Media5200 User's Guide

MEDIA5200UG
Rev. 0, 5/2006

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064, Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 26668334
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including “Typicals”, must be validated for each customer application by customer’s technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

MEDIA5200UG
Rev. 0
5/2006

About This Book

This user’s manual describes the functionality of the Media5200 Multimedia Development for software and hardware developers.

The information in this book is subject to change without notice. Updates for this document may be found at:

http://www.freescale.com/mobileGT.

Organization

Following is a summary and brief description of the major sections of this manual:

- Chapter 1, “Introduction,” includes general descriptions of the features incorporated on the Media5200 Multimedia Development Platform.
- Chapter 2, “Kit Contents,” describes the contents of the Media5200 package.
- Chapter 3, “Getting Started,” describes the steps of powering on the Media5200, communicating to a host computer, and installing the desired Real-Time Operating System on the system.
- Chapter 4, “Hardware Description,” contains the block diagram and outlines all board settings.
- Chapter 5, “Boot Monitor,” describes the steps to download and flash code using Ethernet and UART protocols.
- Chapter 6, “FPGA Register Space,” provides the entire register map for the FPGA.

This manual includes the following two appendixes:

- Appendix A, “Build History,” provides an overview of various development builds of the Media5200 Multimedia Development Platform.
- Appendix B, “Glossary of Terms and Abbreviations,” contains an alphabetical list of terms, phrases, and abbreviations used in this book.

Additional Documentation

Additional useful documentation for the Media5200 system include:

- *MPC5200B User’s Manual*, (MPC5200BUG)
- Media 5200 Board Schematics
  - Media5200 MAIN Board Schematics
  - Media5200 AUDIO Board Schematics
  - Media5200 POWER Schematics

  - Media5200 GPS Schematics
  - Media5200 MOST Schematics
- Media5200 PCB Layout Files
- *U-Boot Quick Reference*
- Real-Time Operating System Partners
  - Green Hills Software - Integrity
  - QNX Software Systems Ltd. - Neutrino
  - Wind River Systems - VxWorks
  - Linux

Chapter 1 Introduction

The Media5200 Multimedia Development Platform is designed to provide a working hardware environment in which to evaluate the MPC5200B, develop software, and verify the performance of a fully integrated system using actual applications software. The board has built in graphics control, an integrated display as well as physical interface devices for the CAN, I2C and ETHERNET ports.

The MPC5200B microcontroller (MCU) has two external data/address bus structures. The LocalPlus Bus is used to address FLASH, SRAM, peripheral devices and other types of memory. The LocalPlus Bus can perform memory accesses in several multiplexed and non-multiplexed addressing modes. While it is possible to execute code residing in memory devices on the LocalPlus Bus, code execution is generally performed on the SDRAM bus which is specifically made to interface to Synchronous Single- and Double Data Rate Dynamic RAMs. In general, many MPC5200B systems are designed such that programs to be executed are stored as data in FLASH devices residing on the LocalPlus Bus. On system initialization, a program being executed on the LocalPlus Bus will copy the applications code to memory on the SDRAM Bus and then code execution will jump to a device on the SDRAM bus. The SDRAM bus is specifically designed to interface with the MPC5200B's internal instruction cache for superior instruction throughput.

The MPC5200B microcontroller has several external ports to interface the internal peripheral elements to external devices. In general, these ports can be configured to perform different functions. That is, a particular port can be configured to take on one of several different configurations. For instance, the TIMER module can appear on one of two different ports. Also, a port, such as Programmable Serial Controller Port 1, can be configured to be a UART, a CODEC, General Purpose Input/Output or an AC97 interface. The Media5200 Multimedia Development Platform does not provide for all possible configurations of the MPC5200B MCU ports. However, all of the internal peripheral elements are available on at least one port.

The Media5200 Multimedia Development Platform is actually intended to be used by engineers and programmers who are using high level software tools to write and debug software on a fully functional system. The onboard monitor program, U-Boot, is used to control loading programs, setting up communications protocols, loading and examining memory and other such functions. The U-Boot Monitor Program has very limited software debug capabilities. The Media5200 Multimedia Development Platform does provide a very effective platform to execute user applications code.

Chapter 2 Kit Contents

2.1 Kit Contents for the Media5200

The following contents are included in the Media5200 kit:

1. Media5200 Multimedia Development System with Base and Head Units
2. Power supply (input 100-240VAC, 50-60Hz; output 12VDC, 5A)
   a) USA adapter
   b) European adapter
   c) British adapter
   d) Australian adapter
3. Serial cable
4. GPS antenna (Synergy Systems P/N 10001699)
5. MPC5200B Development Kit CD
6. Green Hills Software
   a) Evaluation CDs
   b) Thumbdrive with demonstration software
7. QNX Systems Software Ltd
   a) Evaluation CDs
   b) Thumbdrive with demonstration software
8. Wind River Systems
   a) Evaluation CDs
   b) Thumbdrive with demonstration software
9. Linux demonstration thumbdrive
10. ALT software demonstration CD
11. Other paper collateral

Chapter 3 Getting Started

3.1 Introduction

This document walks through setup and installation of a desired real-time operating system on the mobileGT® Media5200 Multimedia Development Platform. Its purpose is for the developer to get started with the system as quickly as possible.

3.2 Serial Connection

A serial connection is used to communicate between the host terminal (user's PC) and the target system (mobileGT Media5200.) Included with the mobileGT Media5200 is a 9-pin null modem serial cable. Connect this cable from the UART INT0 port on the mobileGT Media5200 to the UART port on the host system.

3.3 Terminal Emulator Configuration

Use a terminal emulator to connect from the host to the mobileGT Media5200 Target System. HyperTerminal is distributed with Windows if no other is available. The terminal emulator must be set up with the following parameters:

- 115200 baud
- 8 data bits
- no parity
- 1 stop bit
- no flow control.

3.4 Power Connection

Power the mobileGT Media5200 Target System by connecting the power supply to the connector on the back of the target system.

3.5 Booting Up the System

The Media5200 U-Boot is the only image pre-programmed into the target flash at the factory. The user will need to install the desired RTOS using the provided thumb drives. On power-on, the default U-Boot banner will display informational text similar to

```
U-Boot 1.1.3 (Apr 25 2006 - 22:09:33)

CPU:   MPC5200 v2.2 at 396 MHz
       Bus 132 MHz, IPB 132 MHz, PCI 33 MHz
Board: Media5200 (FPGA 02090403)
```

```
I2C:   85 kHz, ready
DRAM:  128 MB
FLASH: 64 MB
PCI:   Bus Dev VenId DevId Class Int
        00  1c  10cf  201e  0380  3f
        00  1d  1057  5809  0680  00
GFX:   CoralP at 0x40000000

Autostarting. Press any key to abort...

Hit any key to stop autoboot:  0
=>
```

For UBOOT commands, refer to the *U-Boot Quick Reference* document provided on the MPC5200B Development Kit CD, or visit http://www.denx.de/wiki/DULG/Manual.

3.6 RTOS Installation

Read the end-user agreement before unpacking the USB thumb drives. Four real-time operating systems (RTOS) are included with the Media5200: Green Hills Software Integrity, Linux, QSSL QNX Neutrino, and Wind River VxWorks. Select the desired RTOS thumb drive and insert into the USB port on the Media5200 Target System, located on the front panel. Press the RESET button at the back of the Media5200.

Once the target system reboots, a script will begin installing the RTOS image located on the USB thumbscrew. The autoboot script will:

1. Erase the on-board flash.
2. Copy the RTOS image onto RAM
3. Copy image from RAM to on-board flash

Once the installation is completed, reboot the system again allow to autoboot. The system will now boot to the operating system.

Chapter 4 Hardware Description

4.1 Block Diagram

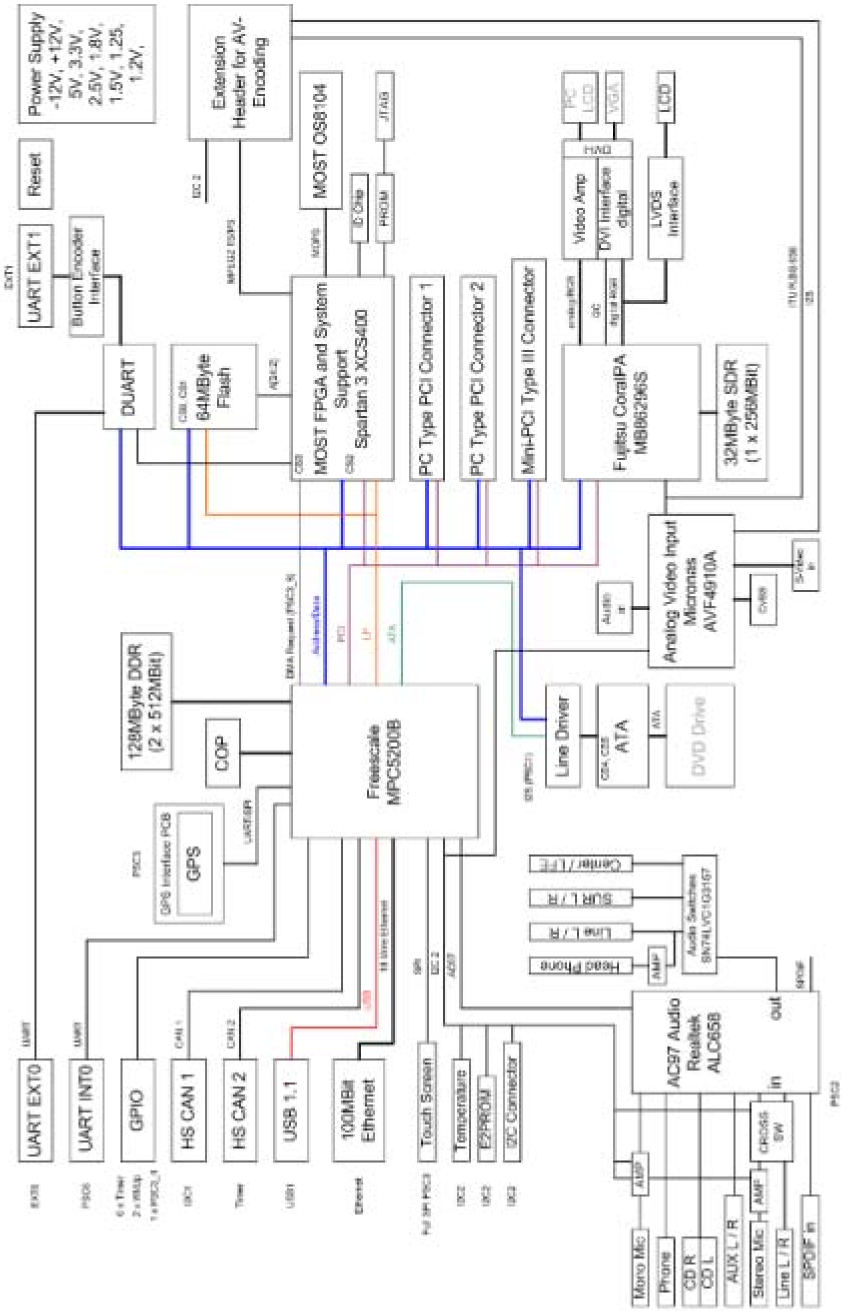


Figure 4-1. Media5200 Block Diagram

4.2 MAIN Board Switches, Jumpers, Connectors, and Headers

4.2.1 MAIN Board Switches

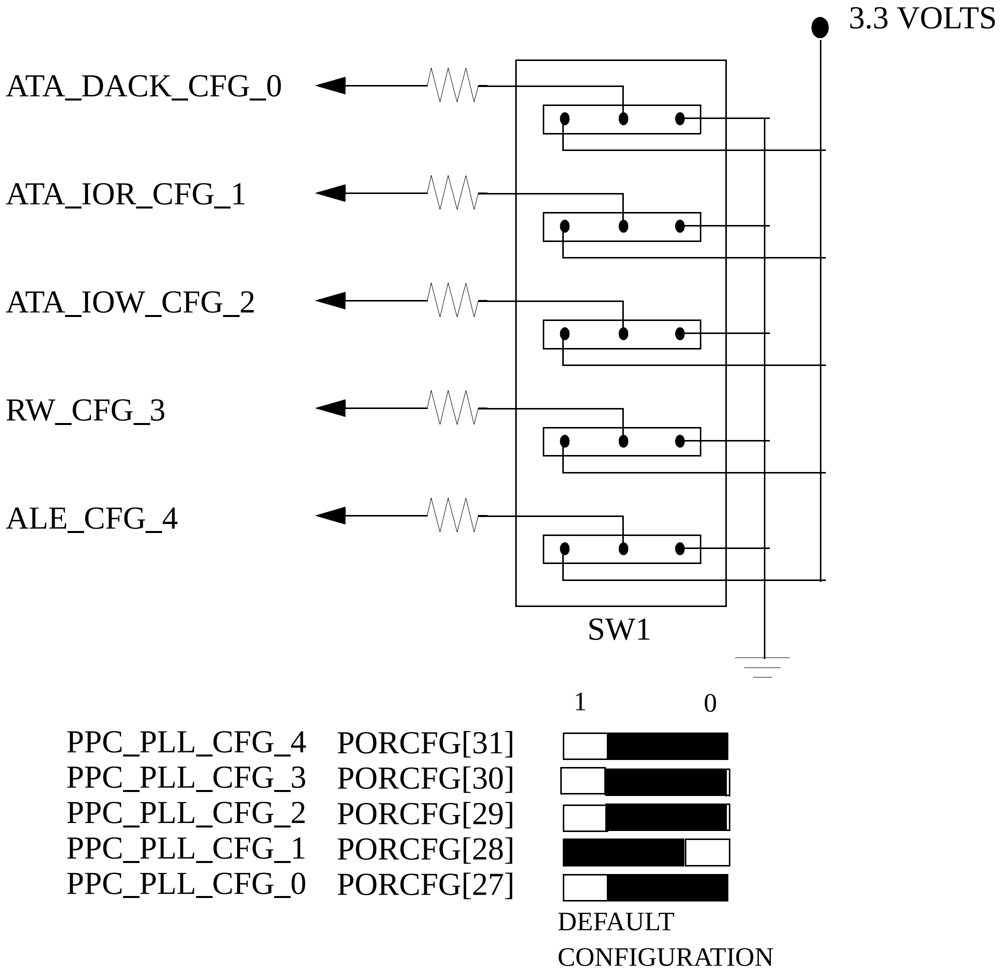


Figure 4-2. MAIN Board Switch SW1 Configuration Settings

Table 4-1. MAIN Board Configuration Switch Settings

Package Ball #	Signal Name	CDM Reset Config. Bit	MPC5200B Reset Config. Reg. BIT	Description
Y18	ATA_DACK	PORCFG[31]	PPC_PLL_CFG_4	MPC5200B's PPC Core PLL Configuration
Y17	ATA_IOR	PORCFG[30]	PPC_PLL_CFG_3	
W17	ATA_IOW	PORCFG[29]	PPC_PLL_CFG_2	
W16	LP_R/W	PORCFG[28]	PPC_PLL_CFG_1	
V14	LP_ALE	PORCFG[27]	PPC_PLL_CFG_0	

Y13	LP_TS	PORCFG[26]	xlb_clk_sel	bit = 0: XLB_CLK = SYS_PLL FVCO/4
				bit = 1: XLB_CLK = SYS_PLL_FVCO/8

H02	USB1	PORCFG[25]	sys_pll_cfg_0	bit =0 : SYS_PLL FVCO = 16x SYS_PLL_FREF
				bit =1 : SYS_PLL FVCO = 12x SYS_PLL_FREF

H03	USB2	PORCFG[24]	2x_FVCO	bit = 0: Fvco = 12x or 16x sys_xtal_in (default)
				bit = 1: Fvco = 24x or 32x sys_xtal_in

K01	ETH0	PORCFG[23]	most_graphics_sel	bit = 0: Most Graphics boot not enabled bit = 1: Most Graphics boot enabled.
K02	ETH1	PORCFG[16]	large_flash_sel	bit = 0: Large Flash boot not enabled bit = 1: Large Flash boot enabled.

K03	ETH2	PORCFG[21]	ppc_msrip	PPC Boot Address / Exception Table Loc.
				bit = 0: 0000 0100 (hex)
				bit = 1: fff0 0100 (hex)

J01	ETH3	PORCFG[20]	boot_rom_wait	bit = 0: 4 IPbus clocks of waitstate* bit = 1: 48 IPbus clocks of waitstate*
J02	ETH4	PORCFG[19]	boot_rom_swap	bit = 0: no byte lane swap - same endian ROM image bit = 1: byte lane swap - different endian ROM image (This option is typically not used because MPC5200 can boot from either endian)

L03	ETH5	PORCFG[18]	boot_rom_size	For “non-muxed” boot ROMs
				bit = 0: 8-bit boot ROM data bus 24-bit boot ROM address
				bit = 1: 16-bit boot ROM data bus 16-bit boot ROM address
				For “muxed” boot ROMs boot ROM addr is max 25 significant bits during address tenure.
				bit = 0: 16-bit ROM data bus
				bit = 1: 32-bit ROM data bus
				For “large flash” boot case boot Flash addr is 25 bits.
				bit = 0: 8-bit Flash data bus
				bit = 1: 16-bit Flash data bus

N02	ETH6	PORCFG[17]	boot_rom_type	bit = 0: non-muxed boot ROM bus, single tenure transfer. bit = 1: muxed boot ROM bus, PPC like with address & data tenures, ALE_b & TS_b active.

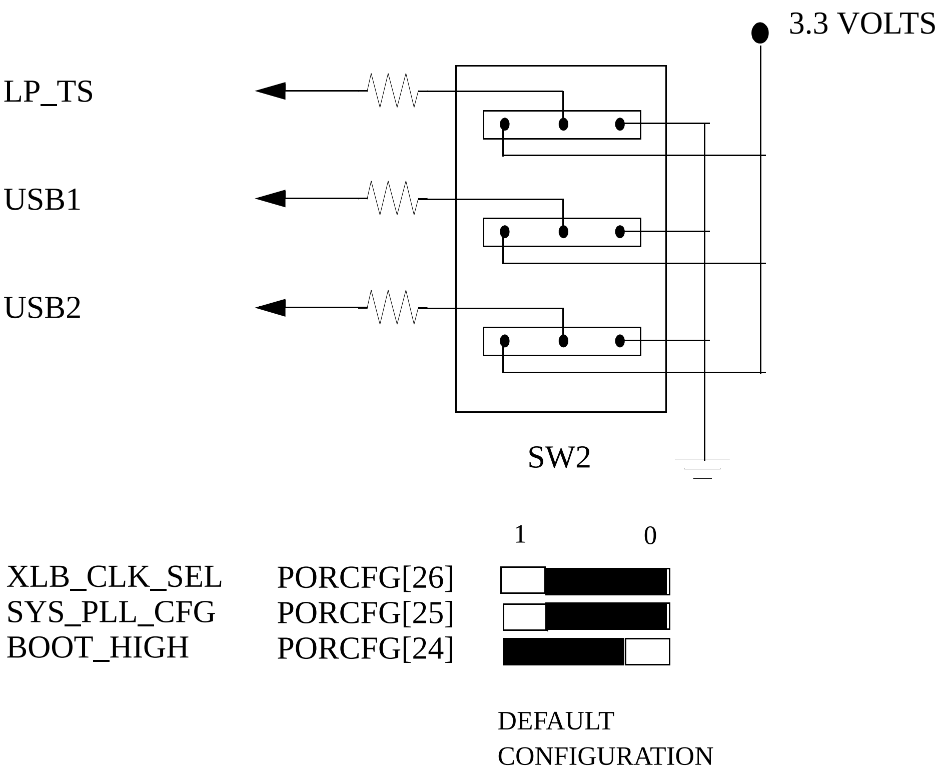


Figure 4-3. MAIN Board Switch SW2 Configuration Settings

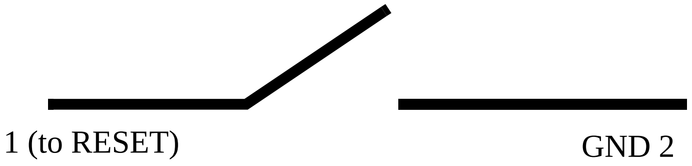


Figure 4-4. MAIN Board Switch SW3 Manual Reset Switch

4.2.2 MAIN Board Jumpers

Table 4-2. MAIN Board Jumpers J1 – J9

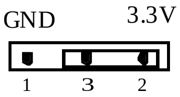
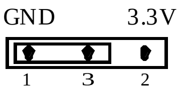
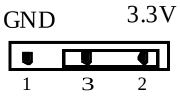
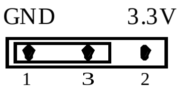
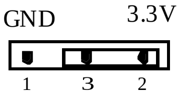
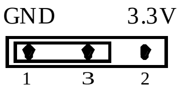
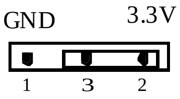
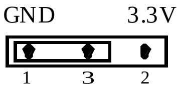
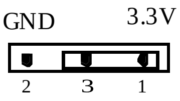
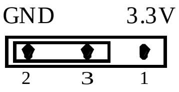
Jumper	Function	Settings	Switch Position	Description
J1	LARGE FLASH	2 - 3 (3.3 V)		Large Flash Boot enabled
		1 - 3 (GND)		Large Flash Boot not enabled (default setting)
J2	2xFVCO	2 - 3 (3.3 V)		Fvco = 24x or 32x sys_xtal_in
		1 - 3 (GND)		Fvco = 12x or 16x sys_xtal_in (default setting)
J3	MOST GRAPHICS	2 - 3 (3.3 V)		Most Graphics Boot enabled
		1 - 3 (GND)		Most Graphics Boot not enabled (default setting)
J4	MANY WAITSTATES	2 - 3 (3.3 V)		48 IP BUS clock periods of wait states
		1 - 3 (GND)		4 IP BUS clock periods of wait states (default setting)

Table 4-2. MAIN Board Jumpers J1 – J9 (continued)

Jumper	Function	Settings	Switch Position	Description

J5	WIDE BOOT DATA LANE	1 - 3 (3.3 V)		For “non-muxed” boot ROMs
				bit = 0: 8-bit boot ROM data bus 24-bit boot ROM address
				bit = 1: 16-bit boot ROM data bus 16-bit boot ROM address
				For “muxed” boot ROMs boot ROM addr is max 25 significant bits during address tenure.
				bit = 0: 16-bit ROM data bus
				bit = 1: 32-bit ROM data bus
				For “large flash” boot case boot Flash addr is 25 bits.
				bit = 0: 8-bit Flash data bus
				bit = 1: 16-bit Flash data bus
				(default setting)

		2 - 3 (GND)		For “non-muxed” boot ROMs
				bit = 0: 8-bit boot ROM data bus 24-bit boot ROM address
				bit = 1: 16-bit boot ROM data bus 16-bit boot ROM address
				For “muxed” boot ROMs boot ROM addr is max 25 significant bits during address tenure.
				bit = 0: 16-bit ROM data bus
				bit = 1: 32-bit ROM data bus
				For “large flash” boot case boot Flash addr is 25 bits.
				bit = 0: 8-bit Flash data bus
				bit = 1: 16-bit Flash data bus

