SA-IOP Evaluation Board

Developer's Manual

January 2001

Order Number: 278235-002

Contents

Figures

Tables

Introduction 1

This document describes the SA-IOP evaluation board. This evaluation board is for the Intel® StrongARM SA-110 microprocessor, Intel® 21285 core logic controller, and Intel® 21554 embedded PCI-to-PCI bridge. The SA-IOP is a PCI card used as an add-in card. It is designed to:

- Provide a software environment for embedded applications.
- Allow benchmarks of prototype software algorithms.
- Act as a reference platform for operating system ports.
- Demonstrate the performance of the 21554 PCI interface and I_2O* capabilities, as well as the 21285 memory controller and internal functional blocks.
- Provide a building block that can be used to build software-accurate models of I_2O target applications.

1.1 Evaluation Board Features

The SA-110 microprocessor serves as the I/O processor for the SA-IOP. The board allows the developer to connect PCI devices to the SA-110 microprocessor/21285 intelligent subsystem using one standard 64-bit PCI connector and a custom connector for two 32-bit devices. The PCI devices are hidden from the host behind the 21554, which creates a secondary 4 GB address space. Figure 1-1 shows a functional block diagram of the board. Features of the SA-IOP are:

- SA-110 microprocessor I/O processor operating at any one core clock frequency (between 88.3 MHz and 233 MHz)
- 21554 embedded PCI-to-PCI bridge with 64-bit primary and secondary interface
- 21285 core logic controller with SDRAM and flash ROM interface, DMA and interrupt controllers, doorbell registers/mailboxes, and I_2O message unit
- 168-Pin DIMM socket supporting between 16 Mbytes and 128 Mbytes of synchronous DRAM
- Serial console port
- Eight user-programmable LEDs
- 4 MB of 21285/SA-110 microprocessor flash ROM organized in 32-bit width
- 1 MB of 21554 flash ROM
- 4 KB serial EPROM for 21554 preload or VPD
- Socket for 8-bit wide EEPROM or ROM emulator
- Standard 64-bit PCI connector on secondary PCI bus
- Non-standard 32-bit PCI connector for two bus masters on secondary PCI bus
- Access to 8-bit X-Bus though headers
- JTAG header
- GPIO interface for software directed SPD communication with DIMM

