23. I²S transmitter and receiver bus clocks are generated using a clock synthesizer device, which makes it possible to support multiple sampling frequencies. See Section 5.2.12 “Clock Synthesizer” on page 41 for details. More information regarding AIC registers and device programming are in the TLV320AIC23 Stereo Audio CODEC data manual, available at <http://focus.ti.com/lit/ds/sym-link/tlv320aic23.pdf>.

5.2.11 HDMI Transmitter

The daughter board provides an Analog Devices AD9889B HDMI transmitter, which can be used to play multichannel-digital audio output through the HDMI port. The device is controlled using the I²C interface. It responds to 7-bit address 0111101. Output audio data can be written to the HDMI device using four or less I²S transmitter channels. I²C and I²S out interfaces are described in Section 4.3 “Audio Specific Interfaces” on

page 23. As the I²S transmitter uses bus clock generated by clock synthesizer, it is possible to support multiple sampling frequencies by programming the clock synthesizer. In addition, the HDMI device also needs a pixel clock to function properly, even though it does not play any video. This pixel clock is also provided using the clock synthesizer device. See Section 5.2.12 “Clock Synthesizer” on page 41 for details. Refer to the AD9889B data sheet for more information regarding the HDMI registers and device programming, which is available at http://www.analog.com/static/imported-files/data_sheets/AD9889B.pdf.

5.2.12 Clock Synthesizer

Clock synthesizer device TI CDCE706 is provided on the daughter board for supporting different audio sampling frequencies. The device can be controlled using the I²C interface, and responds to 7-bit address 1101001. The clocks generated from the clock synthesizer are routed back to the FPGA and used as bus clocks for the I²S transmitter and receiver. I²S interfaces divide down on the bus clock to generate bit and word clocks for AIC and HDMI devices. The clock synthesizer also generates a pixel clock, which is required for the HDMI device. This clock is preset to 50.4 MHz by Tensilica and corresponds to a 640x480 video resolution. The pixel clock frequency should not be changed, otherwise the HDMI interface may not work properly.

The clock synthesizer makes it possible to support multiple audio sampling frequencies. For example, use the clock synthesizer to set the I²S bus clock to 12.288 MHz and set RATIO bits in I²S transmitter to 0, this sets up the AIC for 32-bit, stereo, 48KHz sampling frequency audio payout. Similarly, the I²S bus clock 11.2986 MHz and RATIO 0 correspond to 32-bit, stereo, 44.1 KHz sampling frequency. For details regarding clock synthesizer internal registers and programming, refer to the device data sheet at <http://focus.ti.com/lit/ds/symlink/cdce706.pdf>.

5.3 Jumper Settings

Install jumper on pins 1:2 of J14 to select SPI control for the AIC device. The other factory settings for the daughter card's jumper on pins are as follows:

- 1:2 of J15
- 1:2 of J23
- 1:2 of J24
- 2:3 of J25

There should be no jumpers on J9, J10, J16, J20, J21, J22 and J26. Ensure that the jumpers are set correctly before running any of the following example programs.

5.4 Running a Sample Audio Playout Program

The XT-ML605 package contains two example programs for audio playout, `sinewave` and `hdmi_sinewave`. The `sinewave` example produces a sine wave audio output through the headphone and line-out jacks, whereas the `hdmi_sinewave` example produces a sinewave through the HDMI port on the daughter card.

These programs are provided as pre-built executables in the Tensilica core package at:

```
<xtensa_root>/xtensa-elf/bin/xtml605/sinewave
<xtensa_root>/xtensa-elf/bin/xtml605/hdmi_sinewave
```

The source code is provided in the Tensilica tools package at:

```
<xtensa_tools_root>/xtensa-elf/src/boards/examples/xtml605/sinewave.c
<xtensa_tools_root>/xtensa-elf/src/boards/examples/xtml605/hdmi_sinewave.c
```

Following are step-by-step instructions for running the example audio playout programs using GDB and OCD.