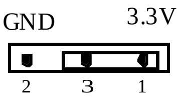
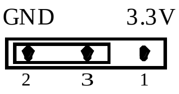
J6	MUXED BOOT	1 - 3 (3.3 V)		Muxed Boot ROM Bus (PPC like with address & data tenures, ALE_b & TS_b active.) (default setting)
		2 - 3 (GND)		Non-muxed boot ROM bus, single tenure transfer

Table 4-2. MAIN Board Jumpers J1 – J9 (continued)

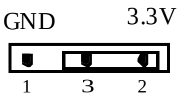
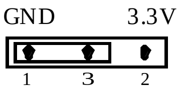
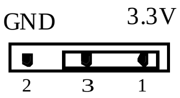
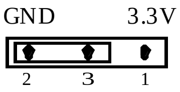
Jumper	Function	Settings	Switch Position	Description
J7	BYTE LANE SWAP	2 - 3 (3.3 V)		byte lane swap—different endian ROM image
		1 - 3 (GND)		No byte lane swap (default setting)
J8	BOOT HIGH	2 - 3 (3.3 V)		
		1 - 3 (GND)		
J9	BOOT HIGH / LOW	1 - 3 (3.3 V)		PPC Boot Address / Exception Table Location 0x 0000 0100
		2 - 3 (GND)		PPC Boot Address / Exception Table Location 0x FFF0 0100

Table 4-3. MAIN Board Jumpers J10 – J76

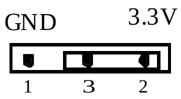
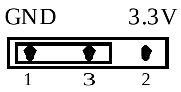
Jumper	Function	Switch Positions	Description
J10	JTAG_TRST_B		Pull JTAG Reset pin high (enable)
			Pull JTAG Reset pin low (disable) (Default)

Table 4-3. MAIN Board Jumpers J10 – J76 (continued)

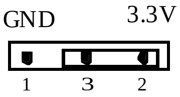
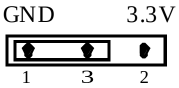
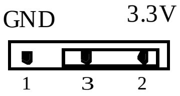
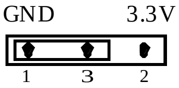
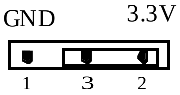
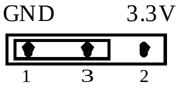
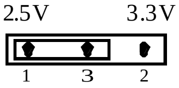
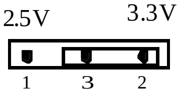
Jumper	Function	Switch Positions	Description
J11	TEST_MODE_0		
			(Default)
J12	TEST_MODE_1		
			(Default)
J13	TEST_SEL_1		
			(Default)
J14	FPGA Pull Up Power Supply Select		(Default)
			

Table 4-3. MAIN Board Jumpers J10 – J76 (continued)

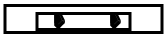
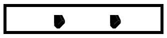
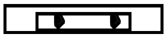
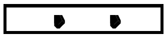
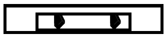
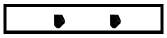
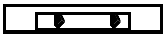
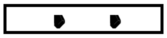
Jumper	Function	Switch Positions	Description
J15	FPGA Connection		Default - Pins connected with zero ohm resistor
			No connection between pins
J16	FPGA Connection		Pins connected with zero ohm resistor
			Default - No connection between pins
J17	FPGA Connection		Default - Pins connected with zero ohm resistor
			No connection between pins
J18	FPGA Connection		Pins connected with zero ohm resistor
			Default - No connection between pins

Table 4-3. MAIN Board Jumpers J10 – J76 (continued)

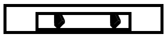
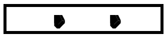
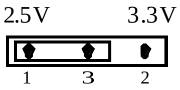
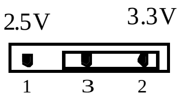
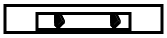
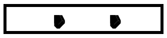
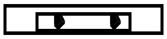
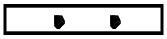
Jumper	Function	Switch Positions	Description
J19	FPGA Connection		Pins connected with zero ohm resistor
			Default - No connection between pins
J20	JTAG Power Supply Select		(Default)
			
J21	JTAG Connector		Default - Pins connected with zero ohm resistor
			No connection between pins
J22	JTAG Connector		Default - Pins connected with zero ohm resistor
			No connection between pins

Table 4-3. MAIN Board Jumpers J10 – J76 (continued)

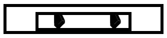
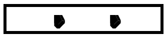
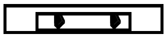
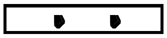
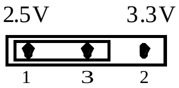
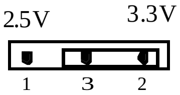
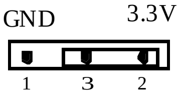
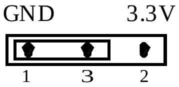
Jumper	Function	Switch Positions	Description
J23	JTAG Connector		Default - Pins connected with zero ohm resistor
			No connection between pins
J24	JTAG Connector		Default - Pins connected with zero ohm resistor
			No connection between pins
J25	HSWAP ENABLE		(Default)
			
J26	M66EN		
			(Default)

Table 4-3. MAIN Board Jumpers J10 – J76 (continued)

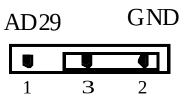
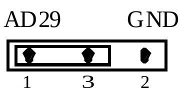
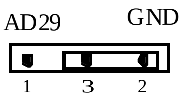
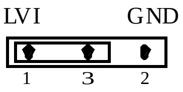
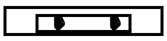
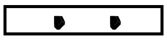
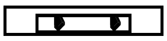
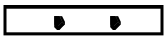
Jumper	Function	Switch Positions	Description
J27	PCI_IDSEL		
			Default
J28	Power On Reset		Power On Reset pin is pulled up by 100 KΩ resistor.
			Power On Reset pin is controlled by U7 - MAX6714DUB Reset Supervisor Device. (Default Setting)
J29	RESET INPUT for MAX6714 DUB		
			(Default Setting—no jumper)
J30			CORALP device held in RESET condition.
			CORALP device controlled by Power On Reset (Default Setting—no jumper)

Table 4-3. MAIN Board Jumpers J10 – J76 (continued)

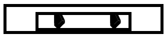
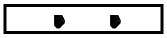
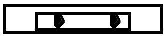
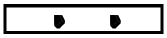
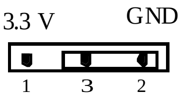
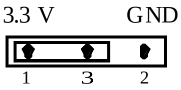
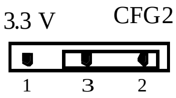
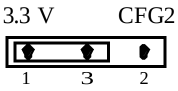
Jumper	Function	Switch Positions	Description
J31	INH pin on CAN Driver of CAN1 (U15)		Enable CAN Physical Interface Device (U15) Default Setting
			CAN Physical Interface Device (U15) not enabled. (Zero ohm resistor removed.)
J32	INH pin on CAN Driver of CAN2 (U16)		Enable CAN Physical Interface Device (U16) Default Setting
			CAN Physical Interface Device (U16) not enabled. (Zero ohm resistor removed.)
J33	USB MODE Select (U18)		
			Default Setting
J34	CFG2		Pin 3 — LED_YEL_A (J34 and J40 control CFG2)
			Pin 3 — LED_YEL_A (J34 and J40 control CFG2) Default Setting

Table 4-3. MAIN Board Jumpers J10 – J76 (continued)

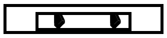
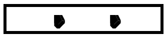
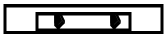
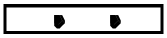
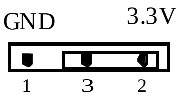
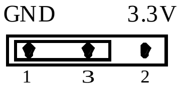
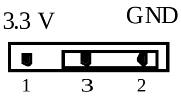
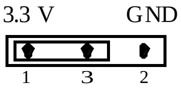
Jumper	Function	Switch Positions	Description
J35	ETHERNET DEVICE (U19) SLEEP EN		
			Default setting
J36	ETHERNET DEVICE (U19) POWERDOWN		
			Default setting
J37	MDDIS SETTING		
			Default Setting
J38			
			TXSLEW1 Default setting

Table 4-3. MAIN Board Jumpers J10 – J76 (continued)

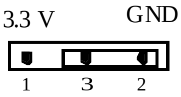
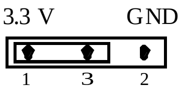
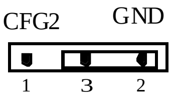
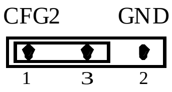
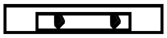
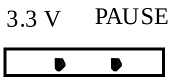
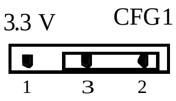
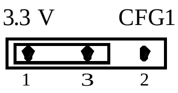
Jumper	Function	Switch Positions	Description
J39			
			TXSLEW0 Default Setting
J40	CFG2		Pin 3 — LED_YEL_C (J34 and J40 control CFG2)
	CFG2		Pin 3 — LED_YEL_C (J34 and J40 control CFG2) Default Setting
J41	PAUSE		
			Default Setting
J42	CFG1		Pin 3 — LD5 GREEN (J42 and J44 control CFG1)
			Pin 3 — LD5 GREEN (J42 and J44 control CFG1) Default Setting

Table 4-3. MAIN Board Jumpers J10 – J76 (continued)

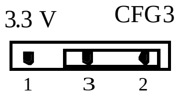
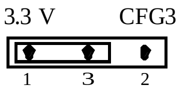
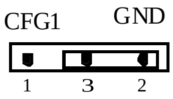
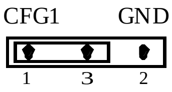
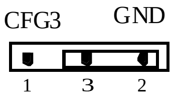
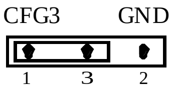
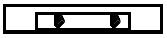
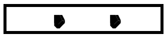
Jumper	Function	Switch Positions	Description
J43	CFG3		Pin 3 — LED_GRN_A (J43 and J45 control CFG3)
			Pin 3 — LED_GRN_A (J43 and J45 control CFG3) Default Setting
J44			(J42 and J44 control CFG1) Pin 3 — LD5 GREEN
			(J42 and J44 control CFG1) Pin 3 — LD5 GREEN Default Setting
J45			Pin 3 — LED_GRN_C (J43 and J45 control CFG3)
			Pin 3 — LED_GRN_C (J43 and J45 control CFG3) Default Setting
J46	Vref for Touch Screen Encoder (U23)		Power connected to Vref on U23 Default Setting
			Vref on U23 not powered

Table 4-3. MAIN Board Jumpers J10 – J76 (continued)

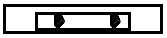
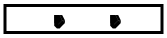
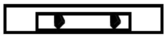
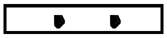
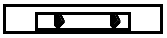
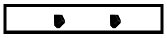
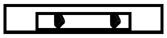
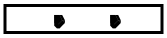
Jumper	Function	Switch Positions	Description
J47	Power connection for Touch Screen Decoder		Connect power to DZ2 Default Setting
			Power not connected to DZ2
J48	Voltage reference for Touch Screen Decoder pin IN4		Connect regulated power from DZ2 to IN4 on Touch Screen Decoder (U23) Default Setting
			Regulated power from DZ2 not connected to IN4 on Touch Screen Decoder (U23)
J49	FPGA_STAT0		Connect FPGA_STAT0 pin to MINI PCI Connector CN9
			Do not connect FPGA_STAT0 pin to MINI PCI Connector CN9 Default Setting
J50	FPGA_STAT1		Connect FPGA_STAT1 pin to MINI PCI Connector CN9
			Do not connect FPGA_STAT1 pin to MINI PCI Connector CN9 Default Setting

Table 4-3. MAIN Board Jumpers J10 – J76 (continued)

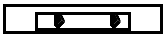
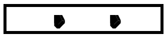
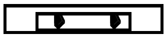
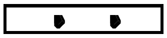
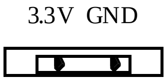
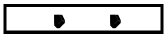
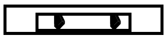
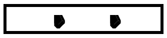
Jumper	Function	Switch Positions	Description
J51	FPGA_STAT2		Connect FPGA_STAT2 pin to MINI PCI Connector CN9
			Do not connect FPGA_STAT2 pin to MINI PCI Connector CN9 Default Setting
J52	FPGA_STAT3		Connect FPGA_STAT3 pin to MINI PCI Connector CN9
			Do not connect FPGA_STAT3 pin to MINI PCI Connector CN9 Default Setting
J53	PCI Connector CLKRUN pin		Default setting
			
J54	FPGA_STAT4		Connect FPGA_STAT4 pin to MINI PCI Connector CN9
			Do not connect FPGA_STAT4 pin to MINI PCI Connector CN9 Default Setting

Table 4-3. MAIN Board Jumpers J10 – J76 (continued)

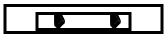
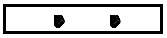
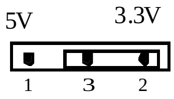
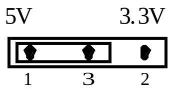
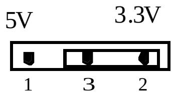
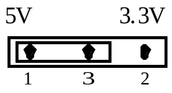
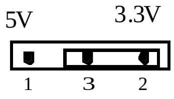
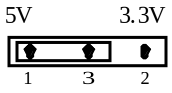
Jumper	Function	Switch Positions	Description
J55	FPGA_STAT5		Connect FPGA_STAT5 pin to MINI PCI Connector CN9
			Do not connect FPGA_STAT5 pin to MINI PCI Connector CN9 Default Setting
J56	ATA Power Select		ATA Power Supply = 3.3V (Pin 3 is connected to ATA Power Supply)
			ATA Power Supply = 5V (Pin 3 is connected to ATA Power Supply)
J57	AVF4910A DVSUP pin Voltage Supply Select		DVSUP pin voltage = 3.3 Volts
			ADVSUP pin voltage = 5 Volts Default Setting
J58			1 - Vref_i of U29 2 - AVF_sync_AVSync 3 - D_CTR_I/O_0 of U29
			1 - Vref_i of U29 2 - AVF_sync_AVSync 3 - D_CTR_I/O_0 of U29 Default Setting

Table 4-3. MAIN Board Jumpers J10 – J76 (continued)

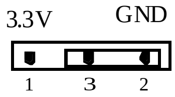
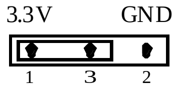
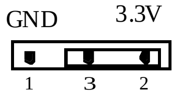
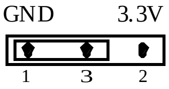
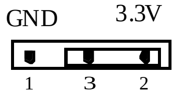
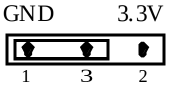
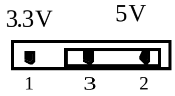
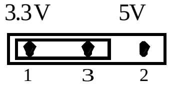
Jumper	Function	Switch Positions	Description
J59	CORAL P CLK SEL 0		CORAL P Clock Select CLK SEL0
			CORAL P Clock Select CLK SEL0 default setting
J60	CORAL P CLK SEL 1		CORAL P Clock Select CLK SEL1
			CORAL P Clock Select CLK SEL1 default setting
J61	CORAL P CLK SEL 2		CORAL P Clock Select CLK MODE
			CORAL P Clock Select CLK MODE default setting
J62	14.31818 MHz Clock Device (U33) Power Supply Select		14.31818 MHz Clock Device (U33) Power Supply = 5 Volts
			14.31818 MHz Clock Device (U33) Power Supply = 3.3 Volts Default Setting

Table 4-3. MAIN Board Jumpers J10 – J76 (continued)

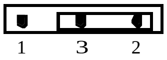
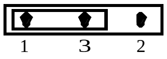
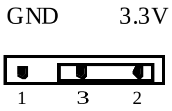
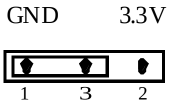
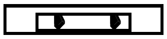
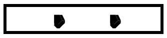
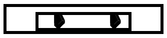
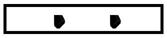
Jumper	Function	Switch Positions	Description
J63	GPU_cntrl_CLOCK inverter select		14.31818 Clock Driver U33 pin 3 is inverted and then drives GPU_cntrl_ CLOCK pin 1 - U33 pin 3 OUT pin 2 - U32 inverter output pin 3 - GPU_cntrl_ CLOCK
			14.31818 Clock Driver U33 pin 3 drives GPU_cntrl_ CLOCK Pin 1 - U33 Pin 3 OUT Pin 2 - U32 inverter output Pin 3 - GPU_cntrl_ CLOCK Default Setting
J64	PD input of SIL 164 (U37)		PD input of SIL 164 (U37) is supplied by the 3.3 V DVI_D supply.
			PD input of SIL 164 (U37) is connected to GND Default Setting
J65			B and W pins of AD5246 (U38) are shorted together.
			B and W pins of AD5246 (U38) are not shorted together. Default Setting
J66	DPS pin on LVDS Connector CN15 voltage select		DPS input on Connector CN15 is connected to 3.3 volts.
			DPS input on Connector CN15 is left floating Default Setting

Table 4-3. MAIN Board Jumpers J10 – J76 (continued)

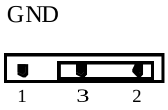
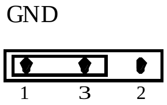
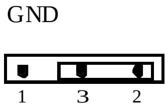
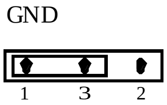
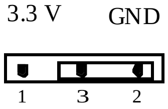
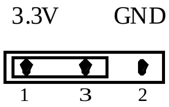
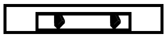
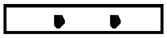
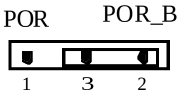
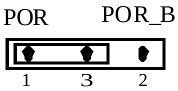
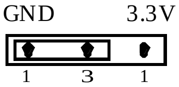
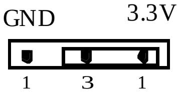
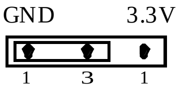
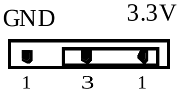
Jumper	Function	Switch Positions	Description
J67			Pin 1 - gnd Pin 2 - RGB_aV_VSYNC Pin 3 - TxIN25 pin of U39
			Pin 1 - gnd Pin 2 - RGB_sV_VSYNC Pin 3 - TxIN25 pin of U39 Default Setting
J68			Pin 1 - GND Pin 2 - RGB_sV_HSYNC Pin 3 - TxIN24 pin of U39
			Pin 1 - GND Pin 2 - RGB_sV_HSYNC Pin 3 - TxIN24 pin of U39 Default Setting
J69	R_FB pin of DS90C383 (U39) voltage select		Pin 3 - R_FB pin (U39) - GND
			Pin 3 - R_FB pin (U39) - 3.3 V Default Setting
J70	USB1T11ABQX (U18) PAD pin voltage select		PAD pin of U18 - GND
			PAD pin of U18 - no voltage applied Default Setting
J71	NOT USED		NOT USED

Table 4-3. MAIN Board Jumpers J10 – J76 (continued)

Jumper	Function	Switch Positions	Description
J72	NOT USED		NOT USED
J73	NOT USED		NOT USED
J74	DUART RESET PIN control		DUART RESET pin controlled by the inversion of Power On Reset
			DUART RESET pin controlled by Power On Reset (Default Setting)
J75	NC1 (no connect) pin of Dual UART		Ground pin 12 (NC1) of Dual Uart (U14) Reserved for future use - do not populate.
			Connect pin 12 (NC1) of Dual UART (U14) to 3.3 Volts. Reserved for future use - do not populate.
J76	NC2 (no connect) pin of Dual UART		Ground pin 24(NC1) of Dual UART (U14) Reserved for future use - do not populate.
			Connect pin 24 (NC1) of Dual UART (U14) to 3.3 Volts. Reserved for future use - do not populate.