Figure 1-1. SA-IOP Functional Block Diagram

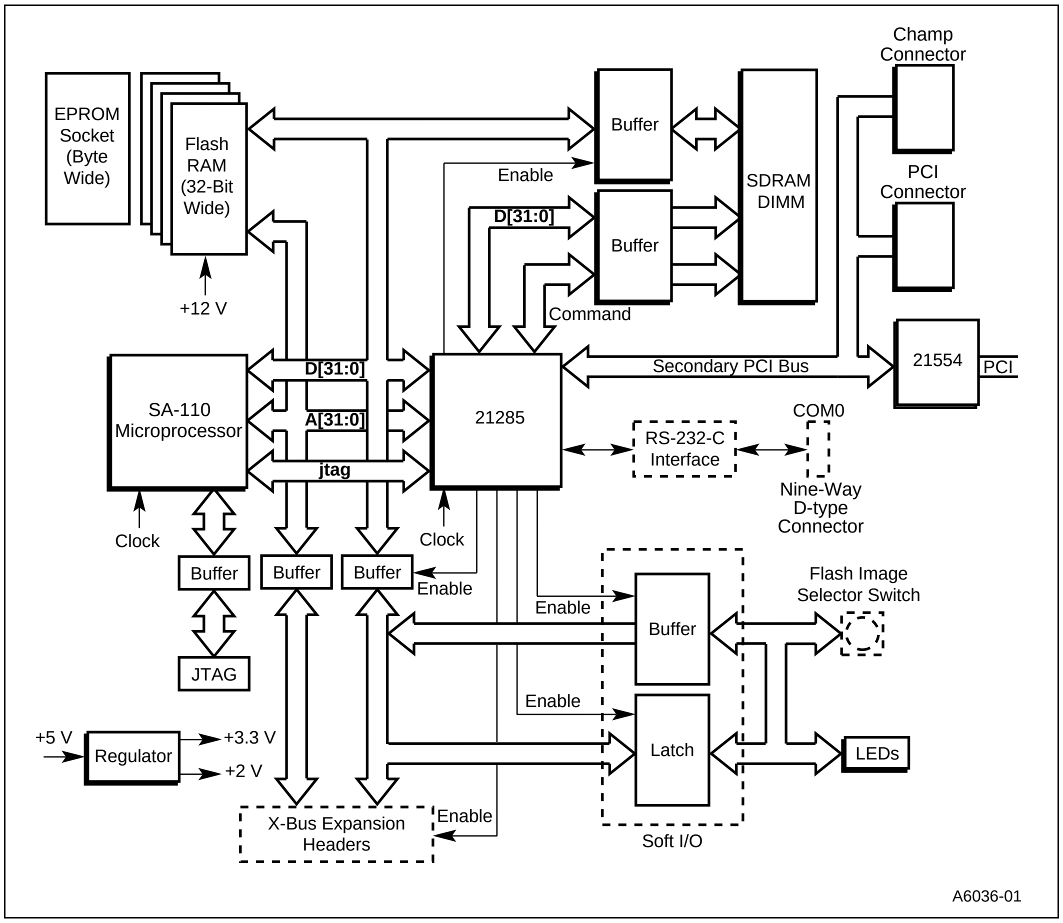


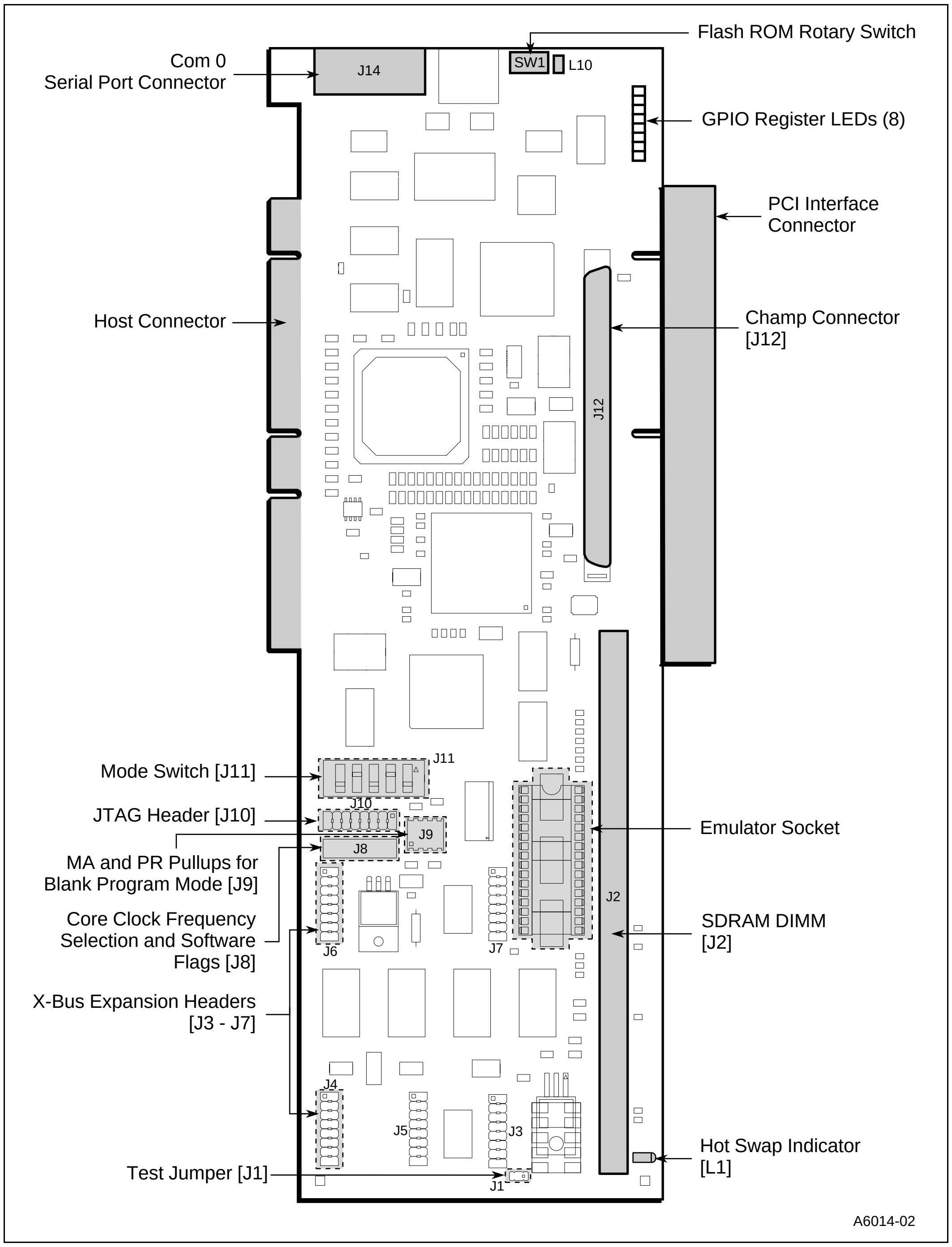
Table 1-1 lists a brief description of the connectors located on the SA-IOP.

Table 1-1. SA-IOP Connectors

Reference	Description
J1	Compact PCI hot swap test jumper
J2	SDRAM DIMM socket (up to 4 array DIMMs)
J3 through J7	X-Bus expansion headers (to connect low-speed ISA peripherals)
J8	Core clock frequency selection and software flags
J9	21554 PR and 21285 MA reset conditions and mode selection
J10	JTAG connector (test access to I/O pins of device)
J11	Mode switches (Blank Program, Flash, Emulator modes)
J12	120-pin Champ* connector (for PCI daughter card)
J14	COM 0 serial port connector (RS-232 console port)
J101	PCI interface connector (for peripherals)

Figure 1-2 shows the physical diagram of the board.

Figure 1-2. SA-IOP Physical Layout



1.2 Manual Conventions

Certain manual conventions are used to convey specific types of information. Table 1-2 lists a description of these conventions.

Table 1-2. Manual Conventions

Convention	Description
`Courier New`	Shows items displayed on the screen.
`Courier New Italics`	Shows EDU source code, EDU or DOS command lines.

1.3 About This Manual

This document is a single point-of-reference, both for configuring and using the board and for engineers who intend to copy parts of its design. Table 1-3 lists a brief description of the contents of this manual.

Table 1-3. Manual Contents

Title	Contents
Chapter 1, “Introduction”	Describes the SA-IOP functional and physical features.
Chapter 2, “Getting Started”	Describes jumper configurations, software tools and installation, creating and downloading executables.
Chapter 3, “Hardware Reference”	Describes the memory subsystem, I/O subsystem, PCI interface, X-Bus interface, and board reset operation.
Chapter 4, “SA-110 Microprocessor Overview”	Describes the features and operation of the SA-110 microprocessor on the SA-IOP.
Chapter 5, “Software Initialization and Configuration”	Describes the software initialization and configuration of the system hardware after a power-up or reset.
Appendix A, “Schematics”	Contains a complete set of schematics for the SA-IOP.
Support, Products, and Documentation	Contains technical support and ordering information.

The following Intel documentation may be useful:

- *21285 Core Logic for SA-110 Microprocessor Datasheet*
- *21554 PCI-to-PCI Bridge for Embedded Applications Hardware Reference Manual*
- *Getting Started with the 21554 Embedded PCI-to-PCI Bridge: An Application Note*
- *Using a Serial ROM with the 21554 Embedded PCI-to-PCI Bridge: An Application Note*
- *StrongARM™** SA-110 Microprocessor Brief Datasheet*
- *ARM™** V4 Architecture Reference Manual*
- *StrongARM™** SA-110 Microprocessor Technical Reference Manual*

Getting Started 2

This chapter provides instructions for installing the SA-IOP in a host system, configuring the jumpers properly, flashing the onboard ROM, and downloading and executing programs using either Wind River System IxWorks* or the Embedded Debug Utility (EDU).

2.1 Installation of the EB554 in a Host System

When installing the EB554 for the first time, visually inspect the board to verify that it was not damaged during shipping. If there are physical defects, return the board for replacement. The EB554 is a full length PCI board and requires a system slot free of obstructions. If the top PCI slot is used for DDM development, the extended height of the board with an add-in card will be higher than that specified in the *PCI Local Bus Specification, Rev. 2.1* and so the PC cover must remain off.

2.2 Jumper Configuration on the SA-IOP

The SA-IOP has two sets of jumpers used for setting the core clock frequency of the SA-110 microprocessor and specific features of the SA-110 microprocessor and 21554. In addition, a five pole switch is used for configuring the SA-IOP into one of three modes. The following sections provide detailed descriptions of these jumper and switch settings.

2.2.1 CPU Core Clock Frequency Selection

Figure 2-1 shows the header links that determine the SA-110 microprocessor core clock frequency. Table 2-1 provides the jumper information for specific frequencies. When a link is inserted, the corresponding CCCFG pin is grounded. The EB554 default core clock frequency is set at 228.1 MHz which is the maximum limit of the 233 MHz SA-110 microprocessor when using a 3.68 MHz input clock.

Figure 2-1. J9 Default Jumper Configuration

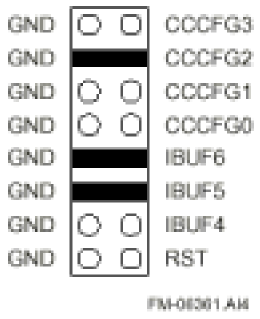


Table 2-1. J9 Core Clock Frequency Selection Jumpers

CCCFG	CCLK Frequency (3.68 MHz Input)	Unit
0	88.3	MHz
1	95.6	MHz
2	99.4	MHz
3	106.7	MHz
4	143.5	MHz
5	150.9	MHz
6	161.9	MHz
7	169.3	MHz
8	191.3	MHz
9	202.4	MHz
10	213.4	MHz
11	228.1	MHz
12	242.8	MHz
13	257.6	MHz
14	276.0	MHz
15	287.0	MHz

2.2.2 J11 Mode Switch

The J11 mode five-pole switch sets the EB554 to one of three modes:

- Flash
- Emulator
- Blank program

In addition, one switch setting can either enable or disable the primary lockout bit of the EB554. When enabled, it locks access by the host until the subsystem processor or serial ROM preload clears the primary lockout bit.