1. Connect the daughter board to XT-ML605 board using the FMC connectors as shown in Figure 5–9. Make sure that you connect the daughter board to the right connectors with appropriate pin alignment.
2. Make sure that the jumper settings are correctly done as described in Section 5.3 “Jumper Settings”.
3. Plug the OCD probe to connector J19 of XT-ML605-D daughter board and make sure it is connected to the OCD daemon host machine (or Ethernet). Be sure to set the DIP switch to boot from SDRAM as described in Section 2.7 “Boot Select Switch” on page 7. After the board is powered up, you should be able to connect to the target’s Tensilica core from the `xt-gdb` debugger or XtenSA Explorer (not shown here). Start the debugger as you normally would (see Section 3.4 “Running a Program Using OCD and GDB on the Command Line” on page 16). For the `sinewave` example type:

```
$ xt-gdb <xtensa_root>/xtensa-elf/bin/xtml605/sinewave :
```

For `hdmi_sinewave` example type:

```
$ xt-gdb <xtensa_root>/xtensa-elf/bin/xtml605/hdmi_sinewave
```

4. To connect to the target, at the GDB prompt, type the following:

```
target remote [<ip_address>]:20000 <target_memory_address>
```

where `<ip_address>` is the IP address (or DNS name) of the PC running the XtenSA OCD Daemon (this may be omitted if that is localhost, the same machine running `xt-gdb`), `20000` is the default port number, and `<target_memory_address>` is any 16 byte aligned address at which there is RAM. If you have trouble connecting to the target, the XtenSA OCD Daemon window may display an informative error message on the lines that start with asterisks.

5. Load the program into RAM by typing:

```
load
```

Note that this also sets the PC register to the reset vector.

6. Start the execution using the command

```
continue
```

A continuous sinewave should play out of the headphone and line-out jacks when running the `sinewave` example. When running the `hdmi_sinewave` example, you should hear the sinewave playing through the HDMI port.

You can regain control by pressing Ctrl-C.

For more information about using the debugger and debugging with the on-chip debug mode, refer to the *Tensilica On-Chip Debugging Guide* and *GNU Debugger User's Guide*.

5.5 Running a Sample Audio Recording Program

The XT-ML605 package also contains an example program, `soundclip`, which records a short audio clip using the microphone jack and then plays the recorded audio output through the headphone jack on the daughter card.

The `soundclip` program is provided as a pre-built executable in the Tensilica core package at:

```
<xtensa_root>/xtensa-elf/bin/xtml605/soundclip
```

The source code is provided in the Tensilica tools package at:

```
<xtensa_tools_root>/xtensa-elf/src/boards/examples/xtml605/soundclip.c
```

Following are step-by-step instructions for running the `soundclip` program using GDB and OCD.

1. Connect the daughter board to XT-ML605 board using the FMC connectors as shown in Figure 5–9. Make sure that you connect the daughter board to the right connector with appropriate pin alignment.
2. Make sure that the jumper settings are correctly done as described in Section 5.3 “Jumper Settings”
3. Plug the OCD probe to connector J19 of XT-ML605-D daughter board and make sure it is connected to the OCD daemon host machine (or ethernet). Be sure to set the DIP switch to boot from SDRAM as described in Section 2.7 “Boot Select Switch” on page 7. Also make sure that you have the USB to UART port connected up to your PC with the baud rate set at 38400 with no flow control because you will be interacting with the test program using the serial console. You may need to install the CP210x VCP drivers on the PC first if it hasn't been done already. After the board is powered up, you should be able to connect to the target's Tensilica core

from the xt-gdb debugger or Xtensa Explorer (not shown here). Start the debugger as you normally would (see Section 3.4 “Running a Program Using OCD and GDB on the Command Line” on page 16).Type:

```
$ xt-gdb <xtensa_root>/xtensa-elf/bin/xtml605/soundclip
```

- To connect to the target, at the GDB prompt, type the following:

```
target remote [<ip_address>]:20000 <target_memory_address>
```

where *<ip_address>* is the IP address (or DNS name) of the PC running the Xtensa OCD Daemon (this may be omitted if that is localhost, the same machine running xt-gdb), 20000 is the default port number, and *<target_memory_address>* is any 16 byte aligned address at which there is RAM. If you have trouble connecting to the target, the Xtensa OCD Daemon window may display an informative error message on the lines that start with asterisks.

- Load the program into RAM by typing:

```
load
```

Note that this also sets the PC register to the reset vector.

- Start the execution using the command

```
continue
```

- Attach a microphone to the microphone jack of the daughter card and the headphones to the headphone jack.