4.2.3 MAIN Board Connectors

Table 4-4. MAIN Board Connector CN1 – COP

Pin Name	Pin	Pin	Pin Name
JTAG_TDO	1	2	N.C.
JTAG_TDI	3	4	JTAG_TRST
N.C.	5	6	3.3 VOLTS
JTAG_TCK	7	8	N.C.
JTAG_TMS	9	10	N.C.
SRESET	11	12	GND
HRESET	13	14	N.C.
TEST_SEL_0	15	16	GND

Table 4-5. MAIN Board Connector CN2– CAN1

Pin Name	Pin	Pin	Pin Name
N.C.	1		
CANL	2	6	N.C.
GND	3	7	CANH
N.C.	4	8	N.C.
AC GND	5	9	N.C.
GND	10		
GND	11		

Table 4-6. MAIN Board Connector CN3 – CAN2

Pin Name	Pin	Pin	Pin Name
N.C.	1		
CANL	2	6	N.C.
GND	3	7	CANH
N.C.	4	8	N.C.
ANALOG GROUND	5	9	N.C.
GND	10		
GND	11		

Table 4-7. MAIN Board Connector CN4 – USB

L	SHIELD 1
1	VBus
2	D-
3	D+
4	GND
R	SHIELD 2

Table 4-8. MAIN Board Connector CN5 SPDIF IN

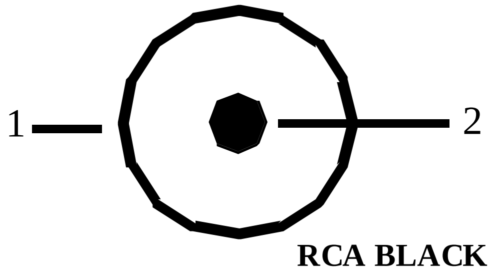
Pin	Function
1	GND
2	SPDIF_DA_INPUT
	

Table 4-9. MAIN Board Connector CN6 SPDIF OUT

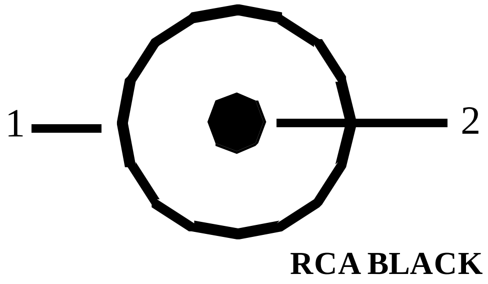
Pin	Function
1	GND
2	SPDIF_DA_INPUT
	

Table 4-10. MAIN Board Connector CN7 PCI 1

PIN NAME	PIN	PIN	PIN NAME
-12V	B1	A1	TRST
TCK	B2	A2	+12
GND0	B3	A3	TMS
TDO	B4	A4	TDI
+5V_1	B5	A5	+5

Table 4-10. MAIN Board Connector CN7 PCI 1 (continued)

PIN NAME	PIN	PIN	PIN NAME
+5V_2	B6	A6	INTA
INTB	B7	A7	INTC
INTD	B8	A8	+5V_5
PRSNT1	B9	A9	RESERVED3
RESERVED1	B10	A10	+3.3V (I/O)
PRSNT2	B11	A11	RESERVED 4
	B12	A12	
	B13	A13	
RESERVED 2	B14	A14	+3.3V (AUX)
GND1	B15	A15	RST
CLK	B16	A16	+3.3V (I/O)_3
GND2	B17	A17	GNT
REQ	B18	A18	GND9
3.3V (I/O) 1	B19	A19	PME
AD31	B20	A20	AD30
AD29	B21	A21	+3.3V (I/O)_7
GND19	B22	A22	AD28
AD27	B23	A23	AD26
AD25	B24	A24	GND10
3.3V_1	B25	A25	AD24
C/$\overline{\text{BE3}}$	B26	A26	IDSEL
AD23	B27	A27	+3.3V (I/O)_8
GND20	B28	A28	AD22
AD21	B29	A29	AD20
AD19	B30	A30	GND11
3.3V_2	B31	A31	AD18
AD17	B32	A32	AD16
C/$\overline{\text{BE2}}$	B33	A33	+3.3V (I/O) 9
GND3	B34	A34	FRAME
IRDY	B35	A35	GND12
3.3V_3	B36	A36	TRDY
DEVSEL	B37	A37	GND13
GND4	B38	A38	STOP

Table 4-10. MAIN Board Connector CN7 PCI 1 (continued)

PIN NAME	PIN	PIN	PIN NAME
LOCK	B39	A39	+3.3V (I/O) 10
PERR	B40	A40	RESERVED 5
3.3V_4	B41	A41	RESERVED 6
SERR	B42	A42	GND14
3.3V_5	B43	A43	PAR
C/$\overline{\text{BE1}}$	B44	A44	AD15
AD14	B45	A45	+3.3V (I/O) 11
GND5	B46	A46	AD13
AD12	B47	A47	AD11
AD10	B48	A48	GND15
M66EN	B49	A49	AD09
GND6	B50	A50	GND16
GND7	B51	A51	GND17
AD08	B52	A52	C/$\overline{\text{BE0}}$
AD07	B53	BA53	+3.3V (I/O)_12
3.3V_6	B54	A54	AD06
AD05	B55	A55	AD04
AD03	B56	A56	GND18
GND8	B57	A57	AD02
AD01	B58	A58	AD00
3.3V_(I/O) 2	B59	A59	+3.3V (I/O) 4
ACK64	B60	A60	REQ64
+5V_3	B61	A61	+5V_6
+5V_4	B62	A62	+5V_7

Table 4-11. MAIN Board Connector C87 PCI 2

PIN NAME	PIN
-12V	B1
TCK	B2
GND0	B3
TDO	B4
+5V_1	B5
+5V_2	B6
INTB	B7
INTD	B8
PRSNT1	B9
RESERVED1	B10
PRSNT2	B11
	B12
	B13
RESERVED 2	B14
GND1	B15
CLK	B16
GND2	B17
REQ	B18
3.3V (I/O) 1	B19
AD31	B20
AD29	B21
GND19	B22
AD27	B23
AD25	B24
3.3V_1	B25
C/$\overline{\text{BE3}}$	B26
AD23	B27
GND20	B28
AD21	B29
AD19	B30
3.3V_2	B31
AD17	B32

PIN	PIN NAME
A1	TRST
A2	+12
A3	TMS
A4	TDI
A5	+5
A6	INTA
A7	INTC
A8	+5V_5
A9	RESERVED3
A10	+3.3V (I/O)
A11	RESERVED 4
A12	
A13	
A14	+3.3V (AUX)
A15	RST
A16	+3.3V (I/O)_3
A17	GNT
A18	GND9
A19	PME
A20	AD30
A21	+3.3V (I/O)_7
A22	AD28
A23	AD26
A24	GND10
A25	AD24
A26	IDSEL
A27	+3.3V (I/O)_8
A28	AD22
A29	AD20
A30	GND11
A31	AD18
A32	AD16

Table 4-11. MAIN Board Connector C87 PCI 2 (continued)

PIN NAME	PIN	PIN	PIN NAME
C/$\overline{BE2}$	B33	A33	+3.3V (I/O) 9
GND3	B34	A34	FRAME
IRDY	B35	A35	GND12
3.3V_3	B36	A36	TRDY
DEVSEL	B37	A37	GND13
GND4	B38	A38	STOP
LOCK	B39	A39	+3.3V (I/O) 10
PERR	B40	A40	RESERVED 5
3.3V_4	B41	A41	RESERVED 6
SERR	B42	A42	GND14
3.3V_5	B43	A43	PAR
C/$\overline{BE1}$	B44	A44	AD15
AD14	B45	A45	+3.3V (I/O) 11
GND5	B46	A46	AD13
AD12	B47	A47	AD11
AD10	B48	A48	GND15
M66EN	B49	A49	AD09
GND6	B50	A50	GND16
GND7	B51	A51	GND17
AD08	B52	A52	C/$\overline{BE0}$
AD07	B53	BA53	+3.3V (I/O)_12
3.3V_6	B54	A54	AD06
AD05	B55	A55	AD04
AD03	B56	A56	GND18
GND8	B57	A57	AD02
AD01	B58	A58	AD00
3.3V_(I/O) 2	B59	A59	+3.3V (I/O) 4
ACK64	B60	A60	REQ64
+5V_3	B61	A61	+5V_6
+5V_4	B62	A62	+5V_7

Table 4-12. MAIN Board Connector CN9 Mini PCI TYPE III

Pin Name	Pun	Pin	Pin Name
TIP	1	2	RING
8PMJ-3	3	4	8PMJ-1
8PMJ-6	5	6	8PMJ-2
8PMJ-7	7	8	8PMJ-4
8PMJ-8	9	10	8PMJ-5
LED1_GRNP	11	12	LED2_YELP
LED1_GRNN	13	14	LED2_YELN
CJSGMD	15	16	RESERVED 8
INTB	17	18	+5V_2
+3.3V_1	19	20	INTA
RESERVED 1	21	22	RESERVED 5
GND1	23	24	+3.3V (AUX) 1
CLK	25	26	RST
GND2	27	28	+3.3V_5
REQ	29	30	GNT
+3.3V_2	31	32	GND10
AD31	33	34	PME
AD29	35	36	RESERVED
GND3	37	38	AD30
AD27	39	40	+3.3V_6
AD25	41	42	AD28
RESERVED2	43	44	AD26
C/$\overline{BE3}$	45	46	AD24
AD23	47	48	IDSEL
GND4	49	50	GND11
AD21	51	52	AD22
AD19	53	54	AD20
GND5	55	56	PAR
AD17	57	58	AD18
C/$\overline{BE2}$	59	60	AD16
IRDY	61	62	GND12
+3.3V_3	63	64	FRAME
CLKRUN	65	66	TRDY

Table 4-12. MAIN Board Connector CN9 Mini PCI TYPE III (continued)

Pin Name	Pun	Pin	Pin Name
SERR	67	68	STOP
GND6	69	70	+3.3V_7
PERR	71	72	DEVSEL
C/BE1	73	74	GND13
AD14	75	76	AD15
GND7	77	78	AD13
AD12	79	80	AD11
AD10	81	82	AD14
GND8	83	84	AD09
AD08	85	86	C/$\overline{\text{BE0}}$
AD07	87	88	+3.3V_8
+3.3V_4	89	90	AD06
AD05	91	92	AD04
RESERVED	93	94	AD02
AD03	95	96	AD00
+5V_1	97	98	RESERVED_WIP1
AD01	99	100	RESERVED_WIP2
GND9	101	102	GND15
AC_SYNC	103	104	M66EN
AC_SDATA_IN	105	106	AC_SDATA_OUT
AC_BIT_CLK	107	108	AC_CODEC_IDO
AC_CODEC_ID1	109	110	AC_RESET
MOD_AUDIO_MON	111	112	RESERVED7
ADDIO_GND1	113	114	GND16
SYS_AUDIO_OUT	115	116	SYS_AUDIO_IN
SYS_AUDIO_OUT_GND	117	118	SYS_AUDIO_IN GND
AUDIO_GND	119	120	AUDIO_GND2
RESERVED 4	121	122	MPCIACT
VCC5VA	123	124	+3.3V (AUX) 2
GND17	M1	M2	GND18
GND19	M3	M4	GND20

Table 4-13. MAIN Board Connector CN10 ATA

Pin Name	Pin	Pin	Pin Name
RESET	1	2	GND7
DD7	3	4	DD8
DD6	5	6	DD9
DD5	7	8	DD10
DD4	9	10	DD11
DD3	11	12	DD12
DD2	13	14	DD13
DD1	15	16	DD14
DD0	17	18	DD15
GND6	19	20	KEY
DMARQ	21	22	GND5
DIOW/STOP	23	24	GND4
DIOR/HDMARDY/HSTROBE	25	26	GND3
IORDY/DDMARDY/DSTROBE	27	28	CSEL
DMACK	29	30	GND2
INTRQ	31	32	IOCS16
DA1	33	34	PDIAG/CBLID
DA0	35	36	DA2
CS0	37	38	CS1
DASP	39	40	GND1

Table 4-14. MAIN Board Connector CN11 S-VIDEO

Pin Name	Pin
AGND_AVI	1
AGND_AVI	2
VIN1	3
CIN1	4
AGND_AVI	5
AGND_AVI	6
AGND_AVI	7

Table 4-15. MAIN Board Connector CN12 CVBS IN

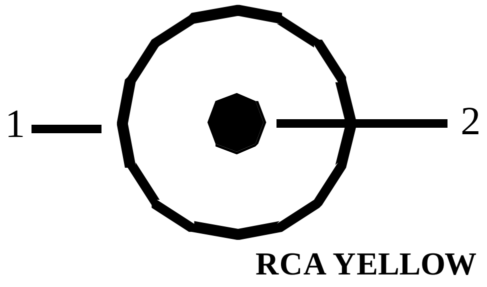
<table>
<tr><th>Pin</th><th>Description</th></tr>
<tr><td>1</td><td>AGND_AVI</td></tr>
<tr><td>2</td><td>VIN2</td></tr>
<tr><td colspan="2"> </td></tr>
</table>

Table 4-16. MAIN Board Connector CN13 DVI - I

Pin		Function		Function		Pin	
C5_1		ANALOG RTN1		ANALOG RTN2		C5_2	
C2		ANALOG GREEN		ANALOG HSYNC		C4	
C1		ANALOG RED		ANALOG BLUE		C3	
Pin	**Function**		**Pin**	**Function**		**Pin**	**Function**
8	ANALOG V-SYNC		16	HP DETECT		24	TXC-
7	DDC DATA		15	GND		23	TXC+
6	DDC CLK		14	+5V		22	TXC SHIELD
5	TX4+		13	TX3+		21	TX5+
4	TX4-		12	TX3-		20	TX5-
3	TX2/4 SHIELD		11	TX1/3 SHIELD		19	TX0/5 SHIELD
2	TX2+		10	TX1+		18	TX0+
1	TX2-		9	TX1-		17	TX0-

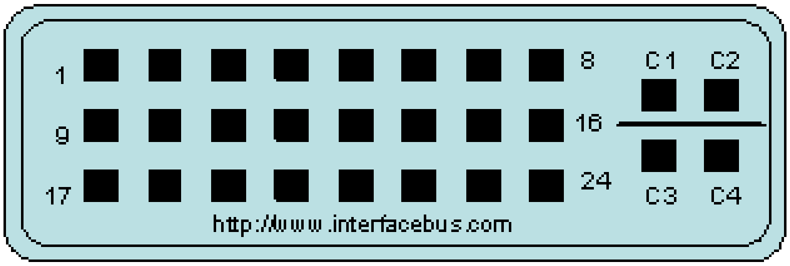


Table 4-17. MAIN Board Connector CN14 Inverter Power

Pin	Pin Name
1	+12 V
2	+12 V
3	GND
4	GND
5	
6	
7	
8	TP 143

Table 4-18. MAIN Board Connector CN15 LVDS

Pin	Pin Name
1	D3+
2	D3-
3	DPS
4	FRC
5	GND1
6	CK+
7	CK-
8	GND2
9	D2+
10	D2-
11	GND3
12	D1+
13	D1-
14	GND4
15	D0+
16	D0-
17	GND5
18	GND6
19	VCC1
20	VCC2

Table 4-19. MAIN Board Connector CN16 Encoder Video Extension

Pin Name	Pin	Pin	Pin Name
ITU_DV_LCC	1		
		2	GND
ITU_DV_VREF	3		
		4	GND
ITU_DV_HREF	5		
		6	GND
ITU_DV_FIELD	7		
		8	GND
ITU_DV_PIXCLK	9		
		10	GND
ITU_DV_VACT	11		
		12	GND
ITU_DV_TDO	13		
		14	GND
ITU_DV_A7	15		
		16	GND
ITU_DV_A6	17		
		18	GND
ITU_DV_A5	19		
		20	GND
ITU_DV_A4	21		
		22	GND
ITU_DV_A3	23		
		24	GND
ITU_DV_A2	25		
		26	1.8 V
ITU_DV_A1	27		
		28	3.3 V
ITU_DV_A0	29		
		30	5 V
POR	31		

Table 4-20. MAIN Board Connector CN17 Encoder Video Extension

Pin Name	Pin	Pin	Pin Name
AVF_sync_AVSync	1		
		2	GND
AVF_dV_CLK20	3		
		4	GND
AVF_dA_CLKOUT	5		
		6	GND
I2SV_bus_CLK	7		
		8	GND
I2SV_bus_WS	9		
		10	GND
I2SV_bus_OUT	11		
		12	GND
VEC_STCLK	13		
		14	GND
VEC_STREQ	15		
		16	GND
VEC_STEN	17		
		18	GND
VEC_TSPSSYNC	19		
		20	GND
VEC_STDATA7	21		
		22	GND
VEC_STDATA6	23		
		24	GND
VEC_STDATA5	25		
		26	GND
VEC_STDATA4	27		
		28	VEC_SCLK
VEC_STDATA3	29		
		30	VEC_SDATAIN
VEC_STDATA2	31		
		32	VEC_SDATAOUT
VEC_STDATA1	33		
		34	FPGA_STAT0

Table 4-21. MAIN Board Connector CN18A – CN18B – CN18C

Function	Pin	Function	Pin	Function	Pin
MOPS_0	1	GND	2	GND	3
MOPS_1	4	GND	5	WAKE_UP_MOST	6
MOPS_2	7	3.3 V	8	GND	9
MOPS_3	10	3.3 V	11	GND	12
MOPS_4	13	GND	14	MOPS_13	15
MOPS_5	16	GND	17	MOPS_14	18
MOPS_6	19	GND	20	MOPS_15	21
MOPS_7	22	GND	23	MOPS_16	24
MOPS_8	25	5 V	26	MOPS_17	27
MOPS_9	28	5 V	29	MOPS_18	30
MOPS_10	31	GND	32	MOPS_19	33
MOPS_11	34	GND	35	MOPS_20	36
MOPS_12	37	GND	38	MOPS_21	39

Table 4-22. MAIN Board Connector CN19 Touch Screen

Pin	Pin Name
1	X+
2	Y+
3	X-
4	Y-
5	GND

Table 4-23. MAIN Board Connector CN20 Button Encoder

Pin	Pin Name
1	3.3 Volts
2	5 Volts
3	GND
4	GND
5	GND
6	GND

4.2.4 MAIN Board Headers

Table 4-24. MAIN Board Header 1 JTAG FPGA

Pin	Pin Name
1	2.5V or 3.3V from J20
2	GND
3	NO CONNECT
4	SI_TCK
5	NO CONNECT
6	SI_TDO
7	SI_TDI
8	NO CONNECT
9	SI_TMS
10	NO CONNECT

Table 4-25. MAIN Board Header 2 GND

Pin	Pin Name
1	GND
2	GND

Table 4-26. MAIN Board Header 3 GND

Pin	Pin Name
1	GND
2	GND

Table 4-27. MAIN Board Header 4 GND

Pin	Pin Name
1	GND
2	GND

Table 4-28. MAIN Board Header 5 GND

Pin	Pin Name
1	GND
2	GND

Table 4-29. MAIN Board Header 6 GND

Pin	Pin Na,e
1	GND
2	GND

Table 4-30. MAIN Board Header 7 GND

Pin	Pin Name
1	GND
2	GND

Table 4-31. MAIN Board Header 9 GPIO

Pin Name	Pin	Pin	Pin Name
3.3 V	1	2	GND
GPIO0	3	4	NO CONNECT
GPIO1	5	6	NO CONNECT
GPIO2	7	8	NO CONNECT
GPIO3	9	10	NO CONNECT
GPIO4	11	12	NO CONNECT
GPIO5	13	14	NO CONNECT
GPIO6	15	16	NO CONNECT
GPIO7	17	18	NO CONNECT
GND	19	20	3.3 V

Table 4-32. MAIN Board Header 10 I2C

Pin	Pin Name
1	SDA
2	SCL
3	3.3V
4	GND

Table 4-33. MAIN Board Header 12 SPI

Pin Name	Pin	Pin	Pin Name
3.3 V	1	2	GND
TS_SPI_CLK	3	4	GPIO2
TS_SPI_MOSI	5	6	TS_SPI_IRQ
TS_SPI_MISO	7	8	TS_SPI_SS
GND	9	10	5 V

Table 4-34. MAIN Board Header 13 GPS SPI/UART

Pin Name	Pin
GPS_TX_SPI_MOSI	1
GPS_TX_SPI_MISO	2
GPS_SPI_CLK	3
GPS_SPI_SS	4
NO CONNECT	5
GND	6

Table 4-35. MAIN Board Header 14 GPS RESET/IRQ

Pin Name	Pin
3.3 V	1
5 V	2
GPS_RESET	3
GPS_IRQ	4
GPS_GPIO	5
GND	6

Table 4-36. MAIN Board Header 15 Board to Board Connector

PIN NAME	PIN	PIN NAME	PIN	PIN NAME	PIN
12 V	A1	GND	B1	12 V	C1
12 V	A2	GND	B2	12 V	C2
- 12 V	A3	GND	B3	- 12 V	C3
5 V	A4	GND	B4	5 V	C4
5 V	A5	GND	B5	5 V	C5
3.3 V	A6	GND	B6	3.3 V	C6
3.3 V	A7	GND	B7	3.3 V	C7
2.5 V	A8	GND	B8	2.5 V	C8
2.5 V	A9	GND	B9	2.5 V	C9
1.25 V	A10	GND	B10	1.25 V	C10

Table 4-37. MAIN Board Heade 16 Board to Board Connector

Pin Name	Pin	Pin Name	Pin	Pin Name	Pin
1.25 V	A1	GND	B1	3.3 V	C1
1.25 V	A2	GND	B2	3.3 V	C2
DSP_MUTE_ND	A3	GND	B3	MUTE_LINE	C3
PSC2_0 AC97_DATA_OUT	A4	GND	B4	PSC1_0 MPC_I2S_DO	C4
PSC2_1AC97_DATA_IN	A5	GND	B5	PSC1_1 MPC_I2S_DI	C5
PSC2_2 AC97_SYNC	A6	GND	B6	PSC1_2 MPC_I2S_MCLK	C6
PSC2_3 AC97_BITCLK	A7	GND	B7	PSC1_3 MPC_I2S_SCLK	C7
PSC2_4AC97_RESET	A8	GND	B8	PSC1_4 MPC_I2S_LRCLK	C8
12 V	A9	GND	B9	NO CONNECT	C9
12 V	A10	GND	B10	NO CONNECT	C10

Table 4-38. MAIN Board Header 17 Board to Board Connector

Pin Name	Pin	Pin Name	Pin	Pin Name	Pin
1.8 V	A1	GND	B1	1.8 V	C1
1.5 V	A2	GND	B2	1.5 V	C2
1.5 V	A3	GND	B3	1.5 V	C3
1.2 V	A4	GND	B4	1.2 V	C4
1.2 V	A5	GND	B5	1.2 V	C5
PSC6_2 TXD	A6	GND	B6	NO CONNECT	C6
PSC6_3 RTS	A7	GND	B7	RS232_FORCEON_EXT1	C7
PSC6_0 RXD	A8	GND	B8	RS232_FORCEOFF_EXT1	C8
PSC6_1 CTS	A9	GND	B9	RS232_FORCEON	C9
NO CONNECT	A10	GND	B10	RS232_FORCEOFF	C10

Table 4-39. MAIN Board Header 18 Board to Board Connector

Pin Name	Pin	Pin Name	Pin	Pin Name	Pin
5 V	A1	GND	B1	SPDIF_DA_OUTPUT	C1
5 V	A2	GND	B2	SPDIF_DA_INPUT	C2
I2C2_CLK	A3	GND	B3	DSP_MODE	C3
I2C2_IO	A4	GND	B4	DSP_RESET	C4
GND	A5	GND	B5	MUTE_SURR	C5
AV_AUDIO_IN_L	A6	GND	B6	AC97 DSP SW0 MIC_INPUT_CROSS_SWITCH	C6
AV_AUDIO_IN_R	A7	GND	B7	AC97 DSP SW1 AC97_DSP_LINE	C7
AGND_AVI	A8	GND	B8	AC97 DSP SW2 AC97_DSP_SURR	C8
-12 V	A9	GND	B9	AC97 DSP SW3 AC97_DSP_CEN_LFE	C9
-12 V	A10	GND	B10	MUTE_CEN_LFE	C10

Table 4-40. MAIN Board Header 19 Board to Board Connector

Pin Name	Pin	Pin Name	Pin	Pin Name	Pin
UART_EXT0_T0	A1	GND	B1	UART_EXT1_T0	C1
UART_EXT0_T1	A2	GND	B2	UART_EXT1_T1	C2
UART_EXT0_T2	A3	GND	B3	UART_EXT1_T2	C3
UART_EXT0_R0	A4	GND	B4	UART_EXT1_R0	C4
UART_EXT0_R1	A5	GND	B5	UART_EXT1_R1	C5
UART_EXT0_R2	A6	GND	B6	UART_EXT1_R2	C6
UART_EXT0_R3	A7	GND	B7	UART_EXT1_R3	C7
UART_EXT0_R4	A8	GND	B8	UART_EXT1_R4	C8
NO CONNECT	A9	GND	B9	NO CONNECT	C9
NO CONNECT	A10	GND	B10	NO CONNECT	C10

4.3 POWER Board Switches, Jumpers, Connectors, and Headers

4.3.1 POWER Board Jumpers

Table 4-41. POWER Board Jumpers J1 - J4

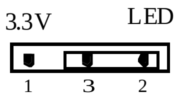
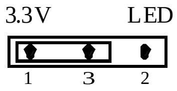
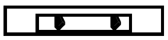
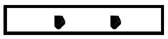
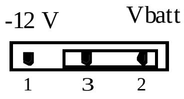
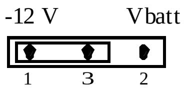
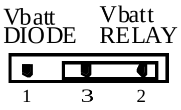
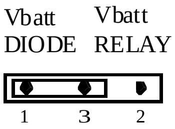
Jumper	Setting	Switch Positions	Description
J1	2 - 3 (LED)		
	1 - 3 (3.3V)		DEFAULT SETTING
J2	2 - 3 (3.3 V)		DEFAULT SETTING ISEN of U2 CONNECTED TO Q8
	1 - 3 (GND)		ISEN of U2 NOT CONNECTED TO Q8
J3	2 - 3 (Vbatt)		U3 POWER SUPPLIED BY Vbatt
	1 - 3 (-12V)		DEFAULT SETTING U3 POWER SUPPLIED BY -12 V SUPPLY