Figure 2-2 shows the basic options on the J11 switch. Figure 2-3 shows the switch settings required for the three modes.

Note: The shipped EB554 board is configured so the top jumper on J9 sets the board to either normal or blank programming mode. The top switch of the J11 five-pole switch is disabled. To enable this switch to set the board to either normal or blank programming mode, install a low ohm resistor (<50 Ω) in R350, on the back side of the board. In this configuration, the top jumper on J9 must be removed.

Figure 2-2. J11 Switch Options

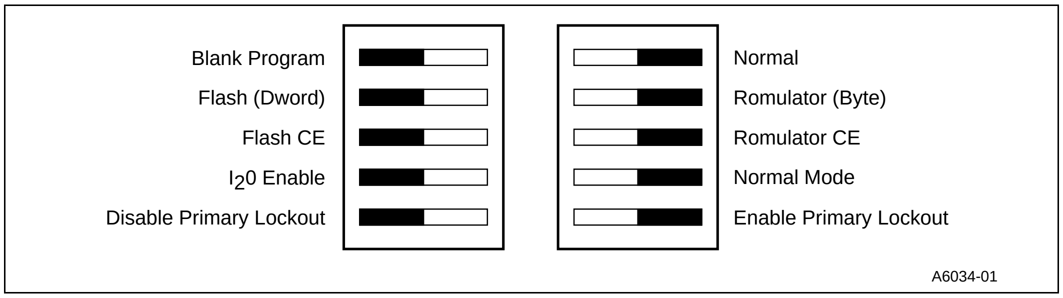


Figure 2-3. J11 Switch Mode Settings

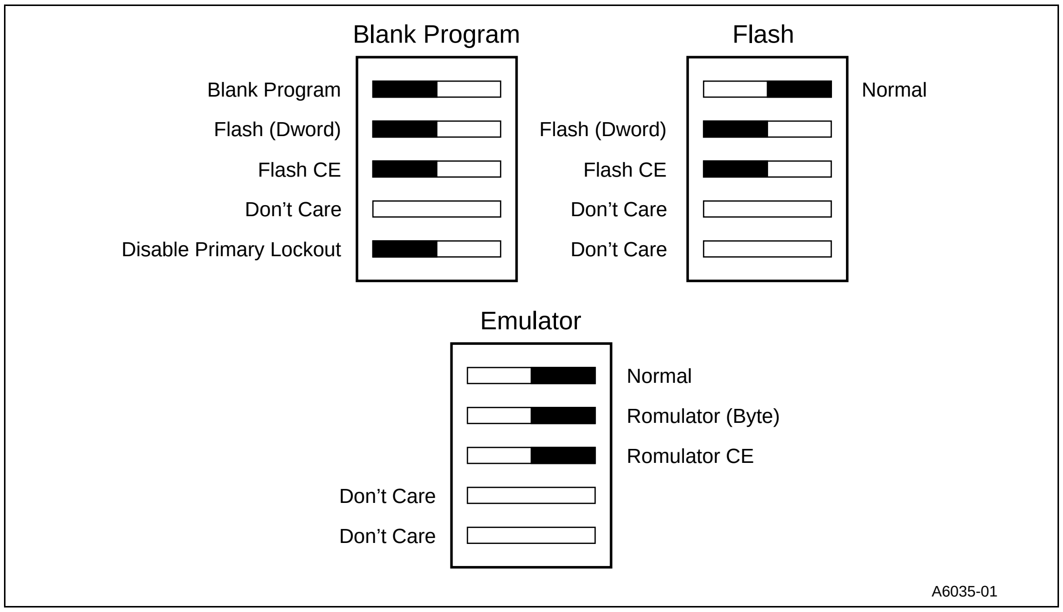
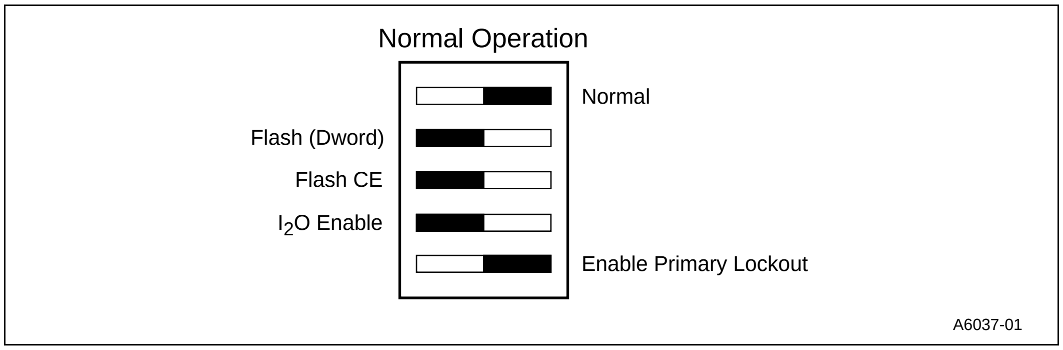


Figure 2-4 shows the typical EB554 switch configuration set for normal operation.

Figure 2-4. Typical Switch Configuration in Normal Operation



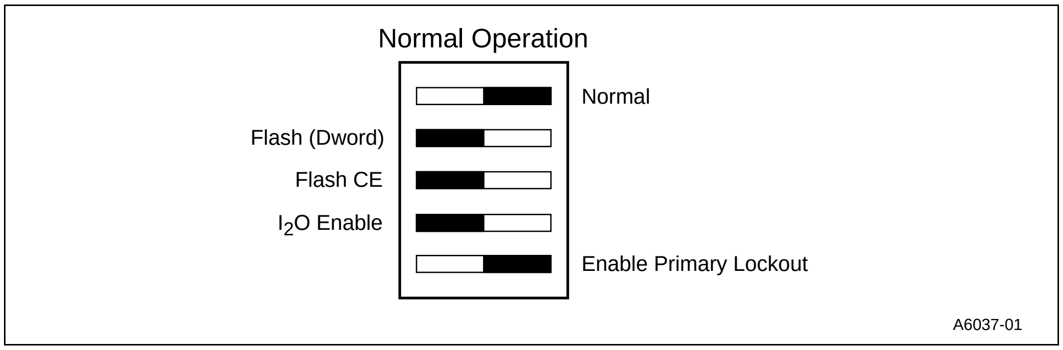
2.2.3 J9 Settings for 21554 and SA-110 microprocessor Reset Conditions

J9 is a 3x4 header connector that is used for setting the SA-110 microprocessor MA and 21554 PR_AD signals at specific levels during reset. The board is typically configured so that the top jumper of J9 controls the blank program mode. In addition, J9 can perform the following functions:

- Set the SA-110 microprocessor in test mode (MA2)
- Set the 21554 in asynchronous or synchronous mode (PR_AD4)
- Set the serial ROM present or not present (PR_AD2)

Figure 2-5 shows the J9 header connections. It also shows the typical J9 header settings in operating mode.