At this point, the serial console should read:

```
Welcome to the Tensilica ADC Recorder Test
This application will record a short clip
and then play it out on the headphone output
```

```
Hit 'Enter' to begin recording
```

- Press the Enter key to begin recording. As the recording proceeds, you will see a series of dots that indicate the progress of the recording:

```
Recording a short
clip.....
.....
```

- Once the recording is complete, you will then be prompted to hit Enter again to begin playback. Press Enter to listen to what you recorded.

For more information about using the debugger and debugging with the on-chip debug mode, refer to the *Tensilica On-Chip Debugging Guide* and *GNU Debugger User's Guide*.

5.6 Writing Device Drivers for AIC and HDMI

The audio playback and recording example programs that Tensilica provides contain some useful C functions, which can be referred to when developing device drivers for AIC and HDMI devices.

5.6.1 Sample C Functions

The following C functions have been provided in `sinewave`, `hdmi_sinewave` and `soundclip` example programs to assist in transmitting data to HDMI/DAC and receiving data from ADC. You might find it convenient to copy them to your program.

- `int hifi_audio_aic_open()`
 Initializes the TLV320AIC chip by programming its internal registers using the SPI interface.
- `int hifi_audio_aic_init(int freq, int num_chan)`
 The two parameters are sampling frequency and number of channels. This function configures concerned I²S interfaces (only I²S transmitter for the `sinewave` example and both I²S and transmitter and receiver for the `soundclip` example). It also initializes the I²C interface, which programs the clock synthesizer with the desired sampling frequency. Some of the AIC internal registers are also modified based on input arguments.
- `int hifi_audio_hdmi_open()`
 Initializes I²C interface and then programs AD9889B chip internal registers to bring it up.
- `int hifi_audio_hdmi_init(int freq, int num_chan)`
 The two parameters are sampling frequency and number of channels. This function configures the I²S transmitter and AD9889B chip for desired sampling frequency and number of channels. It also programs the clock synthesizer accordingly.
- `int hifi_audio_aic_write(short *pcm_buf, int len_32bits)`
 Writes PCM data to the audio DAC. This function expects each audio sample to have 16-bit resolution and left/right channel data to be interleaved.
- `int hifi_audio_hdmi_write(int *pcm_buf, int len_32bits)`
 Writes PCM data to six audio channels on the HDMI using three I²S transmitter channels. This function expects audio samples to have 24-bit resolution, but stored as 32-bit integers. The data from six audio channels is assumed to be interleaved.
- `int hifi_audio_aic_stop(void)` and `int hifi_audio_hdmi_stop(void)`
 Stops the AIC and HDMI playback respectively.

5.6.2 Writing Data to and From Memory Through OCD

As `sinewave` and `hdmi_sinewave` programs contain hard-coded sinewave samples, they do not require an input data stream. If your program needs to preload binary data from a file into SDRAM and/or store out data afterward, the following GDB commands may be helpful.

To load a block of data from a file to the SDRAM on the board through OCD, use the following command at the gdb prompt:

```
restore <filename> binary <address>
```

To dump a block of data from the SDRAM on the board to a file through OCD, use the following command at the gdb prompt:

```
dump binary memory <filename> <address_start> <address_end>
```

Index

B		
button, SW5	7	
C		
Clock frequencies	6	
Communication with XT-ML605.....	1	
D		
DDR SDRAM	7	
speed	15	
Debugging code	31	
Default switch settings.....	5	
Definitions		
Notations.....	ix	
Terms	ix	
Developing code for ML-605	29	
DIP switches	22	
E		
Ethernet	22	
Ethernet MAC	8	
F		
Flash.....	21	
programming.....	31	
I		
Installing		
OCD Daemon	15	
J		
JTAG TAP controller.....	8	
L		
Linking code for XT-ML605	29	
Loading		
configurations.....	2	
with a Diamond core processor.....	11	
M		
Memory map.....	19	
O		
OCD Daemon		
installing	15	
OnCE JTAG		
tools required	12	
On-Chip debugger (OCD).....	8, 9	
P		
Preface	ix	
Programming the configuration.....	12	
PROM, programming.....	12	
R		
Reset	6	
S		
SW5 button	7	
System I/O block.....	20	
T		
Tools, required.....	2	
U		
UART	21	
operation	8	
using.....	33	
X		
xtbsp.h	28	
xt-gdb debugger.....	31	
XT-ML605 board		
clock frequencies	6	
communication with.....	1	
control of	1	
debugging code on.....	31	
developing and linking code for.....	29	
loading configurations	2	
loading with Diamond core.....	11	
xtml605.h	28	