Table 4-41. POWER Board Jumpers J1 - J4 (continued)

Jumper	Setting	Switch Positions	Description
J4	2 - 3		VIN OF U4 SUPPLIED BY V_Batt_DIODE
	1 - 3		DEFAULT SETTING VIN OF U4 SUPPLIED BY V_Batt_RELAY

4.3.2 POWER Board Connectors

Table 4-42. POWER Board Connector CN1 POWER Connector

Pin Name	Pin	Pin	Pin Name
+ 3.3 VDC1	1	11	+ 3.3 VDC3
+ 3.3 VDC2	2	12	-12VDC
COM1	3	13	COM4
+ 5 VDC1	4	14	PS_ON
COM2	5	15	COM5
+ 5 VDC2	6	16	COM6
COM3	7	17	COM7
PWR_OK	8	18	+ 5 VDC
+5VSB	9	19	+ 5 VDC3
+12VDC	10	20	+ 5 VDC4

Table 4-43. POWER Board Connector CN2 – INT UART0

Pin Name	Pin	Pin	Pin Name
TP2	1		
RXD_INT0	2	6	TP3 .
TXD_INT0	3	7	RTS_INT0
TP4	4	8	CTS_INT0.
GND	5	9	TP5
GND	10		
GND	11		

Table 4-44. POWER Board Connector CN3 – EXT UART0

PIN NAME	PIN	PIN	PIN NAME
DCD_EXT0	1		
RXD_EXT0	2	6	DSR_EXT0 .
TXD_EXT0	3	7	RTS_EXT0
DTR_EXT0	4	8	CTS_EXT0
GND	5	9	RI_EXT0
GND	10		
GND	11		

Table 4-45. POWER Board Connector CN4 – EXT UART1

Pin Name	Pin	Pin	Pin Name
DCD_EXT1	1		
RXD_EXT1	2	6	DSR_EXT1 .
TXD_EXT1	3	7	RTS_EXT1
DTR_EXT1	4	8	CTS_EXT1
GND	5	9	RI_EXT1
GND	10		
GND	11		

Table 4-46. POWER Board Connector CN5

Pin Name	Pin	Pin Name	Pin	Pin Name	Pin
12 V	A1	GND	B1	12 V	C1
12 V	A2	GND	B2	12 V	C2
- 12 V	A3	GND	B3	- 12 V	C3
5 V	A4	GND	B4	5 V	C4
5 V	A5	GND	B5	5 V	C5
3.3 V	A6	GND	B6	3.3 V	C6
3.3 V	A7	GND	B7	3.3 V	C7
2.5 V	A8	GND	B8	2.5 V	C8
2.5 V	A9	GND	B9	2.5 V	C9
1.25 V	A10	GND	B10	1.25 V	C10

Table 4-47. POWER Board Connector CN6

Pin Name	Pin	Pin Name	Pin	Pin Name	Pin
1.8 V	A1	GND	B1	1.8 V	C1
1.5 V	A2	GND	B2	1.5 V	C2
1.5 V	A3	GND	B3	1.5 V	C3
1.2 V	A4	GND	B4	1.2 V	C4
1.2 V	A5	GND	B5	1.2 V	C5
UART_INT0_0	A6	GND	B6	NO CONNECT	C6
UART_INT0_1	A7	GND	B7	RS232_FORCEON_EXT1	C7
UART_INT0_2	A8	GND	B8	RS232_FORCEOFF_EXT1	C8
UART_INT0_3	A9	GND	B9	RS232_FORCEON	C9
NO CONNECT	A10	GND	B10	RS232_FORCEOFF	C10

Table 4-48. POWER Board Connector CN7

Pin Name	Pin	Pin Name	Pin	Pin Name	Pin
UART_EXT0_0	A1	GND	B1	UART_EXT1_0	C1
UART_EXT0_1	A2	GND	B2	UART_EXT1_1	C2
UART_EXT0_2	A3	GND	B3	UART_EXT1_2	C3
UART_EXT0_3	A4	GND	B4	UART_EXT1_3	C4
UART_EXT0_4	A5	GND	B5	UART_EXT1_4	C5
UART_EXT0_5	A6	GND	B6	UART_EXT1_5	C6
UART_EXT0_6	A7	GND	B7	UART_EXT1_6	C7
UART_EXT0_7	A8	GND	B8	UART_EXT1_7	C8
NO CONNECT	A9	GND	B9	NO CONNECT	C9
NO CONNECT	A10	GND	B10	NO CONNECT	C10

Table 4-49. POWER Board Connector CN8

Pin Name	Pin
Vbatt	1
GND	2

4.3.3 POWER Board Headers

Table 4-50. POWER Board Header H1 - ATA POWER

Pin	Pin Name	Pin Name	Pin
1	12V	5 V	2
3	GND	GND	4

Table 4-51. POWER Board Header H2

Pin	Pin Name
1	GND
2	GND

Table 4-52. POWER Board Header H3

Pin	Pin Name
1	GND
2	GND

Table 4-53. POWER Board Header H4

Pin	Pin Name
1	GND
2	GND

Table 4-54. POWER Board Header H5

Pin	Pin Name
1	GND
3	GND

Table 4-55. POWER Board Header H6

Pin	Pin Name
1	GND
3	GND

4.4 AUDIO Board Connectors, Jumpers, and Headers

4.4.1 AUDIO Board Jumpers

Table 4-56. AUDIO Board Jumpers J1 - J9

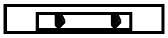
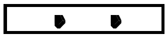
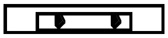
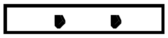
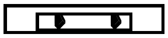
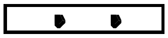
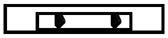
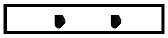
Jumper	Setting	Description
J1		Right channel microphone bypass.
		DEFAULT, not populated.
J2		Left channel microphone bypass.
		DEFAULT, not populated.
J3		Left channel line-in by-pass.
		DEFAULT, not populated.
J4		Right channel line-in bypass.
		DEFAULT, not populated.

Table 4-56. AUDIO Board Jumpers J1 - J9 (continued)

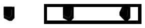
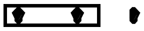
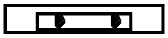
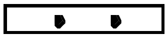
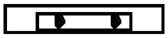
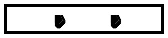
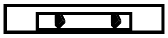
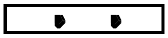
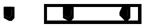
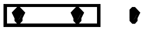
Jumper	Setting	Description
J5		DEFAULT, 1-3. Enables analog voltage regulator.
		2-3. Disables analog voltage regulator.
not used		
not used		
J8		Not populated.
		Not populated.
J9		Not populated.
		Not populated.
J10		Not populated.
		Not populated.
J11		Not populated.
		Not populated.

Table 4-56. AUDIO Board Jumpers J1 - J9 (continued)

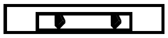
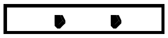
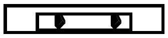
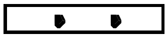
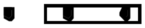
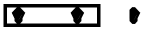
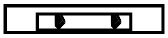
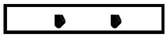
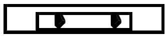
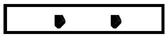
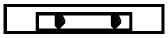
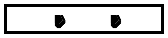
Jumper	Setting	Description
J12		Not populated.
		Not populated.
J13		Not populated.
		Not populated.
J14		DEFAULT, 1-3. Microphone voltage bias connected to tip on CN12.
		2-3. Microphone voltage bias connected to ring on CN12.
J15		Microphone amplifier digital potentiometer feedback resistor bypass.
		DEFAULT, not populated.
J16		Microphone amplifier digital potentiometer feedback resistor bypass.
		DEFAULT, not populated.

Table 4-56. AUDIO Board Jumpers J1 - J9 (continued)

Jumper	Setting	Description
J17		Microphone amplifier digital potentiometer feedback resistor bypass.
		DEFAULT, not populated.

4.4.2 AUDIO Board Connectors

Table 4-57. AUDIO Board Connector CN1 PHONE in

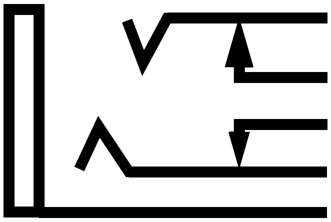
Pin Name	Pin
PHONE	5
JD0/GPIO0	4
NO CONNECT	3
NO CONNECT	2
AGND	1
	

Table 4-58. AUDIO Board Connector CN2 LINE in

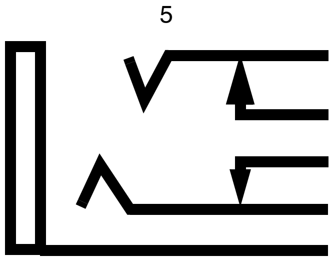
<table>
<tr><th>Pin Name</th><th>Pin</th></tr>
<tr><td>LINE_IN_L</td><td>5</td></tr>
<tr><td>JD1/GPIO1</td><td>4</td></tr>
<tr><td>NO CONNECT</td><td>3</td></tr>
<tr><td>LINE_IN_R</td><td>2</td></tr>
<tr><td>AGND</td><td>1</td></tr>
<tr><td colspan="2"> </td></tr>
</table>

Table 4-59. AUDIO Board Connector CN3 AUX in

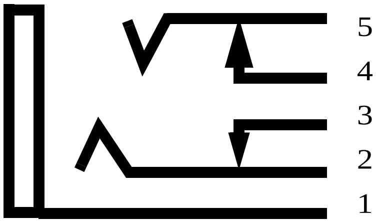
<table>
<tr><th>Pin Name</th><th>Pin</th></tr>
<tr><td>LINE_IN_L</td><td>5</td></tr>
<tr><td>AUX_L</td><td>4</td></tr>
<tr><td>NO CONNECT</td><td>3</td></tr>
<tr><td>AUX_R</td><td>2</td></tr>
<tr><td>AGND</td><td>1</td></tr>
<tr><td colspan="2"> </td></tr>
</table>

Table 4-60. AUDIO Board Connector CN4 CD in

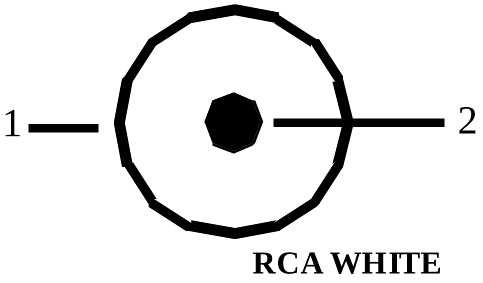
<table>
<tr><th>Pin Name</th><th>Pin</th></tr>
<tr><td>1</td><td>CD_GND FROM U2</td></tr>
<tr><td>2</td><td>CD_L FROM U2</td></tr>
<tr><td colspan="2">
</td></tr>
</table>

Table 4-61. AUDIO Board Connector CN5 CD in

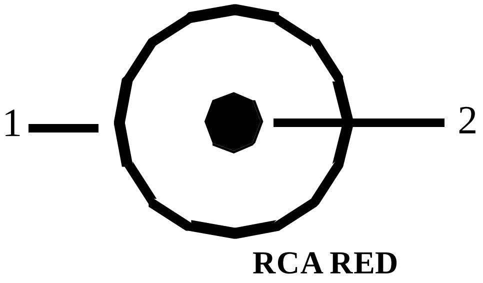
<table>
<tr><th>Pin Name</th><th>Pin</th></tr>
<tr><td>1</td><td>CD_GND FROM U2</td></tr>
<tr><td>2</td><td>CD_R FROM U2</td></tr>
<tr><td colspan="2">
</td></tr>
</table>

Table 4-62. AUDIO Board Connector CN6

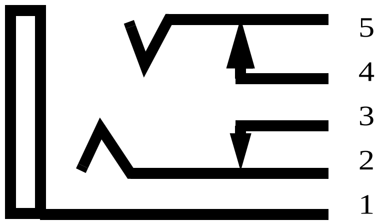
Pin Description	Pin
Left DSP Audio In	5
Analog Ground	4
Analog Ground	3
Right DSP Audio In	2
Filtered Analog Ground	1
	

Table 4-63. AUDIO Board Connector CN7 AV AUDIO INin

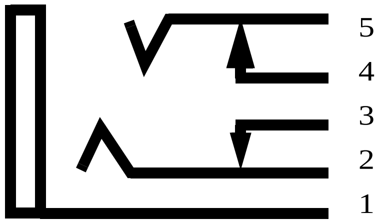
Pin Description	Pin
AV Audio In Left	5
NO CONNECT	4
NO CONNECT	3
AV Audio In Right	2
Ground	1
	

Table 4-64. AUDIO Board Connector CN8

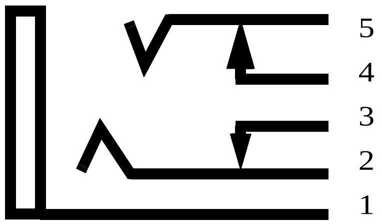
Pin Description	Pin
Headphone Line Out Left	5
NO CONNECT	4
NO CONNECT	3
Headphone Line Out Right	2
Ground	1
	

Table 4-65. AUDIO Board Connector CN9 LINE OUT

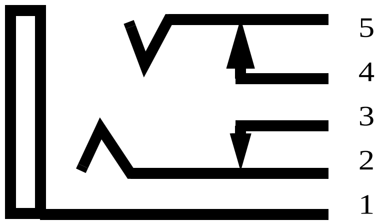
Pin Description	Pin
Line Out Left	5
NO CONNECT	4
NO CONNECT	3
Line Out Right	2
Ground	1
	

Table 4-66. AUDIO Board Connector CN10 SURR OUT

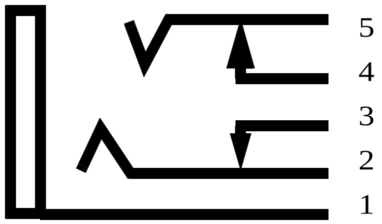
Pin Description	Pin
Surround Out Left	5
NO CONNECT	4
NO CONNECT	3
Surround Out Right	2
Ground	1
	

Table 4-67. AUDIO Board Connector CN11 CEN/LFE OUT

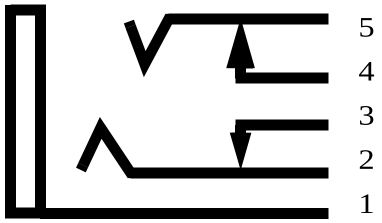
Pin Description	Pin
Center Channel Out	5
NO CONNECT	4
NO CONNECT	3
Subwoofer Out	2
Ground	1
	

Table 4-68. AUDIO Board Connector CN12 MONO MIC

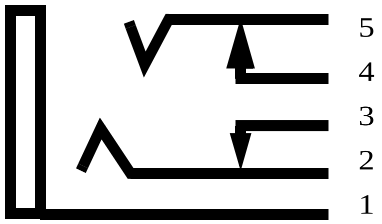
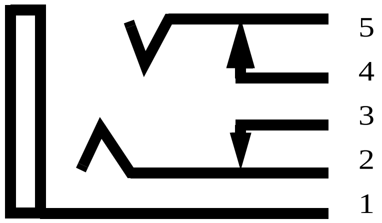
Pin Description	Pin
Signal input/DC Power	5
NO CONNECT	4
NO CONNECT	3
NO CONNECT	2
Ground	1
	

Table 4-69. AUDIO Board Connector CN13 STEREO MIC

Pin Description	Pin
Left microphone input	5
NO CONNECT	4
NO CONNECT	3
Right microphone input	2
Ground	1
	

4.4.3 AUDIO Board Headers

Table 4-70. AUDIO Board Header J6 Board to Board Connector

Pin Name	Pin	Pin Name	Pin	Pin Name	Pin
1.25V	A1	GND	B1	3.3 Volts	C1
1.25V	A2	GND	B2	3.3 Volts	C2
DSP_MUTE	A3	GND	B3	MUTE_aA_LINE	C3
AC97_bus_SDATA_OUT	A4	GND	B4	MPC_I2S_DO	C4
AC97_bus_SDATA_IN	A5	GND	B5	MPC_I2S_DI	C5
AC97_bus_SYNC	A6	GND	B6	MPC_I2S_MCLK	C6
AC97_bus_BIT_CLK	A7	GND	B7	MPC_I2S_I2S_SCLK	C7
AC97_bus_RESET	A8	GND	B8	MPC_I2S_LRCLK	C8
12 V	A9	GND	B9	TP25	C9
12 V	A10	GND	B10	TP26	C10

Table 4-71. AUDIO Board Header J7 Board to Board Connector

Pin Name	Pin	Pin Name	Pin	Pin Name	Pin
5V	A1	GND	B1	SPDIF_dA_Output	C1
5V	A2	GND	B2	SPDIF_dA_Input	C2
I2C2_CLK	A3	GND	B3	DSP_MODE	C3
I2C2_IO	A4	GND	B4	DSP_RESET	C4
GND	A5	GND	B5	MUTE_aA_SURR	C5
AV AUDIO IN L	A6	GND	B6	MIC_cntl_crossSwitch	C6
AV AUDIO IN R	A7	GND	B7	SWITCH_aA_LINEout	C7
GND	A8	GND	B8	SWITCH_aA_SURRout	C8
-12 V	A9	GND	B9	SWITCH_aA_CEN/LFEout	C9
-12 V	A10	GND	B10	MUTE_aA_CEN/LFE	C10

4.4.4 AUDIO Board Headers

Table 4-72. AUDIO Board Header H1

Pin	Pin Name
1	GND
2	GND

Table 4-73. AUDIO Board Header H2

Pin	Pin Name
1	GND
2	GND

Table 4-74. AUDIO Board Header H3

Pin	Pin Name
1	GND
2	GND

Table 4-75. AUDIO Board Header H4

Pin	Pin Name
1	GND
3	GND

Table 4-76. AUDIO Board Header H5

Pin	Pin Name
1	GND
2	GND

Table 4-77. AUDIO Board Header H6

Pin	Pin Name
1	GND
3	GND

4.5 GPS Board Connectors, Jumpers, and Headers

4.5.1 GPS Board Jumpers

Table 4-78. GPS Board Jumpers J1 - J9

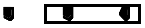
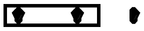
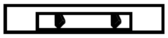
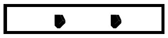
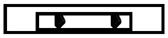
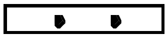
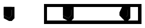
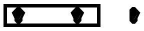
Jumper	Function	Switch Positions	Description
J1	MODE ENABLE		
			DEFAULT
J2	Regulator Bypass		
			DEFAULT
J3	Regulator Bypass		
			DEFAULT
J4	RF Input		V_Ant_Bias
			RF Input DEFAULT

Table 4-78. GPS Board Jumpers J1 - J9 (continued)

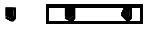
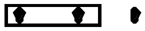
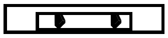
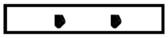
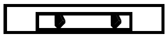
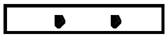
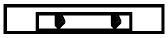
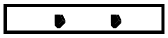
Jumper	Function	Switch Positions	Description
J5			
			
J6	FS Oncore Ground		Ground
			No Ground
J7	FS Oncore Ground		
			
J8	FS Oncore Ground		
			DEFAULT
J9	HOST GPIO		

Table 4-78. GPS Board Jumpers J1 - J9 (continued)

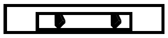
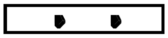
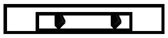
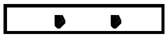
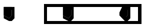
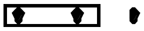
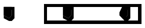
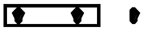
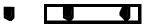
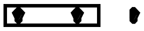
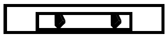
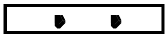
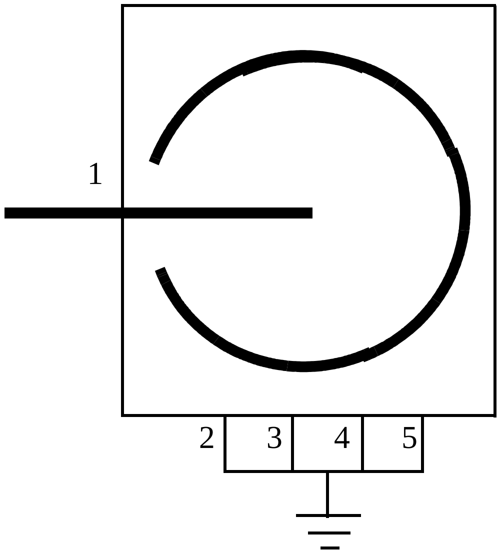
Jumper	Function	Switch Positions	Description
J10	HOST_TX_SPI_ MOSI		Connected to Host DEFAULT
			
J11	HOST_RX_SPI _MISO		Connected to Host DEFAULT
			
J12	FORCE ON		
			DEFAULT
J13	FORCE OFF		
			DEFAULT
J14	ENABLE		
			

Table 4-78. GPS Board Jumpers J1 - J9 (continued)

Jumper	Function	Switch Positions	Description
J15	GPS RESET		RESET Connected
			DEFAULT RESET Not Connected

4.5.2 GPS Board Connectors

Table 4-79. GPS Board Antenna CN1

Pin	Pin Name
1	50 OHM ANTENNA INPUT
2	GND
3	GND
4	GND
5	GND
	

4.5.3 GPS Board Headers

Table 4-80. GPS Board Header H1

Pin	Pin Name
1	V_in
2	V_bias
3	RESET
4	HOST_IRQ
5	HOST_GPIO
6	GND

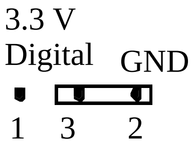
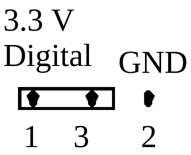
Table 4-81. GPS Board Header H2

Pin	Pin Name
1	HOST_TX_SPI_MOSI
2	HOST_RX_SPI_MISO
3	HOST_SPI_CLK
4	HOST_SPI_SS
5	NO CONNECT
6	GND

4.6 Most Board Connectors, Jumpers, and Headers

4.6.1 MOST Board Jumpers

Table 4-82. MOST Board Jumpers J1 - J9

Jumper	Function	Switch Positions	Description
J1	LEG		
			Default

4.6.2 MOST Board Connectors

Table 4-83. MOST Board Connector CN1A/CN1B/CN1C Board to Board Connector

Pin Name	Pin	Pin Name	Pin	Pin Name	Pin
M_D0	1	GND	2	GND	3
M_D1	4	GND	5	BF_STATUS	6
M_D2	7	3.3 V	8	GND	9
M_D3	10	3.3 V	11	GND	12
M_D4	13	GND	14	M_INT	15
M_D5	16	GND	17	M_AINT	18
M_D6	19	GND	20	M_FSY	21
M_D7	22	GND	23	M_SRC_FL	24
M_PAD0	25	5 V	26	M_ERROR	27
M_PAD1	28	5 V	29	M_STATUS	30
M_RD	31	GND	32	M_CP_FLOW	33
M_WR	34	GND	35	M_RMCK	36
M_RESET	37	GND	38	BF_-3dB	39

Chapter 5 Boot Monitor

The boot monitor provided with the Media5200 is the U-Boot open source project maintained at:

U-Boot Source:

http://sourceforge.net/projects/u-boot

U-Boot Online Documentation:

http://www.denx.de/wiki/DULG/Manual

This boot monitor program will allow the user to download and flash code using Ethernet and UART protocols. Additionally the monitor provides functionality which allows the exercise of:

- EEPROM
- Ethernet
- PCI
- ATA
- USB

For a complete list of see *U-Boot Quick Reference* provided on the MPC5200B Development Kit CD or the U-Boot website.