Figure 2-5. J9 Header Connections



2.3 Verify Functionality of SA-IOP

To verify the functionality of the EB554 board perform the following steps:

1. Connect an RS-232 cable from the 9-pin D-sub connector on the board to a free serial port on a terminal or terminal emulator.
2. Configure the terminal emulator to operate at 9600 baud, 8-bit data, 1 stop bit, no parity, and no flow control.
3. Set the rotary switch to location 0 so that the EDU boots and verify that the top jumper of J9 is set for normal operating mode.
4. Apply power to the system and observe that two of the LEDs light on the board (displaying binary pattern 00000101b).

In addition, the terminal should display the following message:

```
StrongARM™** IPL V1.1 (RAM), chipId: 0x4401a103
Initializing data sections
Branching to C code

Embedded Debug Utility (EDU): Target System [DBEB-SA110:4:SPD]
CPU Speed: 233 Mhz, Build Date: Sep 03 1998, 08:48:25
Load Address: 0x100000, Memory Size: 0x02000000 (32 Mbytes)

EDU>
```

If this response is not observed, check the following:

- If the LEDs are behaving correctly but the terminal does not display the proper message, check the terminal settings and serial cable connections.
- Check the jumper settings outlined in Section 2.2 to confirm that the board is configured properly.
- Check the rotary switch setting and confirm that it is on block 0. If so, try reinstalling EDU into this location.

If the board boots properly, it can be functionally verified by running the tests described in the Section 2.4.2.4.

2.4 Software Tools and Utilities

Configuration software tools and flash ROM management utilities are provided with the SA-IOP. The programs included in the onboard flash are a StongARM console (EDU) and the IxWorks RTOS for I_2O based applications. Please refer to the WindRiver document included with this package for additional information on IxWorks. A license agreement is included in the kit. Also, three MS-DOS* utilities are provided for updating the SA-IOP evaluation board flash ROMs and serial ROM (SROM).

2.4.1 IxWorks Software Development Toolset

IxWorks is a complete toolset featuring an integrated development environment including a compiler, assembler, linker, and debugger. It also features a real-time operating system.

2.4.1.1 IxWorks Real-Time Operating System

The EB554 evaluation board has Wind River Systems, Inc. IxWorks located in its onboard flash (flash block 8). IxWorks provides for the elements of the I_2O standard: an event-driven driver framework, host message protocols, and executive modules for configuration and control. IxWorks also allows for the writing of drivers and provides NOS-to-driver independence. Tornado for I_2O provides a visual environment for building, testing, and debugging of I_2O drivers.

2.4.1.2 TORNADO Build Tools

TORNADO for I_2O provides a visual environment for building, testing and debugging device driver modules (DDMs). It provides a complete development tool chain which includes a compiler, assembler, linker and binary utilities.

2.4.1.3 TORNADO Test and Debug Tools

TORNADO for I_2O includes the dynamic loader, CrossWind* debugger, and Windsh* interactive shell. The dynamic loader is used for interactive loading, testing and replacement of individual object modules.

CrossWind allows the debugging of I_2O drivers by setting breakpoints on desired areas. The windows display valuable information including source code, registers, locals, stack frame, and memory.

Windsh enables communication with the EB554 through an RS-232 serial port. The EB554 baud rate is generated from the FCLK frequency. Based on a frequency of 60 MHz, different baud rates can be selected ranging from 225 bps to 115.2 Kbps. The shell is typically used to:

- Monitor DDMs
- Examine hardware registers
- Send and receive driver messages
- Run automated I_2O test suites

2.4.2 Embedded Debug Utility (EDU)

The following subsections describe the commands that are available from the EDU command line. Each command is interpreted at the command line and the specific routine is called, carrying out each request when parsed. Most commands require parameters to be specified on the command line when called.

2.4.2.1 EDU Access Commands

The EDU provides many different commands that allow the user to access the entire 4 GB address space on the SA-110 microprocessor evaluation board. In addition, there are commands that are focused on more specific address ranges, so that the user does not have to type in the entire 32-bit address. For instance, if the user wants to write to the 21285 registers, then the fbr command will expect the address parameter to be an offset from the start of the 21285 register space (i.e., 90h instead of 4200 0090h).

The following list defines the available access commands:

- Local memory display and change
- Entire system space access
- PCI memory and I/O space
- PCI configuration space
- 21285 CSR register
- X-Bus (serial ports, SIO, LEDs, switch settings)

2.4.2.2 EDU Kernel Commands

Since the EDU is a small kernel, there are commands available that modify the internal state of the EDU runtime environment. For instance, the user can select I and D cache operation and change the state of the processor from the command line. This allows the user to preselect processor settings before running specific tests, fine tuning the environment.

2.4.2.3 EDU Utility Commands

At the command line, the user has a number of utilities available. These utilities provide a range of useful features including:

- SA-110 microprocessor and core logic
- SDRAM size and initialization
- PCI bus initialization
- PCI device resource mapping and allocation
- PCI bus scanning
- Flash ROM programming (next version)
- Dynamic communication port/baud selection
- Intellec hex file support allowing image loading via serial port

2.4.2.4 EDU Testing Commands

The purpose of the EDU is to allow complete testing of SA-110 microprocessor PCI evaluation boards and backplanes. Although many of the access commands provide features that allow tight repetitive looping for instrumentation and analysis of specific suspect transactions, this subsection lists the simple tests available in the production version of EDU:

- Serial port testing at all baud rates (may require loopback)
- DMA engine testing
- Memory testing
- PCI transactions (may require special target hardware)

2.4.2.5 EDU Command Description

Table 2-2 lists a description of the EDU commands.

Table 2-2. EDU Commands (Sheet 1 of 2)

Command	Description	Parameters	Example
p	Print memory contents segment	(Start address)	p 0
c	Change memory contents	(Start address)	c 0
sysr[bwd]	Read physical system space	Physical address	sysrd 4200 0000h
sysw[bwd]	Write physical system space	Physical address Data to write	sysrd 4200 0000h a5a4 a3a2
memr[bwd]	PCI memory read	PCI memory address	memrd 10 0000h
memw[bwd]	PCI memory read	PCI memory address Data to write	memwd 10 0000h a5a4 a3a2
ior[bwd]	PCI I/O read	PCI IO address	iorw 300
iow[bwd]	PCI I/O write	PCI IO address	iowb 300 a5
cfgr[bwd]	PCI configuration space read	Bus device register	cfgrd 0 5 30
cfgw[bwd]	PCI configuration space write	Bus device register data to write	cfgwd 0 5 10 ffff ffffh
xbusr	X-bus device 2 read (com ports)	Port	xbusr 3f9
xbusw	X-bus device 2 write	Port data	xbusw 3f9 a5
fbr	21554 register read	21554 register	fbr 90
fbw	21554 register write	21554 register Data to write	fbw 90 8c
ion	ICache enable	—	Ion
ioff	ICache disable	—	Ioff
don	Dcache/MMU enable. Page table base = C 0000h	—	don
doff	Dcache/MMU disable	—	doff
fill	Fill memory range with data pattern	Start address Stop address Pattern	fill 100000h 200000h a5a4 a3a2
ctty	Change default serial port	Port	ctty2

Table 2-2. EDU Commands (Sheet 2 of 2)

Command	Description	Parameters	Example
setbaud	Change serial port baud rate	Port—new baud rate	setbaud 2 9600
t	Invoke test command. There are a number of tests supported by the EDU. Some tests require host synchronization software to be running in tandem. For more information, refer to Section 2.4.2.	Test	t dma
check	Generate a checksum for a region of specified memory	Start address End address	check 30 0000h 40 0000h
pcimap	Map and enable all devices on the PCI bus	—	pcimap
pciinit	Initialize PCI bus and assign resources to requesting devices	—	pciinit
pscan	Scan the PCI bus for devices	—	pscan
load@	Sets the default image loading address	(Physical memory address)	load@ 10 0000h
dis	SA-110 microprocessor native disassembler	Start address	dis 10 0000h
blockcopy	Copy a region of memory	Start address End address Destination address	blockcopy 1000h 2000h 400000h
rpsr	Read SA-110 microprocessor status register	—	rpsr
xload	Load Intellec-Hex file image to where load@ is set to. This will default to 10 0000h. Use raw ascii on sending protocol.	—	xload
wpsr	Write SA-110 microprocessor status register	Data to write	wpsr 1000

2.5 EB554 Software Installation

The EB554 is supplied with EDU loaded into the SA-110 microprocessor flash block 0 and IxWorks loaded into flash block 8. In addition, both the serial ROM and parallel ROM of the 21554 have been flashed. The user has the option to install additional code in these three ROM locations using one of the MS-DOS based utilities described in the following sections. All three utilities require the system to be running in MS-DOS mode with dos4gw.exe 32-bit DOS extender defined in the path. (Not a DOS window running under either Microsoft* Windows 9x or NT).

2.5.1 FBFLASH Utility

The Fbflash utility is used to update the flash ROMs on the EB554 evaluation board with new ROM binary images. Fbflash runs under MS-DOS and requires the dos4gw.exe 32-bit DOS extender (provided). The EB554 is required to be setup for blank programming mode before the Fbflash utility can be successfully run. In addition, the switch selections described in Section 2.2.2 must be set for flash mode, not romulator mode.

The EB554 has 4 MB of flash ROM available. This is partitioned into the address map listed in Table 2-3. Each block is selected to be run by using the rotary switch on the bulkhead.

Table 2-3. Flash ROM Contents

Block	Physical Address	Contents	Comments
0	4100 0000h	IPL and embedded debug utility	Boot block
1	4104 0000h	Manufacturing diagnostics	To be installed
2	4108 0000h	Available	—
3	410C 0000h	Available	—
4	4110 0000h	Available	—
5	4114 0000h	Available	—
6	4118 0000h	Available	—
7	411C 0000h	Available	—
8	4120 0000h	IxWorks with RAM disk HDM	IxWorks beta release
9	4124 0000h	IxWorks storage	IxWorks reserved
10	4128 0000h	IxWorks storage	IxWorks reserved
11	412C 0000h	IxWorks storage	IxWorks reserved
12	4130 0000h	IxWorks storage	IxWorks reserved
13	4134 0000h	IxWorks storage	IxWorks reserved
14	4138 0000h	IxWorks storage	IxWorks reserved
15	413C 0000h	IxWorks storage	IxWorks reserved

2.5.1.1 Utility Usage

Fbflash utility uses parameter switches to blast images into the different blocks of the flash ROMs. The flash ROMs on the EB554 are setup as DWORD wide access where each one of the four (1 MB) devices is a byte in the DWORD access. This provides 16 blocks (256 KB each) of flash ROM storage. If the image is greater than 256 KB, fbflash automatically erases and updates the appropriate flash blocks. To update any one of the blocks (or multiple blocks if image is greater than 256 KB) the command is as follows:

```
fbflash /b(block number 0-15) image.ext
```

For example, if the user wanted to update flash block 8 with a new IxWorks binary image, first verify that the image is linked to run at block 8 (physical address 4120 0000h). Put the EB554 into blank programming mode (Section 2.2.2). Then boot MS-DOS and execute the following command:

```
fbflash /b8 ixworks.bin
```

During the flash update, the rotary jumper can be in any position. Put the EB554 back into Flash mode (Section 2.2.2). Verify that the rotary jumper is set to the proper image selector before rebooting to get the correct image. If the LEDs start blinking fast with a steady pattern, it means that the image is either not linked to the address (block) that was selected, or the rotary selector is not at the correct selection.

To completely erase the entire 4 MB of flash, the user executes the following command:

```
fbflash /e
```

This will erase all blocks in the flash ROMs. If the initial program loader (EDU) requires updating, the user should execute the following command:

```
fbflash /b0 edufbeb.bin
```

The user will be prompted to ensure that modifying the boot block is *really* what the user intends to do.

2.5.2 DBFLASH Utility

Dbflash.exe is an MS-DOS based program that allows the flash ROM attached to the 21554 to be erased and updated with new images. When dbflash.exe is run on a system that has a 21554 installed on the PCI bus, the program scans all of the PCI buses looking for the 21554 component. When found, the program identifies the 21554 PCI location and starts the update process that was selected on the command line.

2.5.2.1 Utility Usage

Verify that both dbflash.exe and the 32-bit DOS extender dos4gw.exe are in the same directory or both are in the environment path. Dbflash.exe requires parameters to tell it what to do. A typical flash programming update requires the user to specify the flash block to update and the new image to use.

```
Dbflash /b0 NewRomImage.bin
```

This example flashes image 'NewRomImage.bin' into block 0 of the 21554 expansion ROM. During the next boot of the PC, the BIOS finds this image in the ROM and if it has a PCI compliant expansion ROM header, the image is loaded and executed by the system BIOS during POST. For more information on how this works, refer to the *PCI Local Bus Specification, Revision 2.1*. Table 2-4 lists an example of flashing a new ROM image.

Table 2-4. Flashing a new ROM Image

Command	Description	Comments
dbflash /e	Erase entire contents of flash ROM	Erase all blocks
dbflash /bx image.bin	Program block x with image.bin	If image is larger than 1 block, the program continues into the next block until entire image is loaded.

2.5.3 MKSROM Program

The 21554 device allows for initial register and state values to be loaded upon device reset. This provides a mechanism for the device to change some internal default values without any modifications to the device itself. Mksrom.exe runs under MS-DOS and requires the dos4gw.exe 32-bit DOS extender (provided).