5.1 U-Boot Recovery

The Media5200 does not support hardware-assisted U-Boot recovery. If U-Boot has been erased, use a hardware debugger to re-flash the U-Boot binary provided on the MPC5200B Development Kit CD. The debugger connection can be found on the back panel of the system.

5.2 Basic Configuration

The basic monitor configuration delivered with the Media5200 is configured with the following software modules:

- EEPROM
- FAT
- I2C
- IDE
- PCI
- DHCP

- REGINFO
- PING
- USB

5.3 Memory Map

The boot monitor is compiled to boot out of flash at the 0xFFF00000 (Boot high) vector. The memory map is utilized as follows:

Table 5-1. U-Boot Memory Map

Description	Usage	Address Range
DRAM	Vector Table	0x00000000 - 0x00002FFF
DRAM	Unused	0x00003000 - 0x07EFFFFF
DRAM	U-Boot	0x07F00000 - 0x07FFFFFF
	Reserved	0x08000000 - 0xEFFFFFFF
MPC5200 Peripherals	MBAR	0xF0000000 - 0xF0007FFF
Internal SRAM	Unused	0xF0008000 - 0xF000BFFF
	Reserved	0xF000C000 - 0xFDFFFFFF
Flash - CS1	Unused	0xFC000000 - 0xFDFFFFFF
Flash - CS0 (CSBOOT)	Unused	0xFE000000 - 0xFFEFFFFF
Flash - CS0 (CSBOOT)	U-Boot	0xFFF00000 - 0xFFFFFFFF

5.4 Accessing Memory Using U-Boot

The following commands may be used to access any memory mapped locations within the MPC5200B.

5.4.1 MD

The **md** command may be used to display memory in byte, half word, and word increments. Syntax for this command is:

```
md [.b, .w, .l] <address>
```

example:

```
=> md 0xF0000000
f0000000:0000f000 0000ff00 0000ffff 0000fe00 ................
f0000010:0000feff 0000ffff 0000ffff 0000ffff ................

=> md.l 0xF0000000
f0000000:0000f000 0000ff00 0000ffff 0000fe00 ................
f0000010:0000feff 0000ffff 0000ffff 0000ffff ................

=> md.w 0xF0000000
f0000000:0000 f000 0000 ff00 0000 ffff 0000 fe00 ................
```

```
f0000010:0000 feff 0000 ffff 0000 ffff 0000 ffff ................

=> md.b 0xF0000000
f0000000:00 00 f0 00 00 00 ff 00 00 00 ff ff 00 00 fe 00 ................
f0000010:00 00 fe ff 00 00 ff ff 00 00 ff ff 00 00 ff ff ................
```

5.4.2 MW

The **mw** command may be used to write a memory in byte, half word, and word increments. Syntax for this command is:

```
mw [.b, .w, .l] <address> <data>
```

example:

```
=> mw 0x0 0xABCDEF12
```

This will write the value 0xABCDEF12 to address 0x0.

5.5 Environmental Variables

Environmental variable may be used to store u-boot configuration and commnads for later usage after power removed or reset occurs.

5.5.1 printenv

The **printenv** command may be used to print the currently configured environmental variables. Unless specifically saved into flash these variables will reside in ram until reset occurs or power is lost.

The default environment is

```
=> printenv
bootcmd=run install_usb
bootdelay=5
baudrate=115200
preboot=usb reset; setenv stdout serial;setenv stderr serial;echo;echo 'Autostar
ting. Press any key to abort...';echo
netdev=eth0
autoload=no
autostart=no
script_addr=0x200000
install_usb=setenv script bootrc.img;run script_usb
script_usb=run usbstart;fatload usb 0:$(usbpart) $(script_addr) $(script);autosc
r $(script_addr)
usbstart=usb reset; usb scan
usbpart=1
reset_env=erase 0xfff40000 0xfff5ffff
ethaddr=00:04:9f:00:2d:32
ethact=FEC ETHERNET
stdin=serial
stdout=serial
stderr=serial

Environment size: 541/131068 bytes
=>
```

NOTE

The “ethaddr” variable will differ from board to board.

5.5.2 setenv

The **setenv** command may be used to set an environmental variable in ram. Please note that if power is lost currenlty configured variables in ram will not be saved unless the environment has been stored to non-volatile memory.

usage:

```
=> setenv autoload n
```

5.5.3 saveenv

This command will save the current environment to flash for later usage.

usage:

```
=> saveenv
Saving Environment to Flash...
Un-Protected 1 sectors
Erasing Flash...Erasing Sector: 244 - 0xfff40000
 done
Erased 1 sectors
Writing to Flash... done
Protected 1 sectors
=>
```

5.5.4 reset_env

The command restores the flash to factory-fresh.

After booting up the Media5200, press any key to escape the auto-boot process. Then type “run **reset_env**”. This command will erase the sectors of flash needed to reset the flash to the default state. The “bootdelay” variable must be greater than 0 to escape the auto-boot process.

```
=> run reset_env

. done
Erased 1 sectors
```

5.6 Configuring Ethernet

5.6.1 Configuring a MAC Address

To configure a mac address for your Media5200 the environmental variable for **ethaddress** can be set to any desired mac address for network access. By default your Media5200 is programmed with a default MAC address. To change this address use the following command.

```
setenv ethaddr 00:11:22:33:44:55
```

5.6.2 Configuring IP Address

5.6.2.1 Static IP

To boot via TFTP the following environmental variables must be configured for operation. To set a static IP the environmental variables in Table 5-2 must be specified through the command line interface.

Table 5-2. Static IP Ethernet Configuration

Environmental Variable	Description
ipaddr	local IP address for the Media5200
serverip	TFTP/NFS server address
netmask	net mask
gatewayip	gateway IP address
netdev	eth0 - default
ethaddr[1]	MAC address

[1] Care must be taken to ensure that each MAC address on the network is unique

5.6.2.2 DHCP

The Media5200 is configured with the necessary software required to perform **dhcp** functionality. In order to utilize the U-boot **dhcp** function your **dhcp** server must be configured with the variables designated in Table 5-3

Table 5-3. DHCP Ethernet Configuration

Environmental Variable	Description	Value
ipaddr	local IP address for the Media5200	Configured by DHCP
serverip	TFTP/NFS server address	This value Must be configured AFTER DHCP IP address is acquired[1]
netmask	net mask	Obtained by DHCP
gatewayip	gateway IP address	Obtained by DHCP
netdev	Ethernet device to use for	Set by user

Table 5-3. DHCP Ethernet Configuration

Environmental Variable	Description	Value
ethaddr[2]	MAC address	Set be user
autoload	TFTP boot image from DHCP server after DHCP acquisition	no

[1] The value obtained by the DHCP server may not be applicable to your development application.

[2] Care must be taken to ensure that each MAC address on the network is unique

When the variables listed in Table 5-3 have been configured the **dhcp** command can be used to automatically obtain an IP address from the network.

example:

```
=> dhcp
BOOTP broadcast 1
DHCP client bound to address 10.81.108.96
```

5.7 Downloading An Image

Two methods of downloading an image are described in this section. Serial download and TFTP (network) download. The areas suggested for code development can be seen in Table 5-4. When code is downloaded to RAM, it may be executed directly or written to flash.

5.7.1 TFTP Download

Table 5-4. Development Code Space

Physical Memory	Usage	Address Range
DRAM	Vector Table	0x00000000 - 0x00002FFF
DRAM	User	0x00003000 - 0x07FFFFFF
Peripherals	MBAR	0xF0000000 - 0xF0007FFF
Internal SRAM	Unused	0xF0008000 - 0xF000BFFF
Flash - CS1	Unused	0xFC000000 - 0xFDFFFFFF
Flash - CS0 (CSBOOT)	Unused[1]	0xFE000000 - 0xFFEFFFFF

[1] This section will be mapped to the reset vector if boot low is selected through SW1 - Jumper 6.

The TFTP method is the fastest method of downloading large images. We suggest using this method for large images and code development.

To perform an TFTP download an IP address must be first configured using one of the methods described in 5.6/5-4. Once an IP address is configured the **tftpboot** command should be used. The syntax for the command is as follows:

usage:

```
tftp [loadaddress] [bootfilename]
```

Executing this command will download the file specified by the argumented boot filename from the IP address configured in the environmental variable **setenv**. Code will be downloaded to the load address specified at load address. The value for load address must conform to a read/writable ram location. **serverip** must be configured prior to executing the **tftp** command using the **setenv** command (see 5.5.2/5-4).

example:

```
=> dhcp
BOOTP broadcast 1
DHCP client bound to address 10.81.109.231
=> setenv serverip 10.81.109.159
=> tftp 0x20000 u-boot.bin
Using FEC ETHERNET device
TFTP from server 10.81.109.159; our IP address is 10.81.109.231
Filename 'u-boot.bin'.
Load address: 0x20000
Loading: ##########################################
done
Bytes transferred = 213872 (34370 hex)
=>
```

5.7.2 Serial Download

Two serial protocols are available for serial download over the UART connection. the **loadb** command may be used to download a binary image file using the Kermit protocol. Additionally the loads command may be used to download **srecord** (ascii) formatted images.

Usage:

```
loadb [ address ] [ baud ]
This will download a binary file over the UART using the Kermit protocol. Specifying the
address and baud rate are optional, however specifying the address to be laoded is highly
recommended.
```

Usage:

```
loads [ address ] [ baud ]
This will download an srecord file over the UART using the Kermit protocol. Specifying
the address and baud rate are optional, however specifying the address to be laoded is
highly recommended.
```

example:

```
=> loadb 0x3000
## Ready for binary (kermit) download to 0x00003000 at 115200 bps...
## Total Size      = 0x00000e62 = 3682 Bytes
## Start Addr      = 0x00003000
=> go 0x3000
```

5.8 Executing an Image

After code has been downloaded the **go** command may be used to execute an application.

Usage:

```
go [ address ] [ arg .. ]
This will start execution of code at the address specified. Arguments may be optionally
provided and passed using the EABI specification.
```

5.9 Writing an Image to Flash

Once an image has been downloaded into ram it may be written to flash by first performing an erase and then performing a memory copy to the specified flash region. Instructions for erasing a flash region can be found in 5.9/5-8.

Usage:

```
cp [ .b, .w, .l ] source target count
This will copy a region of memory from the source to the target writing to flash if
required.
```

Example:

```
=> erase 0xFE000000 0xFE03FFFF

.. done
Erased 2 sectors
=> cp.b 0x20000 0xFE000000 0x40000
Copy to Flash... done
```

5.10 Erasing Flash

Before writing to flash area of memory, it must first be empty or an error will occur. Erasing the flash may be done using the following command.

Usage:

```
erase [ sector start ] [ sector end ]
```

Care must be taken to specify ending address on a sector boundary.

WARNING

Do not erase U-Boot on the system, which is located on 0xFFF00000 to 0xFFFFFFFF. There is no hardware-assisted U-Boot recovery on the Media5200.

5.10.1 Flash Configuration

Flash information for the device provided with the Media5200 can be displayed by typing the **flinfo** command. The following information should be displayed.

```
=> flinfo

Bank # 1: CFI conformant FLASH (32 x 16)  Size: 32 MB in 256 Sectors
 Erase timeout 16384 ms, write timeout 0 ms, buffer write timeout 4096 ms, buffe
r size 32
  Sector Start Addresses:
 FC000000 E     FC020000 E     FC040000 E     FC060000 E     FC080000 E
 FC0A0000 E     FC0C0000 E     FC0E0000 E     FC100000 E     FC120000 E
 FC140000 E    FC160000 E    FC180000 E    FC1A0000 E    FC1C0000 E
 FC1E0000 E    FC200000 E    FC220000 E    FC240000 E    FC260000 E
 FC280000 E    FC2A0000 E    FC2C0000 E    FC2E0000 E    FC300000 E
 FC320000 E    FC340000 E    FC360000 E    FC380000 E    FC3A0000 E
 FC3C0000 E    FC3E0000 E    FC400000 E    FC420000 E    FC440000 E
 FC460000 E    FC480000 E    FC4A0000 E    FC4C0000 E    FC4E0000 E
 FC500000 E    FC520000 E    FC540000 E    FC560000 E    FC580000 E
 FC5A0000 E    FC5C0000 E    FC5E0000 E    FC600000 E    FC620000 E
 FC640000 E    FC660000 E    FC680000 E    FC6A0000 E    FC6C0000 E
 FC6E0000 E    FC700000 E    FC720000 E    FC740000 E    FC760000 E
 FC780000 E    FC7A0000 E    FC7C0000 E    FC7E0000 E    FC800000 E
 FC820000 E    FC840000 E    FC860000 E    FC880000 E    FC8A0000 E
 FC8C0000 E    FC8E0000 E    FC900000 E    FC920000 E    FC940000 E
 FC960000 E    FC980000 E    FC9A0000 E    FC9C0000 E    FC9E0000 E
 FCA00000 E    FCA20000 E    FCA40000 E    FCA60000 E    FCA80000 E
 FCAA0000 E    FCAC0000 E    FCAE0000 E    FCB00000 E    FCB20000 E
 FCB40000 E    FCB60000 E    FCB80000 E    FCBA0000 E    FCBC0000 E
 FCBE0000 E    FCC00000 E    FCC20000 E    FCC40000 E    FCC60000 E
 FCC80000 E    FCCA0000 E    FCCC0000 E    FCCE0000 E    FCD00000 E
 FCD20000 E    FCD40000 E    FCD60000 E    FCD80000 E    FCDA0000 E
 FCDC0000 E    FCDE0000 E    FCE00000 E    FCE20000 E    FCE40000 E
 FCE60000 E    FCE80000 E    FCEA0000 E    FCEC0000 E    FCEE0000 E
 FCF00000 E    FCF20000 E    FCF40000 E    FCF60000 E    FCF80000 E
 FCFA0000 E    FCFC0000 E    FCFE0000 E    FD000000 E    FD020000 E
 FD040000 E    FD060000 E    FD080000 E    FD0A0000 E    FD0C0000 E
 FD0E0000 E    FD100000 E    FD120000 E    FD140000 E    FD160000 E
 FD180000 E    FD1A0000 E    FD1C0000 E    FD1E0000 E    FD200000 E
 FD220000 E    FD240000 E    FD260000 E    FD280000 E    FD2A0000 E
 FD2C0000 E    FD2E0000 E    FD300000 E    FD320000 E    FD340000 E
 FD360000 E    FD380000 E    FD3A0000 E    FD3C0000 E    FD3E0000 E
 FD400000 E    FD420000 E    FD440000 E    FD460000 E    FD480000 E
 FD4A0000 E    FD4C0000 E    FD4E0000 E    FD500000 E    FD520000 E
 FD540000 E    FD560000 E    FD580000 E    FD5A0000 E    FD5C0000 E
 FD5E0000 E    FD600000 E    FD620000 E    FD640000 E    FD660000 E
 FD680000 E    FD6A0000 E    FD6C0000 E    FD6E0000 E    FD700000 E
 FD720000 E    FD740000 E    FD760000 E    FD780000 E    FD7A0000 E
 FD7C0000 E    FD7E0000 E    FD800000 E    FD820000 E    FD840000 E
 FD860000 E    FD880000 E    FD8A0000 E    FD8C0000 E    FD8E0000 E
 FD900000 E    FD920000 E    FD940000 E    FD960000 E    FD980000 E
 FD9A0000 E    FD9C0000 E    FD9E0000 E    FDA00000 E    FDA20000 E
 FDA40000 E    FDA60000 E    FDA80000 E    FDAA0000 E    FDAC0000 E
 FDAE0000 E    FDB00000 E    FDB20000 E    FDB40000 E    FDB60000 E
 FDB80000 E    FDBA0000 E    FDBC0000 E    FDBE0000 E    FDC00000 E
 FDC20000 E    FDC40000 E    FDC60000 E    FDC80000 E    FDCA0000 E
 FDCC0000 E    FDCE0000 E    FDD00000 E    FDD20000 E    FDD40000 E
 FDD60000 E    FDD80000 E    FDDA0000 E    FDDC0000 E    FDDE0000 E
 FDE00000 E    FDE20000 E    FDE40000 E    FDE60000 E    FDE80000 E
 FDEA0000 E    FDEC0000 E    FDEE0000 E    FDF00000 E    FDF20000 E
 FDF40000 E    FDF60000 E    FDF80000 E    FDFA0000 E    FDFC0000 E
 FDFE0000 E

Bank # 2: CFI conformant FLASH (32 x 16)  Size: 32 MB in 256 Sectors
 Erase timeout 16384 ms, write timeout 0 ms, buffer write timeout 4096 ms, buffe
r size 32
  Sector Start Addresses:
 FE000000 E    FE020000 E    FE040000 E    FE060000 E    FE080000 E
 FE0A0000 E    FE0C0000 E    FE0E0000 E    FE100000 E    FE120000 E
```

```
  FE140000 E    FE160000 E    FE180000 E    FE1A0000 E    FE1C0000 E
  FE1E0000 E    FE200000 E    FE220000 E    FE240000 E    FE260000 E
  FE280000 E    FE2A0000 E    FE2C0000 E    FE2E0000 E    FE300000 E
  FE320000 E    FE340000 E    FE360000 E    FE380000 E    FE3A0000 E
  FE3C0000 E    FE3E0000 E    FE400000 E    FE420000 E    FE440000 E
  FE460000 E    FE480000 E    FE4A0000 E    FE4C0000 E    FE4E0000 E
  FE500000 E    FE520000 E    FE540000 E    FE560000 E    FE580000 E
  FE5A0000 E    FE5C0000 E    FE5E0000 E    FE600000 E    FE620000 E
  FE640000 E    FE660000 E    FE680000 E    FE6A0000 E    FE6C0000 E
  FE6E0000 E    FE700000 E    FE720000 E    FE740000 E    FE760000 E
  FE780000 E    FE7A0000 E    FE7C0000 E    FE7E0000 E    FE800000 E
  FE820000 E    FE840000 E    FE860000 E    FE880000 E    FE8A0000 E
  FE8C0000 E    FE8E0000 E    FE900000 E    FE920000 E    FE940000 E
  FE960000 E    FE980000 E    FE9A0000 E    FE9C0000 E    FE9E0000 E
  FEA00000 E    FEA20000 E    FEA40000 E    FEA60000 E    FEA80000 E
  FEAA0000 E    FEAC0000 E    FEAE0000 E    FEB00000 E    FEB20000 E
  FEB40000 E    FEB60000 E    FEB80000 E    FEBA0000 E    FEBC0000 E
  FEBE0000 E    FEC00000 E    FEC20000 E    FEC40000 E    FEC60000 E
  FEC80000 E    FECA0000 E    FECC0000 E    FECE0000 E    FED00000 E
  FED20000 E    FED40000 E    FED60000 E    FED80000 E    FEDA0000 E
  FEDC0000 E    FEDE0000 E    FEE00000 E    FEE20000 E    FEE40000 E
  FEE60000 E    FEE80000 E    FEEA0000 E    FEEC0000 E    FEEE0000 E
  FEF00000 E    FEF20000 E    FEF40000 E    FEF60000 E    FEF80000 E
  FEFA0000 E    FEFC0000 E    FEFE0000 E    FF000000 E    FF020000 E
  FF040000 E    FF060000 E    FF080000 E    FF0A0000 E    FF0C0000 E
  FF0E0000 E    FF100000 E    FF120000 E    FF140000 E    FF160000 E
  FF180000 E    FF1A0000 E    FF1C0000 E    FF1E0000 E    FF200000 E
  FF220000 E    FF240000 E    FF260000 E    FF280000 E    FF2A0000 E
  FF2C0000 E    FF2E0000 E    FF300000 E    FF320000 E    FF340000 E
  FF360000 E    FF380000 E    FF3A0000 E    FF3C0000 E    FF3E0000 E
  FF400000 E    FF420000 E    FF440000 E    FF460000 E    FF480000 E
  FF4A0000 E    FF4C0000 E    FF4E0000 E    FF500000 E    FF520000 E
  FF540000 E    FF560000 E    FF580000 E    FF5A0000 E    FF5C0000 E
  FF5E0000 E    FF600000 E    FF620000 E    FF640000 E    FF660000 E
  FF680000 E    FF6A0000 E    FF6C0000 E    FF6E0000 E    FF700000 E
  FF720000 E    FF740000 E    FF760000 E    FF780000 E    FF7A0000 E
  FF7C0000 E    FF7E0000 E    FF800000 E    FF820000 E    FF840000 E
  FF860000 E    FF880000 E    FF8A0000 E    FF8C0000 E    FF8E0000 E
  FF900000 E    FF920000 E    FF940000 E    FF960000 E    FF980000 E
  FF9A0000 E    FF9C0000 E    FF9E0000 E    FFA00000 E    FFA20000 E
  FFA40000 E    FFA60000 E    FFA80000 E    FFAA0000 E    FFAC0000 E
  FFAE0000 E    FFB00000 E    FFB20000 E    FFB40000 E    FFB60000 E
  FFB80000 E    FFBA0000 E    FFBC0000 E    FFBE0000 E    FFC00000 E
  FFC20000 E    FFC40000 E    FFC60000 E    FFC80000 E    FFCA0000 E
  FFCC0000 E    FFCE0000 E    FFD00000 E    FFD20000 E    FFD40000 E
  FFD60000 E    FFD80000 E    FFDA0000 E    FFDC0000 E    FFDE0000 E
  FFE00000 E    FFE20000 E    FFE40000 E    FFE60000 E    FFE80000 E
  FFEA0000 E    FFEC0000 E    FFEE0000 E    FFF00000      FFF20000
  FFF40000 E    FFF60000 E    FFF80000 E    FFFA0000 E    FFFC0000 E
  FFFE0000 E
 =>
```

Chapter 6 FPGA Register Space

6.1 FPGA Register Space

The FPGA interface uses 64 Kbytes in the LocalPlus Bus Chip Select 2 address space and another 64 Kbytes in the Chip Select 3 address space. The LocalPlus bus mode used is the Muxed Mode with 25 bits of address and 32 bits of data with active ACK. When running the pre-installed dBUG monitor program CS1 is mapped to address 0xF0010000 and supports only 32 bit wide accesses, i.e. byte and half-word accesses are not supported. CS2 is mapped to 0xF0020000.

Table 6-1 shows an outline of the CS1 register map. The register map of CS2 is identical to the one given in the data sheet of the external DUART from Texas Instruments (TL16C752BPT).

MOST generates an interrupt to the processor core routed to IRQ1.

Table 6-1. External Address Space

Offset	Register Name	Type	Bits	Address
0x0400	REV	R	32	0x02090307
0x0404	INT_MASK1	R/W	32	0x00000000
0x0408	INT_STAT1	R/W	32	0x00000000
0x040C	INT_MASK0	R/W	32	0x00000000
0x0410	INT_STAT0	R/W	32	0x00000000
0x0450	SCRATCH	R/W	32	0x00000000
0x0480	PERI_CTRL	R/W	32	0xC4000000
0x0484	ID_CTRL	R/W	32	0x00000000
0x0488	ID_DATA_HI	R	32	0x00000000
0x048C	ID_DATA_LO	R	32	0x00000000
0x0490	AUDIO_CTRL	R/W	32	0x70000000
0x0494	GPS_CTRL	R/W	32	0x0000000
0x0708	MOST_ASYNC_TX_FIFO	W	32	0x00000000
0x070C	MOST_ASYNC_RX_FIFO	R	32	0x00000000
0x0714	MOST_STAT	R/W	32	0x00000000

Table 6-1. External Address Space (continued)

Offset	Register Name	Type	Bits	Address
0x0718	MOST_CP_CMD	R/W	32	0x00000000
0x071C	MOST_INT_MASK	R/W	32	0x00000000
0x0720	MOST_INT_STAT	R/W	32	0x00000000
0x0724	MOST_SBC	R/W	32	0x00000000
0x0728	MOST_STXCA	R/W	32	0x00000000
0x072C	MOST_SRXCA	R/W	32	0x00000000
0x0730	MOST_ARXMA	R/W	32	0x00000000
0x0734	MOST_ARXMGA	R/W	32	0x00000000
0x0738	MOST_AS_TXFILL_LVL	R/W	32	0x00000000
0x073C	MOST_AS_RXFILL_LVL	R/W	32	0x00000000
0x0748	MOST_ENABLE	R/W	32	0x00000000
0x074C	MOST_ASFILL	R/W	32	0x00000000

NOTE

The numbers of the bits in the registers are in PowerPC notation, i.e. Bit 0 is MSB, Bit 31 is LSB.

6.1.1 REV

This register holds the current FPGA identification code, major and minor revision number.

REV																
BIT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
FIELD	ID															
RESET	0x0209															
R/W	R															
BIT	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
FIELD	MAJ_REV								MIN_REV							
RESET	0x03								0x06							
R/W	R								R							
ADDR	0x0400															

Figure 6-1. REV Register

Table 6-2. REV Register Bit Encoding

Field	Description
ID	Internal identification code of the FPGA. The MSB defines the card type (00=AIOX, 01=Total5200, 02=Media5200). The lower nibble of the LSB defines the size of the FIFOs (2^n). The upper nibble of the LSB is undefined.
MAJ_REV	Major revision number of the FPGA. Differences in the major revision number are used to distinguish FPGA revisions that are incompatible from a software stand of view.
MIN_REV	Minor revision number of the FPGA. This number points out changes in the FPGA that are software compatible like bug fixes in the design.

6.1.2 INT_MASK1

This register is used to mask out certain interrupt sources by writing a ‘0’ to a bit position. Writing a ‘1’ to a bit position activates the assigned interrupt source. An activated interrupt source will generate an $\overline{\text{IRQ1}}$ core pin interrupt.

This register uses the same layout as the INT_STAT1 register (Section 6.1.3, “INT_STAT1).

INT_MASK1																
BIT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
FIELD	HAL	CP	RXA	TXA	RXAF	TXAF	RESERVED		TS	GPS	RESERVED					
RESET	0	0	0	0	0	0	0		0	0	0					
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		R/W	R/W	R/W					
BIT	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
FIELD	RESERVED															
RESET	0															
R/W	R/W															
ADDR	0x0404															

Figure 6-2. INT_MASK1 Register

Table 6-3. INT_MASK1 Register Bit Encoding

Field	Description
HAL	MOST HAL master interrupt
CP	MOST Control Port State Machine interrupt
RXA	Asynchronous Receiver FIFO interrupt
TXA	Asynchronous Transmitter FIFO interrupt
RXAF	Asynchronous Receiver FIFO fill level interrupt

Table 6-3. INT_MASK1 Register Bit Encoding (continued)

Field	Description
TXAF	Asynchronous Transmitter FIFO fill level interrupt
TS	Touch Screen interrupt
GPS	GPS interrupt

6.1.3 INT_STAT1

INT_STAT1																
BIT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
FIELD	HAL	CP	RXA	TXA	RXAF	TXAF	RESERVED		TS	GPS	RESERVED					
RESET	0	0	0	0	0	0	0		0	0	0					
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		R/W	R/W	R/W					
BIT	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
FIELD	RESERVED															
RESET	0															
R/W	R/W															
ADDR	0x0408															

Figure 6-3. INT_STAT1 Register

This register displays the status of the interrupt controller. A bit position set to ‘1’ indicates that the interrupt assigned to this bit is pending. Pending interrupts can be removed by writing a ‘1’ to the interrupt bit. This operation can only be successful if the interrupt condition was removed before. To mask an interrupt use the INT_MASK1 register (Section 6.1.2, “INT_MASK1).

Table 6-4. INT_STAT1 Register Bit Encoding

Field	Description
HAL	MOST HAL Master Interrupt. Indicates that a HAL interrupt occurred. This bit is connected to the MASTER_INTR_N signal of the HAL. This signal is the masked sum of the HAL’s interrupt register (see registers MOST_INT_MASK and MOST_INT_STAT in Section 6.1.17, “MOST_INT_MASK and Section 6.1.18, “MOST_INT_STAT.
CP	MOST Control Port FSM Interrupt. This interrupt signals that the Control Port of the OS8104 is ready for a new command, i.e. has finished processing the last command.This bit is connected to the CP_FSM_INTR_N signal of the HAL. It is asserted by the HAL to signal the completion of a Control Port access.
RXA	Asynchronous Receiver FIFO Interrupt. This interrupt signals that an asynchronous message was received. This bit is connected to the RX_ASYNC_INTR_N signal of the HAL. It is asserted by the HAL when the asynchronous reception state machine (RX_ASYNC_FSM) has detected the end of a message.

Table 6-4. INT_STAT1 Register Bit Encoding (continued)

Field	Description
TXA	Asynchronous Transmitter FIFO Interrupt. This interrupt signals that an asynchronous message has been transmitted. This bit is connected to the TX_ASYNC_INTR_N signal of the HAL. It is asserted by the HAL when the asynchronous transmission state machine (TX_ASYNC_FSM) has sent all frames of a message and the OS8104 negates AINT_N again.
RXAF	Asynchronous Receiver FIFO Fill Level Interrupt. This interrupt signals that the actual fill level of the asynchronous reception FIFO is larger than the programmed threshold. See register MOST_AS_RX_FILL_LVL Section 6.1.25, “MOST_AS_RXFILL_LVL for details.
TXAF	Asynchronous Transmitter FIFO fill level interrupt. This interrupt signals that the actual fill level of the asynchronous transmission FIFO is less than the programmed threshold. See register MOST_AS_TX_FILL_LVL Section 6.1.24, “MOST_AS_TXFILL_LVL.
TS	Touch Screen interrupt. This interrupt is signalled by the touch screen controller and flags a pen down event.
GPS	GPS interrupt. This interrupt is triggered by the GPS module plugged into the GPS module socket.

6.1.4 INT_MASK0

INT_MASK0																
BIT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
FIELD	PCI3	PCI2	PCI1	PCI0	EU0	EU1	RESERVED									
RESET	0	0	0	0	0	0	0									
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W									
BIT	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
FIELD	RESERVED															
RESET	0															
R/W	R/W															
ADDR	0x040C															

Figure 6-4. INT_MASK0

This register is used to mask out certain interrupt sources by writing a ‘0’ to a bit position. Writing a ‘1’ to a bit position activates the assigned interrupt source. An activated interrupt source will generate an $\overline{\text{IRQ0}}$ core pin interrupt.

This register uses the same layout as the INT_STAT0 register Section 6.1.5, “INT_STAT0. See this register for details about the interrupts.

Table 6-5. INT_MASK0 Bit Encodings

Field	Description
PCI3	PCI /IRQ3
PCI2	PCI /IRQ2
PCI1	PCI /IRQ1
PCI0	PCI /IRQ0
EU0	External UART interrupt channel A
EU1	External Dual UART interrupt channel B

6.1.5 INT_STAT0

INT_STAT0																
BIT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
FIELD	PCI3	PCI2	PCI1	PCI0	EU0	EU1	RESERVED									
RESET	0	0	0	0	0	0	0									
R/W	R	R	R	R	R	R	R									
BIT	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
FIELD	RESERVED															
RESET	0															
R/W	R															
ADDR	0x0410															

Figure 6-5. INT_STAT0 Register

This register displays the status of the PCI and DUART interrupt lines. A bit position set to ‘1’ indicates that the interrupt line assigned to this bit is asserted. If the line is deasserted the bit will be ’0’. That means, the interrupt condition needs to be removed at the PCI or DUART device to deassert IRQ1. To mask an interrupt use the INT_MASK0 register Section 6.1.4, “INT_MASK0.

Table 6-6. INT_STAT0 Register Bit Encodings

Field	Description
PCI3	PCI /IRQ3. PCI interrupt line /IRQ3.
PCI2	PCI /IRQ3. PCI interrupt line /IRQ3.
PCI1	PCI /IRQ3. PCI interrupt line /IRQ3.
PCI0	PCI /IRQ3. PCI interrupt line /IRQ3.

Table 6-6. INT_STAT0 Register Bit Encodings (continued)

Field	Description
EU0	External UART interrupt A. This interrupt signals an interrupt request from the external Dual UART channel A.
EU1	External UART interrupt B. This interrupt signals an interrupt request from the external Dual UART channel B.

6.1.6 SCRATCH

SCRATCH																
BIT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
FIELD	SCRATCH															
RESET	0															
R/W	R/W															
BIT	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
FIELD	SCRATCH															
RESET	0															
R/W	R/W															
ADDR	0x0450															

Figure 6-6. SCRATCH Register

This register is a general purpose read / write register which has no effect on other portions of the chip. Its MAIN purpose is to serve debugging the FPGA. It may be used by software as a scratch pad register.

6.1.7 PERI_CTRL

HSO																
BIT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
FIELD	FOFF	FON	FOFF1	FON1	ARST	AOE	APOW	DRST	DMODE	RESERVED						
RESET	1	1	0	0	0	1	0	0	0	0						
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W						
BIT	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
FIELD	RESERVED															
RESET	0															
R/W	R/W															
ADDR	0x0480															

Figure 6-7. PERI_CTRL Register

The PERI_CTRL register controls features of various peripherals.

Table 6-7. PERI_CTRL Register Bit Encoding

Field	Description
FOFF	ForceOff internal UART and external UART0. Set the level of the ForceOff line of the RS232 level shifter used by the internal UART and the external UART0.
FON	ForceOn internal UART and external UART0. Set the level of the ForceOn line of the RS232 level shifter used by the internal UART and external UART0.
FOFF1	ForceOff external UART1. Set the level of the ForceOff line of the RS232 level shifter used by external UART1.
FON1	ForceOn external UART1. Set the level of the ForceOn line of the RS232 level shifter used by external UART1.
ARST	ATA Software Reset. If set to ’0’ the ATA software reset is asserted, if set to ’1’ the ATA software reset is negated.
AOE	ATA Output Enable. If set to ’0’ the ATA 3.3V to 5V level shifters are enabled.
APOW	ATA Power On. If set to ’1’ the ATA power supply of the level shifters is switched on.
DRST	DSP Reset. If set to ’0’ the DSP reset is asserted, if set to ’1’ the DSP reset is negated.
DMODE	DSP Mode Control DSP Mode line.

6.1.8 ID_CTRL Register

HSI																
BIT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
FIELD	RESERVED															
RESET	0															
R/W	R/W															
BIT	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
FIELD	RESERVED															
RESET	0															
R/W	R/W															
ADDR	0x0484															

Figure 6-8. ID_CTRL Register

NOTE

The ID_CTRL register will contain a method to control reading the 64 bit unique serial number of the board. This feature will be implemented in a future version of the FPGA.

6.1.9 ID_DATA_HI

HSI																
BIT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
FIELD	RESERVED															
RESET	0															
R/W	R/W															
BIT	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
FIELD	RESERVED															
RESET	0															
R/W	R/W															
ADDR	0x0488															

Figure 6-9. ID_DATA_HI

NOTE

The ID_DATA_HI register will contain the upper 32 bits of the 64 bit unique serial number of the board. This feature will be implemented in a future version of the FPGA.

6.1.10 ID_DATA_LO

HSI																
BIT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
FIELD	RESERVED															
RESET	0															
R/W	R/W															
BIT	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
FIELD	RESERVED															
RESET	0															
R/W	R/W															
ADDR	0x048C															

Figure 6-10. ID_DATA_HI

NOTE

The ID_DATA_LO register will contain the lower 32 bits of the 64 bit unique serial number of the board. This feature will be implemented in a future version of the FPGA.

6.1.11 AUDIO_CTRL

HSI																
BIT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
FIELD	MUTE	MLIN	MSUR	MCEN	MICX	SLIN	SSUR	SCEN	RESERVED							
RESET	0	1	1	1	0	0	0	0	0							
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W							
BIT	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
FIELD	RESERVED															
RESET	0															
R/W	R/W															
ADDR	0x0490															

Figure 6-11. AUDIO_CTRL Register

Table 6-8. AUDIO_CTRL Register Bit Encoding

Field	Description
MUTE	dsp_mute_ind
MLIN	mute_line
MSUR	mute_surround
MCEN	mute_cen_lfe
MICX	mic_input_cross
SLIN	ac97_dsp_line
SSUR	ac97_dsp_surround
SCEN	ac97_dsp_cen_lfe

6.1.12 GPS_CTRL

HSI																
BIT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
FIELD	RST	DIR	VAL	RESERVED												
RESET	0	0	0	0												
R/W	R/W	R/W	R/W	R/W												
BIT	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
FIELD	RESERVED															
RESET	0															
R/W	R/W															
ADDR	0x0494															

Figure 6-12. GPS_CTRL Register

The GPS_CTRL register controls the reset and the GPIO line of the GPS module socket.

Table 6-9. GPS_CTRL Register Bit Encoding

Field	Description
RST	Reset - If set to ’0’ reset is asserted, if set to ’1’ reset is negated.
DIR	GPIO - If set to ’0’ the GPIO is set to input, if set to ’1’ the GPIO is set to output.
VAL	GPIO - If DIR is set to ’0’ reading VAL returns the level the GPIO is stimulated to. Writing VAL does not have an effect. If DIR is set to ’1’ writing VAL changes the level the GPIO drives. Reading VAL returns what is currently driven.

6.1.13 MOST_ASYNC_TX_FIFO

MOST_ASYNC_TX_FIFO																
BIT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
FIELD	DOUT															
RESET	-															
R/W	W															
BIT	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
FIELD	DOUT															
RESET	-															
R/W	W															
ADDR	0x0708															

Figure 6-13. MOST_ASYNC_TX_FIFO

This register is the interface to the asynchronous TX FIFO of the HAL. Every write access to this register will put another quadlet in the FIFO. The asynchronous TX FIFO only accepts quadlets. This means four bytes have to be written at a time.

Letting the TX FIFO run empty during a asynchronous packet transmission will result into a broken packet. Writing more quadlets in the FIFO than it can fit will also result in a corrupted packet. See Section 6.1.24, “MOST_AS_TXFILL_LVL and Section 6.1.27, “MOST_ASFILL on how to prevent FIFO over- and underflow.

6.1.14 MOST_ASYNC_RX_FIFO

MOST_ASYNC_RX_FIFO																
BIT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
FIELD	DIN															
RESET	-															
R/W	R															
BIT	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
FIELD	DIN															
RESET	-															
R/W	R															
ADDR	0x070C															

Figure 6-14. MOST_ASYNC_RX_FIFO

This register is the interface to the asynchronous RX FIFO of the HAL. The asynchronous RX FIFO contains packets that passed the address checking of the HAL. That means only packets that are addressed to this node (direct, group cast or broad cast) will be stored into the FIFO. Packets are stored quadlet wise (four bytes). So reading the asynchronous RX FIFO must be done quadlet by quadlet. See Section 6.1.25, “MOST_AS_RXFILL_LVL and Section 6.1.27, “MOST_ASFILL on how to prevent FIFO overflows.

6.1.15 MOST_STAT

MOST_STAT																
BIT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
FIELD	RESERVED															
RESET	0															
R/W	R															
BIT	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
FIELD	RESERVED							BFS	RESERVED						STAT	ERR
RESET	0							0	0						0	0
R/W	R							R	R						R	R
ADDR	0x0714															

Figure 6-15. MOST_STAT

NOTE

See OS8104 data sheet for details.

Table 6-10. MOST_STAT Bit Encoding

Field	Description
BFS	Bigfoot BF_STATUS pin
STAT	MOST Status pin
ERR	MOST Error pin

6.1.16 MOST_CP_CMD

MOST_CP_CMD																
BIT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
FIELD	RESERVED															
RESET	0															
R/W	R/W															
BIT	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
FIELD	RESERVED				MRST	EXEC	OP1	OP0	DATA							
RESET	0				0	0	0	0	0							
R/W	R/W				R/W	R/W	R/W	R/W	R/W							
ADDR	0x0718															

Figure 6-16. MOST_CP_CMD Register

The MOST_CP_CMD register grants access to the control port of the MOST controller.

Table 6-11. MOST_CP_CMD Register Bit Encoding

Field	Description
MRST	MOST Reset If the MRST bit is set to '1' the MOST controller will be reset. After the execution of the reset the bit will cleared automatically. During reset access to the MOST controller must be postponed until the bit is cleared. ***Please note:*** This bit must be read at least once after it was cleared to terminate the reset cycle. Otherwise the HAL is rendered inoperable until this bit is read.
EXEC	Execute Control Port Command If the EXEC bit is set to '1' it starts the execution of the operation defined in the OP bits. After execution of the control port command the bit will be cleared again and a Control Port FSM Interrupt will be generated. Note: When reading the Control Port Status the CP_FLOW signal of the MOST chip does not go high.
OP[1:0]	Control Port Operation Mode 00 : Write Control Port MAP 01 : Read Control Port Status 10 : Write Control Port Data 11 : Read Control Port Data
DATA	Control Port Data These are the data bytes transfered over the Control Port.

6.1.17 MOST_INT_MASK

MOST_INT_MASK																
BIT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
FIELD	RESERVED															
RESET	0															
R/W	R/W															
BIT	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
FIELD	RESERVED								MAINT	MINT	ERR	SRX	STX	ARX	ATX	CPFSM
RESET	0								0	0	0	0	0	0	0	0
R/W	R/W								R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	0x071C															

Figure 6-17. MOST_INT_MASK Register

This register is used to mask out certain interrupt sources of the MOST HAL Master Interrupt by writing a ‘0’ to a bit position. Writing a ‘1’ to a bit position activates the assigned interrupt source.

This register uses the same layout as the MOST_INT_STAT register described in Section 6.1.18, “MOST_INT_STAT. See this register for details about the interrupts.

Table 6-12. MOST_INT_MASK Register Bit Encoding

Field	Description
MAINT	MOST Asynchronous Interrupt
MINT	MOST Control Message Interrupt
ERR	Error Interrupt
SRX	Synchronous RX Interrupt
STX	Synchronous TX Interrupt
ARX	Asynchronous RX Interrupt
ATX	Asynchronous TX Interrupt
CPFSM	MOST Control Port FSM Interrupt

6.1.18 MOST_INT_STAT

MOST_INT_STAT																
BIT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
FIELD	RESERVED															
RESET	0															
R/W	R/W															
BIT	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
FIELD	RESERVED								MAINT	MINT	ERR	SRX	STX	ARX	ATX	CPFSM
RESET	0								0	0	0	0	0	0	0	0
R/W	R/W								R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	0x0720															

Figure 6-18. MOST_INT_STAT Register

This register displays the status of the MOST HAL Master Interrupt controller. A bit position set to ‘1’ indicates that the interrupt assigned to this bit is pending. Pending interrupts can be removed by writing a ‘1’ to the interrupt bit. This operation can only be successful if the interrupt condition was removed before. To mask an interrupt use the MOST_INT_MASK register (see Section 6.1.17, “MOST_INT_MASK). Further details to these interrupts can be found in the HAL documentation. Figure 6-19 illustrates how the MOST HAL Master interrupt is cascaded into the INT_STAT register (Section 6.1.3, “INT_STAT1).

Table 6-13. MOST_INT_STAT Register Bit Encoding

Field	Description
MAINT	MOST Asynchronous Interrupt This bit indicates that the MOST chip has generated an asynchronous interrupt. This interrupt is used when sending asynchronous (packet) data.
MINT	Interrupt Status for MOST Control Message Interrupt This bit indicates that the MOST chip has generated a control message interrupt. This interrupt indicates events from power-up to “message received”.
ERR	Error Interrupt This interrupt is not implemented yet (HAL feature).
SRX	Synchronous RX Interrupt This interrupt is not implemented yet (HAL feature).
STX	Synchronous TX Interrupt This interrupt is not implemented yet (HAL feature).
ARX	Asynchronous RX Interrupt This interrupt signals that an asynchronous message was received. This bit is connected to the RX_ASYNC_INTR_N signal of the HAL. It is asserted by the HAL when the asynchronous reception state machine (RX_ASYNC_FSM) has detected the end of a message.

Table 6-13. MOST_INT_STAT Register Bit Encoding (continued)

Field	Description
ATX	Asynchronous TX Interrupt This interrupt signals that an asynchronous message has been transmitted. This bit is connected to the TX_ASYNC_INTR_N signal of the HAL. It is asserted by the HAL when the asynchronous transmission state machine (TX_ASYNC_FSM) has sent all frames of a message and the OS8104 negates AINT_N again.
CPFSM	MOST Control Port FSM Interrupt This interrupt signals that the Control Port of the OS8104 is ready for a new command, i.e. has finished processing the last command. This bit is connected to the CP_FSM_INTR_N signal of the HAL. It is asserted by the HAL to signal the completion of a Control Port access.