The MKSROM program allows the serial ROM connected to the 21554 device to be updated with new SROM values. MKSROM reads in a previously constructed file (see *Using a Serial ROM with the 21554 Embedded PCI-to-PCI Bridge: An Application Note*) and writes those values into the SROM.

2.5.3.1 Program Usage

Typically mksrom.exe reads in a file (filename.ext) as a parameter and parses the file to determine the values to program into each location of the SROM. If no parameter is used, then the program searches for the first 21554 device and displays its contents on the screen.

```
Mksrom [ filename.ext]
```

2.5.3.2 SROM file

The SROM file that is loaded requires specific configuration syntax. Table 2-5 provides a description of the symbols used for the required syntax.

Table 2-5. Configuration Syntax

Symbol	Description
;	Comment
[	Start of ROM data
]	End of ROM data
:	Start of valid address data line

The following is an example that programs SROM addresses 0, 1, 2, and 1A with data 80, 00, 00, 08 respectively:

```
;
; SROM Data file
;
[
:0 80
:1 00
:2 00
:1A 08
]
```

2.6 Creating and Downloading Executables

When running IxWorks on the SA-IOP, refer to Wind River TORNADO documentation when downloading code. If running EDU, XLOAD downloads code to the board using the serial port.

2.6.1 Serial Image Load

This section describes the simple image development environment. EDU allows the user to load images to SA-110 microprocessor memory via the serial port or the PCI bus. Loading from the serial port requires the user to give an EDU command to get ready for the image file, and to load the file at a user defined location. Also, the image must be in Intellec Hex Format (IHF). Loading an image via the PCI bus does not require cooperation from EDU, or any format, but some precautions must be addressed first, for a successful image load operation.

To load an image into SA-110 microprocessor memory from the serial port under EDU, the image must be in IHF and the target (EDU running on EB554) must be connected to a host running a terminal emulator, capable of uploading files in ASCII format. Programs such as Hyperterm or Procomm work fine. The IHF allows EDU loader and any other supporting IHF loader to know specifics about the image. Also, all data is treated as ASCII text. Most developer toolkits provide a linker switch to generate this format. For example, with the following ARM™** SDT command:

```
armlink -ihf -o myfile.ihf -ro 0x100000 myfile.o
```

It links myfile.o to start at 10 0000h and it generates an IHF file called myfile.ihf. This is the format that EDU accepts for a serial download.

At the EDU prompt the user types:

```
load@
```

This displays the loading default address. If the user wants to load the image at a different address, refer to the command usage in Table 2-2.

Next, the user types:

```
xload
```

This starts the sequence to load the image.

```
EDU>> xload
Int-Hex File Loader, transferring to address 100000
start ASCII file transfer now...
```

On the host side, the user must be running a terminal emulator that allows uploading (send file) connected to the serial port on the EB554. Select upload (send file) making sure that ASCII mode is used and start the transfer. When the image is loaded, the EDU prompt is returned.

The user types:

```
g 100000
```

This instructs the SA-110 microprocessor to start execution at address 10 0000h. Usage of the LEDs and alternate serial ports are extremely useful debugging features.

2.6.2 Runtime Environment

The EDU target runtime configuration can consist of three possible environments.

- Standalone central function (on a bench, self powered)
- Standalone in available PCI backplane
- Target device installed in host generated PCI bus (that is, personal computer)

In any of these configurations, the EDU gives the user the option of selecting the default com port by listening to all ports and selecting the port where any key is pressed first. After booting the EDU, it displays the following banner and is ready for commands from the user (at this point, online help can be used):

```
Embedded Debug Utility (EDU): Target System [StrongARM™**-DBEB]
CPU Speed: 233 Mhz, Build Date: June 16, 1998
Load Address: 0x100000, Memory Size: 0x2000000 (32 Mbytes)
EDU>> h

help: help is available on the following commands,
for help on each command type 'h command<cr>'.

p[bwl].......print memory contents     c[bwl].......change memory contents
mem[rw][bwd].PCI memory access         io[rw][bwd]..PCI IO access
cfg[rw][bwd].PCI config access         sys[rw][bwd].system access
xbus[rw].....xbus access               fb[rw].......FB reg access
ion..........icache enable             ioff.........icache off
don..........dcache enable             doff.........dcache off
mmuon........mmu enable                mmuoff.......mmu disable
pciinit......init PCI registers        pcimap.......map PCI devices
db...........select 21554 commands     ctty.........set default user terminal
check........generate checksum         fill.........fill memory with pattern
blockcopy....copy memory block         g............run code not in svc mode
jtosvc.......run code in svc mode      wrv..........write/read/verify
ctty.........set default user term     pcireset.....reset PCI bus
setbaud......set tty baud rate         load@........set default load address
pscan........scan the PCI bus          df...........set EDU debugging flags
t............run EDU test              h(elp).......display this help screen
tmode........set testmode              status.......display EDU status
edu>>
```

Hints and Tips:

- Turn Icache on (**ion**) for better performance while running the memory tests
- Before any PCI access, run **pciinit** command so that PCI will be fully initialized
- Command line recall is available by using the up and down arrow keys
- Set default terminal line characteristics to 9600 baud, 8 bits, no parity, 1 stop bit

Hardware Reference 3

This chapter provides a technical description of the EB554 hardware. It should be read in conjunction with the EB554 schematic set (Appendix A). Both should provide a detailed understanding of the hardware which will be beneficial when designing a board based on the 21554, 21285, and SA-110 microprocessor.

3.1 Power Requirements

The integrated circuits on the SA-IOP use two voltage levels: 3.3 V and 5.0 V.

The circuitry that requires 3.3 V includes:

- SA-110 microprocessor (maximum 1 W)
- SDRAM DIMM (maximum 16 W)
- EPROM emulator socket

The circuitry that requires 3.3 V and is 5.0 V tolerant includes:

- 21285. This allows the 21285 to be used on universal cards and the oscillator to be interfaced directly (maximum 2.1 W).
- 21554 (maximum 1.5 W)
- Flash ROM (maximum 1 W, 5 devices)
- LVT buffers

The circuitry that requires 5.0 V includes:

- QS6611
- Oscillators
- JTAG
- PCI
- X-Bus Expansion

Signals that are generated with 5.0 V switching levels must be level converted before using them as inputs to the SA-110 microprocessor.

3.2 The Memory Subsystem

The EB554 provides synchronous DRAM (SDRAM) for its main memory and flash ROM for its boot path and non-volatile storage. It also supports the use of a plug-in ROM emulator to aid software debug. A serial EEPROM is used by the 21554 to load data into its configuration registers.

3.2.1 SDRAM Configuration

The SDRAM configuration usually contains multiple logical banks of memory (typically two or four) within a single SDRAM. The existence of these multiple banks is invisible to software but allows the memory controller to extract greater performance from the memories. To avoid confusion with these internal banks, the term array is used to describe a physical group of memory devices that share a common chip select and provide a 32-bit data path.