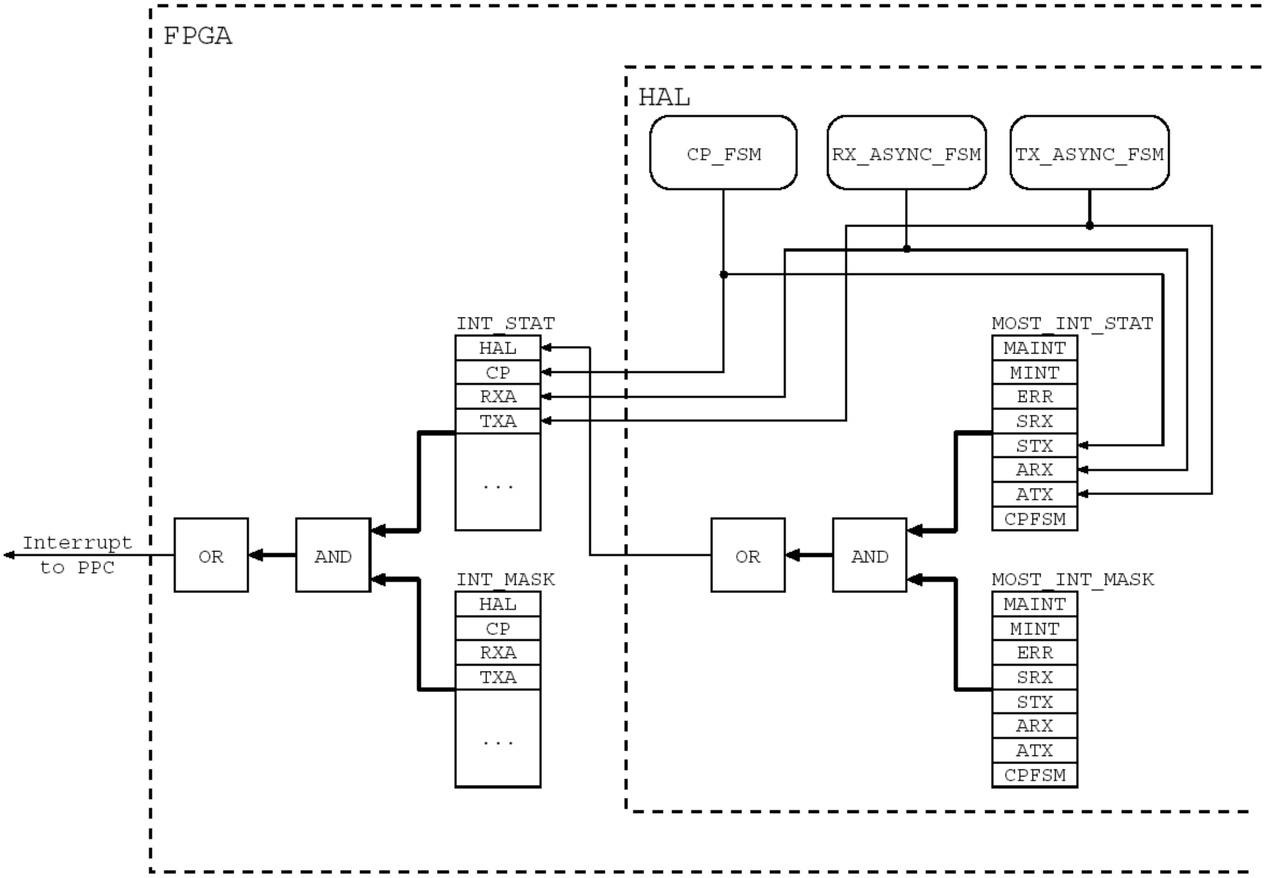


Figure 6-19. MOST HAL Master Interrupt cascade

6.1.19 MOST_SBC

MOST_SBC																
BIT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
FIELD	RESERVED															
RESET	0															
R/W	R/W															
BIT	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
FIELD	RESERVED												SBC			
RESET	0												0			
R/W	R/W												R/W			
ADDR	0x0724															

Figure 6-20. MOST_SBC Register

The Synchronous Bandwidth Register enables the MOST Source Port Controller to distinguish between synchronous and asynchronous data reads from the Source Port of the MOST chip.

The value written to this register must be equal to the value written in the SBC register of the MOST chip decremented by 1.

Table 6-14. MOST_SBC Register Bit Encoding

Field	Description
SBC	

6.1.20 MOST_STXCA

<table>
<tr><td colspan="17">MOST_STXCA</td></tr>
<tr><td>BIT</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td></tr>
<tr><td>FIELD</td><td colspan="16">RESERVED</td></tr>
<tr><td>RESET</td><td colspan="16">0</td></tr>
<tr><td>R/W</td><td colspan="16">R/W</td></tr>
<tr><td>BIT</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td></tr>
<tr><td>FIELD</td><td colspan="12">RESERVED</td><td colspan="4">TX_CHANNELS</td></tr>
<tr><td>RESET</td><td colspan="12">0</td><td colspan="4">0</td></tr>
<tr><td>R/W</td><td colspan="12">R/W</td><td colspan="4">R/W</td></tr>
<tr><td>ADDR</td><td colspan="16">0x0728</td></tr>
</table>

Figure 6-21. MOST_STXCA Register

The Synchronous TX Channel Allocation Register defines the number of transferred synchronous transmit channels. The quadlets are read from the synchronous FIFO and written to the Source Ports of the MOST chip.

It should be noted that the number of transmitted synchronous quadlets is one more than the value in this register.

NOTE

The contents of this register should remain 0 at all times (as no synchronous FIFO is available)!

Table 6-15. MOST_STXCA Register Bit Encoding

Field	Description
TX_CHANNELS	

6.1.21 MOST_SRXCA

MOST_SRXCA																
BIT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
FIELD	RESERVED															
RESET	0															
R/W	R/W															
BIT	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
FIELD	RESERVED												RX_CHANNELS			
RESET	0												0			
R/W	R/W												R/W			
ADDR	0x072C															

Figure 6-22. MOST_SRXCA Register

The Synchronous RX Channel Allocation Register defines the number of transferred synchronous receive channels. The quadlets are read from the Source Ports of the MOST chip and transferred to the synchronous FIFO.

It should be noted that the number of received synchronous quadlets is one more than the value in this register.

NOTE

The contents of this register should remain 0 at all times (as no synchronous FIFO is available)!

Table 6-16. MOST_SRXCA Register Bit Encoding

Field	Description
RX_CHANNELS	

6.1.22 MOST_ARXMA

MOST_ARXMA																
BIT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
FIELD	RESERVED															
RESET	0															
R/W	R/W															
BIT	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
FIELD	RX_ASYNC_ADDR_LOW_BYTE								RX_ASYNC_ADDR_HIGH_BYTE							
RESET	0								0							
R/W	R/W								R/W							
ADDR	0x0730															

Figure 6-23. MOST_ARXMA Register

The Asynchronous RX Message Address is used by the asynchronous address comparison. If the address comparison for asynchronous receive messages is enabled (ASYNC_ADDR_COMP_EN signal is high) the Asynchronous RX Message Address is compared with the destination address of every incoming message. If the address matches, the message will be transferred to the Asynchronous Receive FIFO (see Section 6.1.14, “MOST_ASYNC_RX_FIFO, and Section 6.1.26, “MOST_ENABLE).

Table 6-17. MOST_ARXMA Register Bit Encoding

Field	Description
	RX_ASYNC_ADDR_LOW_BYTE
	RX_ASYNC_ADDR_HIGH_BYTE

6.1.23 MOST_ARXMGA

MOST_ARXMGA																
BIT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
FIELD	RESERVED															
RESET	0															
R/W	R/W															
BIT	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
FIELD	RESERVED								GROUP_ADDR							
RESET	0								0							
R/W	R/W								R/W							
ADDR	0x0734															

Figure 6-24. MOST_ARXMGA Register

The Asynchronous RX Message Group Address is used by the asynchronous group address comparison. If the address comparison for asynchronous receive messages is enabled (ASYNC_ADDR_COMP_EN signal is high) and the high byte of the destination address is 0x03, the Asynchronous RX Message Group Address is compared against the low byte of the destination address. If the low byte of the destination address and the Group Address matches, the message will be transferred to the Asynchronous Receive FIFO (see Section 6.1.14, “MOST_ASYNC_RX_FIFO, and Section 6.1.26, “MOST_ENABLE).

Table 6-18. MOST_ARXMGA Register Bit Encoding

Field	Description
	GROUP_ADDR

6.1.24 MOST_AS_TXFILL_LVL

MOST_AS_TXFILL_LVL																
BIT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
FIELD	RESERVED															
RESET	0															
R/W	R/W															
BIT	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
FIELD	RESERVED							TXFILL_LVL								
RESET	0							0								
R/W	R/W							R/W								
ADDR	0x0738															

Figure 6-25. MOST_AS_TXFILL_LVL Register

The MOST_AS_TXFILL_LVL Register controls the asynchronous TX FIFO fill level interrupt generation. If TXFILL_LVL is 0x000 no interrupt will be generated. If TXFILL_LVL is greater than 0x000 an interrupt will be generated if the present TX FIFO fill level is smaller than TXFILL_LVL. The interrupt will show up in the INT_STAT register at bit TXAF (see Section 6.1.3, “INT_STAT1). It can be masked using the INT_MASK register (see Section 6.1.2, “INT_MASK1).

Table 6-19. MOST_AS_TXFILL_LVL Register Bit Encoding

Field	Description
TXFILL_LVL	TXFILL_LVL

6.1.25 MOST_AS_RXFILL_LVL

<table>
<tr><td colspan="17">MOST_AS_RXFILL_LVL</td></tr>
<tr><td>BIT</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td></tr>
<tr><td>FIELD</td><td colspan="16">RESERVED</td></tr>
<tr><td>RESET</td><td colspan="16">0</td></tr>
<tr><td>R/W</td><td colspan="16">R/W</td></tr>
<tr><td>BIT</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td></tr>
<tr><td>FIELD</td><td colspan="7">RESERVED</td><td colspan="9">RX_FILL_LVL</td></tr>
<tr><td>RESET</td><td colspan="7">0</td><td colspan="9">0</td></tr>
<tr><td>R/W</td><td colspan="7">R/W</td><td colspan="9">R/W</td></tr>
<tr><td>ADDR</td><td colspan="16">0x073C</td></tr>
</table>

Figure 6-26. MOST_AS_RXFILL_LVL Register

The MOST_AS_RXFILL_LVL Register controls the asynchronous RX FIFO fill level interrupt generation. If RXFILL_LVL is 0x000 no interrupt will be generated. If RXFILL_LVL is greater than 0x000 an interrupt will be generated if the present RX FIFO fill level is greater than RXFILL_LVL. The interrupt will show up in the INT_STAT register at bit RXAF (see Section 6.1.3, “INT_STAT1). It can be masked using the INT_MASK register (see Section 6.1.2, “INT_MASK1).

Table 6-20. MOST_AS_RXFILL_LVL Register Bit Encoding

Field	Description
	RX_FILL_LVL

6.1.26 MOST_ENABLE

MOST_ENABLE																
BIT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
FIELD	RESERVED															
RESET	0															
R/W	R/W															
BIT	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
FIELD	RESERVED									SRRES	STRES	CMP	RXS	TXS	RXA	TXA
RESET	0									0	0	0	0	0	0	0
R/W	R/W									R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	0x0748															

Figure 6-27. MOST_ENABLE Register

This register is the MAIN control register to switch on the handling of receive/transmit data by the HAL. If all activities are disabled the HAL will still generate parallel combined mode MOST_RD_N and MOST_WR_N signals to the OS8104 MOST controller but no data will be taken from or passed to the FIFO.

Table 6-21. MOST_ENABLE Register Bit Encoding

Field	Description
SRRES	Soft reset asynchronous RX FIFO If this bit is set to ‘1’ the asynchronous RX FIFO is reset. This will result in an empty RX FIFO. To put the RX FIFO back into operation the bit has to be set back to ‘0’ again.
STRES	Soft reset asynchronous TX FIFO If this bit is set to ‘1’ the asynchronous TX FIFO is reset. This will result in an empty TX FIFO. To put the TX FIFO back into operation the bit has to be set back to ‘0’ again.
CMP	Enables address compare mode in Asynchronous mode If this bit is set to ‘1’ Message Address and Message Group Address checking is enabled. See registers MOST_ARXMA and MOST_ARXMGA (see Section 6.1.22, “MOST_ARXMA and Section 6.1.23, “MOST_ARXMGA).
RXS	Enables Synchronous mode receiver If this bit is set to ‘1’ the reception of synchronous data is enabled.
TXS	Enables Synchronous mode transmitter If this bit is set to ‘1’ the transmission of synchronous data is enabled.

Table 6-21. MOST_ENABLE Register Bit Encoding (continued)

Field	Description
RXA	Enables Asynchronous mode receiver If this bit is set to ‘1’ the reception of asynchronous data is enabled. See register MOST_ASYNC_RX_FIFO (see Section 6.1.14, “MOST_ASYNC_RX_FIFO) how to receive data.
TXA	Enables Asynchronous mode transmitter If this bit is set to ‘1’ the transmission of asynchronous data is enabled. This bit is automatically set to ‘0’ when the asynchronous TX FIFO runs empty, i.e. the isochronous mode transmitter is disabled. Please note that setting this bit to ‘0’ disables the isochronous transmitter immediately.

6.1.27 MOST_ASFILL

MOST_ASFILL																
BIT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
FIELD	RESERVED							TXLVL								
RESET	0							0								
R/W	R							R								
BIT	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
FIELD	RESERVED							RXLVL								
RESET	0							0								
R/W	R							R								
ADDR	0x074C															

Figure 6-28. MOST_ASFILL Register

This register holds the present FIFO fill level of the asynchronous MOST channel.

Table 6-22. MOST_ASFILL Register Bit Encoding

Field	Description
RXLVL	Asynchronous RX FIFO fill level
TXLVL	Asynchronous TX FIFO fill level

6.2 MOST

6.2.1 Hardware Overview

The MOST portion of the Media5200 platform consists of the OS8104 MOST chip from Oasis Silicon Systems on the MOST extension board and the FPGA programmed with a matching firmware. The OS8104 can be accessed through the FPGA only. A detailed register description of the FPGA can be found in Section 6.1, “FPGA Register Space.

The MAIN component of the MOST portion of the FPGA is the HAL (Hardware Abstraction Layer). The HAL’s purpose is to supply an easy to use front-end to the OS8104. The HAL is surrounded by an interrupt controller and two FIFOs for asynchronous packet handling. An extension with two synchronous FIFOs is planned. The interrupt controller manages the interrupt request generated by the HAL and routes them through to the CPU. The asynchronous message FIFOs decouple the CPU from picking up or delivering and receiving quadlets to the HAL at the exact time it needs or returns them, thus lowering real time requirements.

6.2.2 Initializing the OS8104

Before MOST traffic can be processed the OS8104 MOST controller from Oasis must be initialized. Initialization can vary greatly depending on how MOST is going to be used. Therefore, only a standard initialization will be presented here. Further details can be found in the OS8104 MOST Transceiver data sheet supplied by Oasis.

6.2.2.1 Parallel Combined mode

The Parallel Combined mode is activated by writing to the PCMA register of the OS8104.

6.2.2.2 Source Port SCK rate

The source port SCK rate must be set to 256. This is necessary because the Parallel Combined mode is used. SCK rate is controlled by the SDC2 register of the OS8104.

6.2.2.3 ERROR pin configuration

To be able to detect the bus lock condition, the encoding error and SPDIF-Lock error signaling on the ERROR pin must be masked so that only transceiver lock errors are signaled on this pin. This can be done by writing the XSR register of the OS8104.

NOTE

SPDIF-Lock and encoding error signalling on the ERROR pin must be disabled. Bus lock error signalling on the ERROR pin must be enabled. Otherwise proper operation of the FPGA is not guaranteed.

6.2.2.4 Master / Slave configuration

The selection whether the node is going to be the master or a slave on the MOST ring is done by writing the XCR register of the OS8104. When selecting master or slave mode the transmitter can be enabled and bypass mode disabled. When the node wants to take an active role on the MOST ring the transmitter must be enabled and bypass mode must be disabled.

6.2.2.5 Enable PLL

The PLL can be enabled using the CM1 register of the OS8104. In this register RMCK must be set to 1024 and the oscillator divider must be set to 384. In the master mode sync must be set to crystal. In a slave node sync must be set to RX.

6.2.2.6 Set Synchronous Bandwidth

If the node is configured to be the master on the MOST ring it can change the synchronous bandwidth at any time. This can be done right after configuring the node to be the master. The synchronous bandwidth is configured using the bSBC register of the OS8104. Immediately after changing the SBC register of the OS8104 the MOST_SBC register of the HAL must be configured to. See Section 6.1.19, “MOST_SBC for details about this register.

NOTE

> Only even SBC values are allowed for the bSBC register of the OS8104. This is a limitation of the HAL.

If the node is configured to be a slave on the MOST ring it must either poll the SBC register of the OS8104 or read it after interrupt notification by the OS8104. Whenever the value of the SBC register is changed (and after reading it for the first time) the MOST_SBC register of the HAL must be changed, too.

6.2.3 Synchronous Traffic

Synchronous traffic is not supported in this version. An extension is planned in the future.

6.2.4 Asynchronous Traffic

Here the very basics of receiving and transmitting of asynchronous data will be discussed. For specialized things like FIFO fill level interrupts and address filtering refer to the corresponding registers (see Section 6.1, “FPGA Register Space).

6.2.4.1 Receiving asynchronous data

Before asynchronous data packets can be received the routing engine of the OS8104 must be set up. The reason for this is that the HAL needs the incoming packet to be formatted in a slightly different way than it is received.

The first modification needed is that the status quadlet is not expected to be the last quadlet of frame but the first. The second modification is that the status quadlet has to consist of the Error field, destination address and the Start field where destination address is only needed in the status quadlet of the first frame. And the third modification is that the quadlets must be swapped pairwise. All these modifications can be applied to the incoming asynchronous data by a careful setup of the routing engine. See Table 6-23 for the receive status quadlet format.

Table 6-23. RX Status Quadlet Format

Byte	Function
0	Error: The Error byte can be used by the application to check if there was an error during the data transfer of the asynchronous message. 0x00 signals that there was no error.
1	Destination Address (high byte): The high byte of the recipient’s address. This field is valid only in the first frame of the packet.
2	Destination Address (low byte): The low byte of the recipient’s address. This field is valid only in the first frame of the packet.
3	Start: If the Start byte is not 0x00 an asynchronous packet starts.

The os8104SetupAsyncRE C-function illustrates how to set up the routing engine:

```
void os8104SetupAsyncRE (void) {
int i;
int async;
int n;
 async = os8104_sbc * 4;             /* offset to first async byte in a frame */
n = 0x40 - async;                         /* number of async bytes in a frame */
re[0+0x40+async] = async;
re[1+0x40+async] = async + 1;
re[2+0x40+async] = async + 2;
re[3+0x40+async] = async + 3;
re[4+0x40+async] = 0x3F; /* Status Quadlet: Error */
re[5+0x40+async] = async + 1;/* Dest. Address High */
re[6+0x40+async] = async + 2;/* Dest. Address Low */
re[7+0x40+async] = 0x3D; /* Start */
for (i = 8; i < n; i += 8) {
         re[i+0+0x40+async] = async + 0 + i;
         re[i+1+0x40+async] = async + 1 + i;
         re[i+2+0x40+async] = async + 2 + i;
         re[i+3+0x40+async] = async + 3 + i;
         re[i+4+0x40+async] = async - 4 + i;
         re[i+5+0x40+async] = async - 3 + i;
         re[i+6+0x40+async] = async - 2 + i;
         re[i+7+0x40+async] = async - 1 + i;
}
}
```

Now the reception of asynchronous packets can be enabled using bit RXA in the MOST_ENABLE register (see Section 6.1.26, “MOST_ENABLE). Once enabled, all incoming packets will be put in the asynchronous RX FIFO (see MOST_ASYNC_RX_FIFO register, Section 6.1.14, “MOST_ASYNC_RX_FIFO).

When reading the packet from the asynchronous RX FIFO, the packet consists of a standard MOST data packet interleaved with status quadlets (see Table 6-23). The first quadlet of the packet is always a status quadlet and every frame starts with a status quadlet. See Table 6-24 for an example how a received packet looks like (status quadlets are shaded)

NOTE

The size of a frame depends on the setting of the synchronous bandwidth (SBC).

Table 6-24. RX FIFO Reception Example (SBC=0x0C)

Frame	Quadlet	Byte	Function
1	1	0	Error
1	1	1	Destination Address (high byte)
1	1	2	Destination Address (low byte)
1	1	3	Start
1	2	4	Arbitration
1	2	5	Destination Address (high byte)
1	2	6	Destination Address (low byte)
1	2	7	Length of data (in quadlets)
1	3	8	Source Address (high byte)
1	3	9	Source Address (low byte)
1	3	10	Data byte 0
1	3	11	Data byte 1
1	4	...	Data byte 2, 3, 4, 5
2	1	0	Error
2	1	1	Junk
2	1	2	Junk
2	1	3	Start
2	2	4	Data byte 6
2	...	...	Data byte 7 - 17
3	1	0	Error
3	1	1	Junk
3	1	2	Junk
3	1	3	Start
3	2	4	Data byte 18

6.2.4.2 Transmitting asynchronous data

To transmit an asynchronous data packet the routing engine must not be altered from its original setting. This time the packet data has to be written slightly different to the data packet format documented in theOS8104 data sheet. The slight difference is that the quadlets (all frames including status quadlet) must be swapped pairwise compared to the original data packet format. That is, quadlet 1 must be written to the asynchronous TX FIFO (register MOST_ASYNC_TX_FIFO, Section 6.1.13, “MOST_ASYNC_TX_FIFO) after quadlet 2 has been written and so on, i.e., the quadlets must be written in order 2,1,4,3,6,5,8,7,...

Table 6-25. TX FIFO Transmission Example (SBC=0x0C) - no pair-wise swapping applied

Frame	Quadlet	Byte	Function
1	1	0	Start Transmit
1	1	1	reserved
1	1	2	ACK
1	1	3	reserved
1	2	4	Arbitration
1	2	5	Destination Address (high byte)
1	2	6	Destination Address (low byte)
1	2	7	Length of data (in quadlets)
1	3	8	Source Address (high byte)
1	3	9	Source Address (low byte)
1	3	10	Data byte 0
1	3	11	Data byte 1
1	4	...	Data byte 2, 3, 4, 5
2	1	0	Error
2	1	1	Junk
2	1	2	Junk
2	1	3	Start
2	2	4	Data byte 6
2	...	...	Data byte 7 - 17
3	1	0	Error
3	1	1	Junk
3	1	2	Junk
3	1	3	Start
3	2	4	Data byte 18

NOTE

Due to a restriction of the OS8104, an asynchronous data packet must have a length of at least 4 frames. Transmitting fewer frames will result in corrupted packets.

To enable the transmission of asynchronous data packets in the TX FIFO set the TXA bit in the MOST_ENABLE register (see Section 6.1.26, “MOST_ENABLE). If this bit is enabled while writing to the asynchronous TX FIFO make sure the FIFO doesn’t underflow. If that happens the packet will be corrupted.

A method to prevent underflows is to wait until the contents of the asynchronous TX FIFO has been sent, disable transmission of asynchronous data packets, fill the asynchronous TX FIFO with the data packets to be sent and re-enable transmission of asynchronous data packets again. Of course, by using the FIFO fill level alarm feature of the HAL (see Section 6.1.24, “MOST_AS_TXFILL_LVL) dynamic reloading of the asynchronous TX FIFO can be achieved.

Appendix A Build History

A.1 Prototype Build 1: 50 Systems

February 2006

MAIN board serial numbers beginning with 5018xxx and 5019xxx

MPC5200B, mix of CoralP and CoralPA, some without housings, 90ns FLASH, non-RoHS, MOST boards not populated.