The 21285 supports four SDRAM arrays. The EB554 has a socket for one, 168-pin, 64-bit SDRAM DIMM. The 64-bit data path of the DIMM is treated as two separate arrays (since the 21285 requires a 32-bit data path). This leads to an unorthodox wiring scheme for the DIMM. Each array must use a single chip select line that is unique to that array. Table 3-1 shows the relationship between the chip select lines, the DQM lines, clock lines and the data byte for each combination:

Table 3-1. 4-Array Port SDRAM Configuration

Array	Byte	Chip Select	DQM	DQ Register	Clock
1	0	0	0	DQ[7:0]	0
	1	0	1	DQ[15:8]	1
	2	0	4	DQ[39:32]	2
	3	0	5	DQ[47:40]	3
2	0	2	2	DQ[23:16]	0
	1	2	3	DQ[31:24]	1
	2	2	6	DQ[55:48]	2
	3	2	7	DQ[63:56]	3
3	0	1	0	DQ[7:0]	0
	1	1	1	DQ[15:8]	1
	2	1	4	DQ[39:32]	2
	3	1	5	DQ[47:40]	3
4	0	3	2	DQ[23:16]	0
	1	3	3	DQ[31:24]	1
	2	3	6	DQ[55:48]	2
	3	3	7	DQ[63:56]	3

3.2.2 SDRAM Interface

The SDRAM interface consists of:

- Multiplexed address bus
- Data bus
- Command outputs
- Chip and byte selects
- SDRAM clocks

All these interface signals have relatively high switching frequencies with fast rise and fall times, and each signal drives a number of devices. To ensure that the maximum number of devices (that is, 16 SDRAMs on one DIMM) can be driven while maintaining the integrity of the signals, each of the signals is buffered and some also have a 33.2 Ω series termination resistor. The series termination resistors are placed close to the output of the driver so that there is less than 3 cm of etch between the output of the buffer and the resistor.

3.2.2.1 Multiplexed Address Bus

The multiplexed address bus provides the SDRAMs with multiplexed row and column data. A 13-bit bus (MA[12:0]) is provided by the 21285 signals and the DIMM has a 14-bit bus (MA[13:0]), so the most significant bit is tied to ground via a pull-down resistor. The bus is buffered by a 74LVT16244 and has a 33.2 Ω series termination resistor on each line.

3.2.2.2 Data Bus

The data bus is the buffered CPU data bus (BUF_D[31:0]). The data bus is not series terminated because it is a bidirectional bus.

3.2.2.3 Command Outputs

The command outputs (BUF_CMD[2:0]) from the SA-110 microprocessor generate command information to the SDRAMs. Each of the following lines are buffered through a 74LVT16244 and has a 33.2 Ω series termination resistor:

- BUF_CMD[0] is connected to /WE on the DIMM
- BUF_CMD[1] is connected to /CAS on the DIMM
- BUF_CMD[2] is connected to /RAS on the DIMM

The 21285 defines the command signals to be active-high, but their behavior is such that these signals can connect directly to the SDRAM /WE, /RAS and /CAS signals because no inversion is required.

3.2.2.4 Chip and Byte Selects

There are four chip selects (BUF_CS[3:0]**)** and four byte selects (BUF_DQM[3:0]). Each select is buffered through a 74LVT16244 and has a 33.2 Ω series termination resistor. Each chip select is used to select a single SDRAM array, so that a maximum of four arrays may be used in a system.

Byte selects (BUF_DQM[3:0]) are used to provide byte lane information to an SDRAM array. These signals allow each byte within a 32-bit Dword to be accessed, either individually or together. Each signal is buffered via a 74LVT16244 and has a 33.2 Ω series termination resistor.

3.2.2.5 SDRAM Clocks

The SDRAM clocks **(**FBG_SDCLK[3:0]) have matched lengths and impedances with each other and also with the 21285 SDRAM clock reference trace, FBG_FCLK_O. The overall length of the traces have been kept as short as possible to maintain signal integrity. The clock reference trace is used to minimize the effect of skew on the SDRAM clock traces, to give the 21285 a reference for all SDRAM transactions. Each of the lines has a series termination of 33.2 Ω. The 21285 SDRAM clock outputs have high-current drivers and must not be buffered externally; this would introduce too much skew into the SDRAM timing.

3.2.3 21285 Flash ROM

Non-volatile storage is provided by four bytewide 1MB flash ROMs, arranged to provide a 32-bit ROM path. This provides a total of 4 MB of ROM. The SA-110 microprocessor software can make a region of the ROM visible on the PCI bus so that it appears in PCI space as a PCI expansion ROM. The 21285 supports 8-bit, 16-bit, and 32-bit ROMs, but the SA-IOP design only allows the 32-bit mode to be used for accesses to the flash ROM. The flash ROM is divided into a number of separate blocks, which can be erased and reprogrammed independently using FBFLASH.EXE. During all ROM accesses, the 21285 drives output-enable on CPU_A{30] and write-enable on CPU_A[31], so these address lines are wired to the appropriate signals on the flash ROMs. A 12 V supply is required for programming the flash ROM. The 12 V source is from the primary PCI connector.

An EPROM socket (Section 3.2.5) is provided on the card for rapid program development using a ROM emulator. The 512K of the EPROM socket is mapped in place of the flash ROM but is byte wide rather than 32-bit wide. The socket should not be used in Word or Dword modes and it cannot be used when the flash ROM is active. When the 21285 is accessing the flash ROM the CPU address bus drivers are placed in a high impedance state by the 21285 asserting the ABE signal.

Two switches control the selection of flash ROM or the EPROM socket:

- J11 switch 3 routes FBG_ROM_CE_L to either the flash ROM or EPROM. The unselected signal line is held high by a pullup resistor.
- J11 switch 2 either pulls the signal FBG_MA4 up to select the EPROM socket or down to select the flash ROM.

Both switches must be changed together. To effect the new selection, the board must be powered off and then back on. When the EB554 is reset the flash ROM is also reset. This places the flash ROM in a known state at reset or power-up. Resetting the flash ROM halts any automated write/erase cycles that the flash ROM is performing. This avoids a read being made from flash ROM during an automated cycle. If that were to occur, the flash ROM would provide status information preventing the CPU from booting correctly.

3.2.4 21554 Flash ROM

The 21554 has its own parallel 1MB ROM based on an Intel 28F008 providing an 8-bit ROM path to the 21554. The ROM address is driven onto the 8-bit data bus in three consecutive cycles and latched by external octal D registers. The 21554 supports a PCI expansion ROM base address register (BAR) on its primary interface and can provide the PCI expansion ROM for the entire subsystem. If the local subsystem does not require a PCI expansion ROM, this BAR can be disabled. When the host completes the configuration of the primary PCI interface, the parallel ROM can be accessed by the host in the memory address range assigned in the PCI expansion ROM BAR. The secondary PCI interface can indirectly access the parallel ROM through the 21554 control and status registers.

3.2.5 EPROM Emulator

The EPROM emulator is a debugging tool that connects to a target as though it were an EPROM but it allows a fast download and a modification of code. The SA-IOP has a 32-pin, 0.6-inch DIL socket that can be used to connect a byte-wide 512 Kbyte EPROM emulator head. The socket provides 3.3 V power and can only be used with 3.3 V EPROM emulators. A conventional EPROM typically requires 5 V and cannot be installed in the socket.