The following MAIN boards contain CoralPA:

```
5018976, 5018979, 5019145, 5019146, 5019148, 5019149, 5019150, 5019152, 5019164, 5019165,
5019173, 5019174, 5019176, 5019183, 5019184
```

A.2 Production Build 1: 100 Systems

May 2006

MPC5200B, CoralPA, all in housings, 120ns FLASH, non-RoHS

Appendix B Glossary of Terms and Abbreviations

The glossary contains an alphabetical list of terms, phrases, and abbreviations used in this book. Some of the terms and definitions included in the glossary are reprinted from IEEE Standard 754-1985, *IEEE Standard for Binary Floating-Point Arithmetic*, copyright ©1985 by the Institute of Electrical and Electronics Engineers, Inc. with the permission of the IEEE.

A

Architecture. A detailed specification of requirements for a processor or computer system. It does not specify details of how the processor or computer system must be implemented; instead it provides a template for a family of compatible *implementations*.

Asynchronous exception. *Exceptions* that are caused by events external to the processor's execution. In this document, the term 'asynchronous exception' is used interchangeably with the word *interrupt*.

Atomic access. A bus access that attempts to be part of a read-write operation to the same address uninterrupted by any other access to that address (the term refers to the fact that the transactions are indivisible). The PowerPC architecture implements atomic accesses through the **lwarx**/**stwcx.** instruction pair.

B

BAT (block address translation) mechanism. A software-controlled array that stores the available block address translations on-chip.

Beat. A single state on the 603e bus interface that may extend across multiple bus cycles. A 603e transaction can be composed of multiple address or data *beats*.

Biased exponent. An *exponent* whose range of values is shifted by a constant (bias). Typically a bias is provided to allow a range of positive values to express a range that includes both positive and negative values.

Big-endian. A byte-ordering method in memory where the address *n* of a word corresponds to the *most-significant byte*. In an addressed memory word, the bytes are ordered (left to right) 0, 1, 2, 3, with 0 being the *most-significant byte*. See *Little-endian*.

Block. An area of memory that ranges from 128 Kbyte to 256 Mbyte whose size, translation, and protection attributes are controlled by the BAT mechanism.

Boundedly undefined. A characteristic of certain operation results that are not rigidly prescribed by the PowerPC architecture. Boundedly- undefined results for a given operation may vary among implementations and between execution attempts in the same implementation.

Although the architecture does not prescribe the exact behavior for when results are allowed to be boundedly undefined, the results of executing instructions in contexts where results are allowed to be *boundedly undefined* are constrained to ones that could have been achieved by executing an arbitrary sequence of defined instructions, in valid form, starting in the state the machine was in before attempting to execute the given instruction.

Branch folding. The replacement with target instructions of a branch instruction and any instructions along the not-taken path when a branch is either taken or predicted as taken.

Branch prediction. The process of guessing whether a branch will be taken. Such predictions can be correct or incorrect; the term ‘predicted’ as it is used here does not imply that the prediction is correct (successful). The PowerPC architecture defines a means for static branch prediction as part of the instruction encoding.

Branch resolution. The determination of whether a branch is taken or not taken. A branch is said to be resolved when the processor can determine which instruction path to take. If the branch is resolved as predicted, the instructions following the predicted branch that may have been speculatively executed can complete. If the branch is not resolved as predicted, instructions on the mis-predicted path, and any results of speculative execution, are purged from the pipeline and fetching continues from the nonpredicted path.

Burst. A multiple-beat data transfer whose total size is typically equal to a cache block.

Bus clock. Clock that causes the bus state transitions.

Bus master. The owner of the address or data bus; the device that initiates or requests the transaction.

C

Cache. High-speed memory containing recently accessed data or instructions (subset of MAIN memory).

Cache block. A small region of contiguous memory that is copied from memory into a *cache*. The size of a cache block may vary among processors; the maximum block size is one *page*. In PowerPC processors, *cache coherency* is MAINtained on a cache-block basis. Note that the term ‘cache block’ is often used interchangeably with ‘cache line.’

Cache coherency. An attribute wherein an accurate and common view of memory is provided to all devices that share the same memory system. Caches are coherent if a processor performing a read from its cache is supplied with data corresponding to the most recent value written to memory or to another processor’s cache.

Cache flush. An operation that removes from a cache any data from a specified address range. This operation ensures that any modified data within the specified address range is written back to MAIN memory. This operation is generated typically by a Data Cache Block Flush (**dcbf**) instruction.

Caching-inhibited. A memory update policy in which the *cache* is bypassed and the load or store is performed to or from MAIN memory.

Cast out. A *cache block* that must be written to memory when a cache miss causes a cache block to be replaced.

Changed bit. One of two *page history bits* found in each *page table entry* (PTE). The processor sets the changed bit if any store is performed into the *page*. See also *Page access history bits* and *Referenced bit*.

Clean. An operation that causes a cache block to be written to memory, if modified, and then left in a valid, unmodified state in the cache.

Clear. To cause a bit or bit field to register a value of zero. See also *Set*.

Context synchronization. An operation that ensures that all instructions in execution complete past the point where they can produce an *exception*, that all instructions in execution complete in the context in which they began execution, and that all subsequent instructions are *fetched* and executed in the new context. Context synchronization may result from executing specific instructions (such as **isync** or **rfi**) or when certain events occur (such as an exception).

Copy-back operation. A cache operation in which a cache line is copied back to memory to enforce cache coherency. Copy-back operations consist of snoop push-out operations and cache cast-out operations.

D

Denormalized number. A nonzero floating-point number whose exponent has a reserved value, usually the format's minimum, and whose explicit or implicit leading significand bit is zero.

Direct-mapped cache. A cache in which each MAIN memory address can appear in only one location within the cache, operates more quickly when the memory request is a cache hit.

Direct-store segment access. An access to an I/O address space. The 603 defines separate memory-mapped and I/O address spaces, or segments, distinguished by the corresponding segment register T bit in the address translation logic of the 603. If the T bit is cleared, the memory reference is a normal memory-mapped access and can use the virtual memory management hardware of the 603. If the T bit is set, the memory reference is a direct-store access.

E

Effective address (EA). The 32-bit address specified for a load, store, or an instruction fetch. This address is then submitted to the MMU for translation to either a *physical memory* address or an I/O address.

Exception. A condition encountered by the processor that requires special, supervisor-level processing.

Exception handler. A software routine that executes when an exception is taken. Normally, the exception handler corrects the condition that caused the exception, or performs some other meaningful task (that may include aborting the program that caused the exception). The address for each exception handler is identified by an exception vector offset defined by the architecture and a prefix selected via the MSR.

Exclusive state. MEI state (E) in which only one caching device contains data that is also in system memory.

Execution synchronization. A mechanism by which all instructions in execution are architecturally complete before beginning execution (appearing to begin execution) of the next instruction. Similar to context synchronization but doesn't force the contents of the instruction buffers to be deleted and refetched.

Exponent. In the binary representation of a floating-point number, the exponent is the component that normally signifies the integer power to which the value two is raised in determining the value of the represented number. See also *Biased exponent*.

F

Feed-forwarding. A 603e feature that reduces the number of clock cycles that an execution unit must wait to use a register. When the source register of the current instruction is the same as the destination register of the previous instruction, the result of the previous instruction is routed to the current instruction at the same time that it is written to the register file. With feed-forwarding, the destination bus is gated to the waiting execution unit over the appropriate source bus, saving the cycles which would be used for the write and read.

Fetch. Retrieving instructions from either the cache or MAIN memory and placing them into the instruction queue.

Floating-point register (FPR). Any of the 32 registers in the floating-point register file. These registers provide the source operands and destination results for floating-point instructions. Load instructions move data from memory to FPRs and store instructions move data from FPRs to memory. The FPRs are 64 bits wide and store floating-point values in double-precision format.

Floating-point unit. The functional unit in the 603e processor responsible for executing all floating-point instructions.

Flush. An operation that causes a cache block to be invalidated and the data, if modified, to be written to memory.

Fraction. In the binary representation of a floating-point number, the field of the *significand* that lies to the right of its implied binary point.

G

General-purpose register (GPR). Any of the 32 registers in the general-purpose register file. These registers provide the source operands and destination results for all integer data manipulation instructions. Integer load instructions move data from memory to GPRs and store instructions move data from GPRs to memory.

Guarded. The guarded attribute pertains to out-of-order execution. When a page is designated as guarded, instructions and data cannot be accessed out-of-order.

H

Harvard architecture. An architectural model featuring separate caches and other memory management resources for instructions and data.

Hashing. An algorithm used in the *page table* search process.

I

IEEE 754. A standard written by the Institute of Electrical and Electronics Engineers that defines operations and representations of binary floating-point numbers.

Illegal instructions. A class of instructions that are not implemented for a particular PowerPC processor. These include instructions not defined by the PowerPC architecture. In addition, for 32-bit implementations, instructions that are defined only for 64-bit implementations are considered to be illegal instructions. For 64-bit implementations instructions that are defined only for 32-bit implementations are considered to be illegal instructions.

Implementation. A particular processor that conforms to the PowerPC architecture, but may differ from other architecture-compliant implementations for example in design, feature set, and implementation of *optional* features. The PowerPC architecture has many different implementations.

Imprecise exception. A type of *synchronous exception* that is allowed not to adhere to the precise exception model (see *Precise exception*). The PowerPC architecture allows only floating-point exceptions to be handled imprecisely.

Instruction queue. A holding place for instructions fetched from the current instruction stream.

Integer unit. The functional unit in the 603e responsible for executing all integer instructions.

In-order. An aspect of an operation that adheres to a sequential model. An operation is said to be performed in-order if, at the time that it is performed, it is known to be required by the sequential execution model. See *Out-of-order*.

Instruction latency. The total number of clock cycles necessary to execute an instruction and make ready the results of that instruction.

Interrupt. An external signal that causes the 603e to suspend current execution and take a predefined exception.

K

Key bits. A set of key bits referred to as Ks and Kp in each segment register and each BAT register. The key bits determine whether supervisor or user programs can access a *page* within that *segment* or *block*.

Kill. An operation that causes a *cache block* to be invalidated without writing any modified data to memory.

L

Latency. The number of clock cycles necessary to execute an instruction and make ready the results of that execution for a subsequent instruction.

L2 cache. See *Secondary cache*.

Least-significant bit (lsb). The bit of least value in an address, register, field, data element, or instruction encoding.

Least-significant byte (LSB). The byte of least value in an address, register, data element, or instruction encoding.

Little-endian. A byte-ordering method in memory where the address *n* of a word corresponds to the *least-significant byte*. In an addressed memory word, the bytes are ordered (left to right) 3, 2, 1, 0, with 3 being the *most-significant byte*. See *Big-endian*.

M

Mantissa. The decimal part of logarithm.

MEI (modified/exclusive/invalid). *Cache coherency* protocol used to manage caches on different devices that share a memory system. Note that the PowerPC architecture does not specify the implementation of a MEI protocol to ensure cache coherency.

Memory access ordering. The specific order in which the processor performs load and store memory accesses and the order in which those accesses complete.

Memory-mapped accesses. Accesses whose addresses use the page or block address translation mechanisms provided by the MMU and that occur externally with the bus protocol defined for memory.

Memory coherency. An aspect of caching in which it is ensured that an accurate view of memory is provided to all devices that share system memory.

Memory consistency. Refers to agreement of levels of memory with respect to a single processor and system memory (for example, on-chip cache, secondary cache, and system memory).

Memory management unit (MMU). The functional unit that is capable of translating an *effective* (logical) *address* to a physical address, providing protection mechanisms, and defining caching methods.

Modified state. MEI state (M) in which one, and only one, caching device has the valid data for that address. The data at this address in external memory is not valid.

Most-significant bit (msb). The highest-order bit in an address, registers, data element, or instruction encoding.

Most-significant byte (MSB). The highest-order byte in an address, registers, data element, or instruction encoding.

N

NaN**.** An abbreviation for not a number; a symbolic entity encoded in floating-point format. There are two types of NaNs—signaling NaNs and quiet NaNs.

No-op. No-operation. A single-cycle operation that does not affect registers or generate bus activity.

Normalization. A process by which a floating-point value is manipulated such that it can be represented in the format for the appropriate precision (single- or double-precision). For a floating-point value to be representable in the single- or double-precision format, the leading implied bit must be a 1.

O

OEA (operating environment architecture). The level of the architecture that describes PowerPC memory management model, supervisor-level registers, synchronization requirements, and the exception model. It also defines the time-base feature from a supervisor-level perspective. Implementations that conform to the PowerPC OEA also conform to the PowerPC UISA and VEA.

Optional. A feature, such as an instruction, a register, or an exception, that is defined by the PowerPC architecture but not required to be implemented.

Out-of-order. An aspect of an operation that allows it to be performed ahead of one that may have preceded it in the sequential model, for example, speculative operations. An operation is said to be performed out-of-order if, at the time that it is performed, it is not known to be required by the sequential execution model. See *In-order*.

Out-of-order execution. A technique that allows instructions to be issued and completed in an order that differs from their sequence in the instruction stream.

Overflow. An condition that occurs during arithmetic operations when the result cannot be stored accurately in the destination register(s). For example, if two 32-bit numbers are multiplied, the result may not be representable in 32 bits. Since the 32-bit registers of the 603e cannot represent this sum, an overflow condition occurs.

P

Page. A region in memory. The OEA defines a page as a 4-Kbyte area of memory, aligned on a 4-Kbyte boundary.

Page access history bits. The *changed* and *referenced* bits in the PTE keep track of the access history within the page. The referenced bit is set by the MMU whenever the page is accessed for a read or write operation. The changed bit is set when the page is stored into. See *Changed bit* and *Referenced bit*.

Page fault. A page fault is a condition that occurs when the processor attempts to access a memory location that does not reside within a *page* not currently resident in *physical memory*. On PowerPC processors, a page fault exception condition occurs when a matching, valid *page table entry* (PTE[V] = 1) cannot be located.

Page table. A table in memory is comprised of *page table entries*, or PTEs. It is further organized into eight PTEs per PTEG (page table entry group). The number of PTEGs in the page table depends on the size of the page table (as specified in the SDR1 register).

Page table entry (PTE). Data structures containing information used to translate *effective address* to physical address on a 4-Kbyte page basis. A PTE consists of 8 bytes of information in a 32-bit processor and 16 bytes of information in a 64-bit processor.

Park. The act of allowing a bus master to MAINtain bus mastership without having to arbitrate.

Physical memory. The actual memory that can be accessed through the system’s memory bus.

Pipelining. A technique that breaks operations, such as instruction processing or bus transactions, into smaller distinct stages or tenures (respectively) so that a subsequent operation can begin before the previous one has completed.

Precise exceptions. A category of exception for which the pipeline can be stopped so instructions that preceded the faulting instruction can complete and subsequent instructions can be flushed and re-dispatched after exception handling has completed. See *Imprecise exceptions*.

Primary opcode. The most-significant 6 bits (bits 0–5) of the instruction encoding that identifies the type of instruction.

Program order. The order of instructions in an executing program. More specifically, this term is used to refer to the original order in which program instructions are fetched into the instruction queue from the cache.

Protection boundary. A boundary between *protection doMAINs*.

Protection doMAIN. A protection doMAIN is a segment, a virtual page, a BAT area, or a range of unmapped effective addresses. It is defined only when the appropriate relocate bit in the MSR (IR or DR) is 1.

Q

Quiesce. To come to rest. The processor is said to quiesce when an exception is taken or a **sync** instruction is executed. The instruction stream is stopped at the decode stage and executing instructions are allowed to complete to create a controlled context for instructions that may be affected by out-of-order, parallel execution. See *Context synchronization*.

Quiet NaN. A type of *NaN* that can propagate through most arithmetic operations without signaling exceptions. A quiet NaN is used to represent the results of certain invalid operations, such as invalid arithmetic operations on infinities or on NaNs, when invalid. See *Signaling NaN*.

R

rA. The **r**A instruction field is used to specify a GPR to be used as a source or destination.

rB. The **r**B instruction field is used to specify a GPR to be used as a source.

rD. The **r**D instruction field is used to specify a GPR to be used as a destination.

rS. The **r**S instruction field is used to specify a GPR to be used as a source.

Real address mode. An MMU mode when no address translation is performed and the *effective address* specified is the same as the physical address. The processor's MMU is operating in real address mode if its ability to perform address translation has been disabled through the MSR registers IR and/or DR bits.

Record bit. Bit 31 (or the Rc bit) in the instruction encoding. When it is set, updates the condition register (CR) to reflect the result of the operation.

Referenced bit. One of two *page history bits* found in each *page table entry*. The processor sets the *referenced bit* whenever the page is accessed for a read or write. See also *Page access history bits*.

Register indirect addressing. A form of addressing that specifies one GPR that contains the address for the load or store.

Register indirect with immediate index addressing. A form of addressing that specifies an immediate value to be added to the contents of a specified GPR to form the target address for the load or store.

Register indirect with index addressing. A form of addressing that specifies that the contents of two GPRs be added together to yield the target address for the load or store.

Rename register. Temporary buffers used by instructions that have finished execution but have not completed.

Reservation. The processor establishes a reservation on a *cache block* of memory space when it executes an **lwarx** instruction to read a memory semaphore into a GPR.

Reservation station. A buffer between the dispatch and execute stages that allows instructions to be dispatched even though the results of instructions on which the dispatched instruction may depend are not available.

RISC (reduced instruction set computing). An *architecture* characterized by fixed-length instructions with nonoverlapping functionality and by a separate set of load and store instructions that perform memory accesses.

S

Scan interface. The 603e test interface.

Secondary cache. A cache memory that is typically larger and has a longer access time than the primary cache. A secondary cache may be shared by multiple devices. Also referred to as L2, or level-2, cache.

Set (*v*)**.** To write a nonzero value to a bit or bit field; the opposite of *clear*. The term 'set' may also be used to generally describe the updating of a bit or bit field.

Set (*n*)**.** A subdivision of a *cache*. Cacheable data can be stored in a given location in one of the sets, typically corresponding to its lower-order address bits. Because several memory locations can map to the same location, cached data is typically placed in the set whose *cache block* corresponding to that address was used least recently. See *Set-associative*.

Set-associative. Aspect of cache organization in which the cache space is divided into sections, called *sets*. The cache controller associates a particular MAIN memory address with the contents of a particular set, or region, within the cache.

Shadowing. Shadowing allows a register to be updated by instructions that are executed out of order without destroying machine state information.

Signaling NaN. A type of *NaN* that generates an invalid operation program exception when it is specified as arithmetic operands. See *Quiet NaN*.

Significand. The component of a binary floating-point number that consists of an explicit or implicit leading bit to the left of its implied binary point and a fraction field to the right.

Simplified mnemonics. Assembler mnemonics that represent a more complex form of a common operation.

Slave. The device addressed by a master device. The slave is identified in the address tenure and is responsible for supplying or latching the requested data for the master during the data tenure.

Snooping. Monitoring addresses driven by a bus master to detect the need for coherency actions.

Snoop push. Response to a snooped transaction that hits a modified cache block. The cache block is written to memory and made available to the snooping device.

Split-transaction. A transaction with independent request and response tenures.

Split-transaction bus. A bus that allows address and data transactions from different processors to occur independently.

Stage. The term ‘stage’ is used in two different senses, depending on whether the pipeline is being discussed as a physical entity or a sequence of events. In the latter case, a stage is an element in the pipeline during which certain actions are performed, such as decoding the instruction, performing an arithmetic operation, or writing back the results. Typically, the latency of a stage is one processor clock cycle. Some events, such as dispatch, write-back, and completion, happen instantaneously and may be thought to occur at the end of a stage. An instruction can spend multiple cycles in one stage. An integer multiply, for example, takes multiple cycles in the execute stage. When this occurs, subsequent instructions may stall. An instruction may also occupy more than one stage simultaneously, especially in the sense that a stage can be seen as a physical resource—for example, when instructions are dispatched they are assigned a place in the CQ at the same time they are passed to the execute stage. They can be said to occupy both the complete and execute stages in the same clock cycle.

Stall. An occurrence when an instruction cannot proceed to the next stage.

Static branch prediction. Mechanism by which software (for example, compilers) can hint to the machine hardware about the direction a branch is likely to take.

Superscalar machine. A machine that can issue multiple instructions concurrently from a conventional linear instruction stream.

Supervisor mode. The privileged operation state of a processor. In supervisor mode, software, typically the operating system, can access all control registers and can access the supervisor memory space, among other privileged operations.

Synchronization. A process to ensure that operations occur strictly *in order*. See *Context synchronization* and *Execution synchronization*.

Synchronous exception. An *exception* that is generated by the execution of a particular instruction or instruction sequence. There are two types of synchronous exceptions, *precise* and *imprecise*.

System memory. The physical memory available to a processor.

T

Tenure. The period of bus mastership. For the 603e, there can be separate address bus tenures and data bus tenures. A tenure consists of three phases: arbitration, transfer, and termination.

TLB (translation lookaside buffer). A cache that holds recently-used *page table entries*.

Throughput. The measure of the number of instructions that are processed per clock cycle.

Transaction. A complete exchange between two bus devices. A transaction is typically comprised of an address tenure and one or more data tenures, which may overlap or occur separately from the address tenure. A transaction may be minimally comprised of an address tenure only.

Transfer termination. Signal that refers to both signals that acknowledge the transfer of individual beats (of both single-beat transfer and individual beats of a burst transfer) and to signals that mark the end of the tenure.

U

UISA (user instruction set architecture). The level of the architecture to which user-level software should conform. The UISA defines the base user-level instruction set, user-level registers, data types, floating-point memory conventions and exception model as seen by user programs, and the memory and programming models.

Underflow. A condition that occurs during arithmetic operations when the result cannot be represented accurately in the destination register. For example, underflow can happen if two floating-point fractions are multiplied and the result requires a smaller *exponent* and/or *mantissa* than the single-precision format can provide. In other words, the result is too small to be represented accurately.

User mode. The operating state of a processor used typically by application software. In user mode, software can access only certain control registers and can access only user memory space. No privileged operations can be performed. Also referred to as problem state.

V

VEA (virtual environment architecture). The level of the *architecture* that describes the memory model for an environment in which multiple devices can access memory, defines aspects of the cache model, defines cache control instructions, and defines the time-base facility from a user-level perspective. *Implementations* that conform to the PowerPC VEA also adhere to the UISA, but may not necessarily adhere to the OEA.

Virtual address. An intermediate address used in the translation of an *effective address* to a physical address.

Virtual memory. The address space created using the memory management facilities of the processor. Program access to *virtual memory* is possible only when it coincides with *physical memory.*

W

Way. A location in the cache that holds a cache block, its tags and status bits.

Word. A 32-bit data element.

Write-back. A cache memory update policy in which processor write cycles are directly written only to the cache. External memory is updated only indirectly, for example, when a modified cache block is *cast out* to make room for newer data.

Write-through. A cache memory update policy in which all processor write cycles are written to both the cache and memory.