Caution: Installing a 5 V emulator may result in damage to your SA-IOP. Jumpers on the SA-IOP are used to reconfigure the ROM width and disable the Flash ROM when using the emulator; access to the Flash ROM and the EPROM emulator socket are mutually exclusive.

When the SA-IOP is configured correctly, the 21285 makes SA-110 microprocessor and PCI accesses to the EPROM emulator appear as 32-bit accesses. The PROMJet* EPROM emulator, from Emutec*, has been used successfully on the SA-IOP.

3.2.6 21554 Serial ROM

The SA-IOP uses a Microchip* 93LC66 4 Kb serial ROM to interface with the 21554. This serial ROM can be used to preload data into the 21554 configuration registers with vendor specific values. It may also be used to support the vital product data (VPD) interface. The serial ROM can be accessed through 21554 CSR control.

3.3 I/O Subsystem

All local I/O (within the SA-IOP module) is performed as programmed I/O under the control of the SA-110. The I/O subsystem provides the following resources:

- An RS-232 console port (data leads only) is accessed by a 9-way D-type on the bulkhead. This is referred to as COM 0.
- An 8-bit I/O port used to control LEDs and read the state of jumpers and a switch.

The only other I/O facilities on the board are those provided by the 21285 itself.

3.3.1 Console Serial Port

The serial port of the EB554 is a general-purpose, full-duplex, universal asynchronous receiver-transmitter (UART) similar in functionality to the 16C550 UART. Its baud rate is based on the FCLK_IN frequency of the EB554 which is set at 60 MHz. At this frequency, it is capable of operation from 225 bps to 200 Kbps. Refer to the *21285 Core Logic for SA-110 Microprocessor Data Sheet* for additional information on configuring the UART.

3.3.2 GPIO Interface

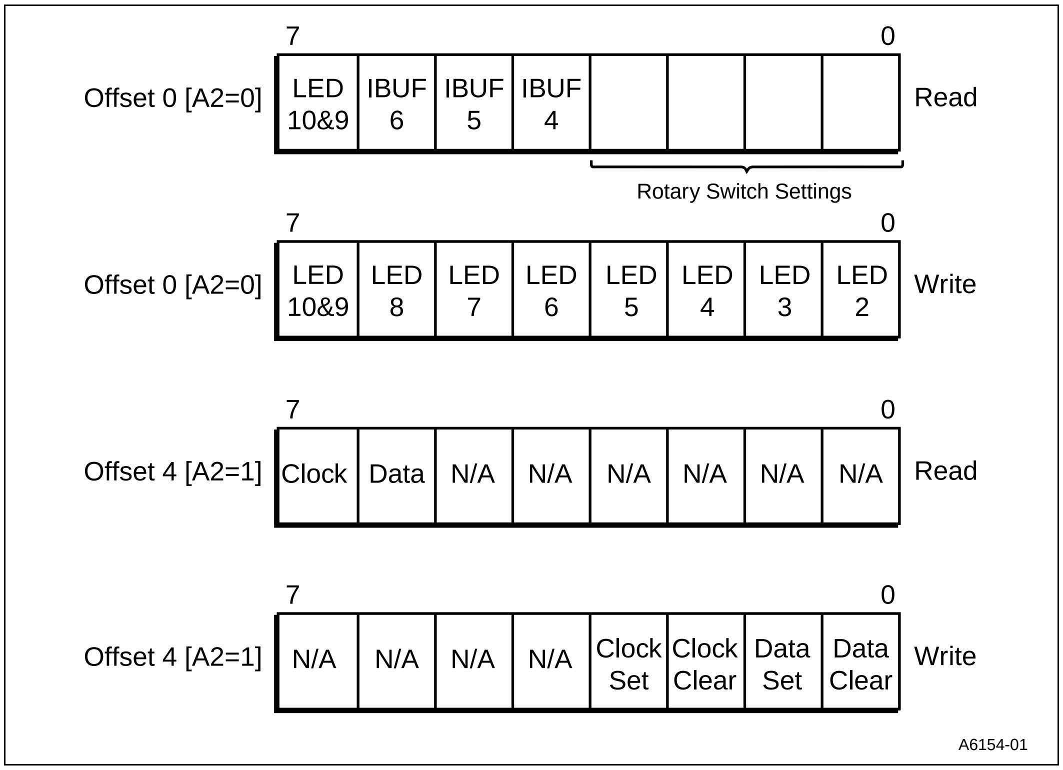
The 21285 XCS2 region is used to decode the onboard general purpose input output (GPIO) interface. This GPIO consists of an 8-bit output latch (74ABT541), and an 8-bit input port (74ABT377), and a programmable logic device (PALCE16V8). The PAL is used to divide the XCS2 region into two distinct areas by using address bit 2. If A2=0, the GPIO bits will write to the LEDs and read from various input mechanisms. If A2=1, the GPIO bits can be used to access the SDCLK and SDA signals of the SDRAM DIMM. Figure 3-1 shows the register bit assignments for these two configurations.

In the upper left corner of the EB554, there are eight LED indicators. These LEDs are activated by writing to the GPIO register. The most significant bit is on the left (toward the bulkhead). The bulkhead mounting bracket holds a 16-position switch and one LED indicator. The GPIO register allows software control of the LEDs, reading of the rotary switch, and reading three onboard jumpers.

The clock enable pins on both the 74ABT377 and 74ABT541 are driven by the PAL generated LED_ENABLE. If both A2 and XCS2 are low, the input and output ports have their clock enabled. During a write with the clock enable signal active, the outputs of the 74ABT377 are latched on the trailing (rising) edge of the clock pulse. The clock pin is connected to the write strobe FBG_XIOW_L. The latched states of the OBUF_D[7:0] drive the eight LEDs and OBUF_D[7] is looped back to the input port for a diagnostic test of the GPIO.

The 74ABT541, used for input, is an octal buffer with 3-state outputs and two output enable signals. Both enables must be asserted for the buffer to drive its outputs. The output enable signals are wired to the FBG_XCS2_L and FBG_XIOR_L lines. In addition to the output monitoring function of OBUF_D7, input lines LNK_SOFTI[3:0] hold the state of the Flash image selector switch, while IBUF_D[6:4] monitor jumpers on J9.

Figure 3-1. GPIO Register Bit Assignments



3.3.3 JTAG

The SA-110, 21285, and 21554 all contain joint test action group (JTAG) test ports that allow test access to the I/O pins of the device. The SA-IOP daisy-chains the four JTAG ports and provides access to the port through an 8x2 0.1-inch pitch header, J10. Table 3-2 lists the pinout for the JTAG header.

Table 3-2. JTAG Header J10 Pinout

Pin	Signal	Pin	Signal
1	Open	2	Open
3	VDD	4	GND
5	TRST_L	6	Open
7	TDI	8	GND
9	TMS	10	GND
11	TCK	12	GND
13	TDO	14	SRST_L
15	VDD	16	GND

3.3.4 I/O Expansion

The I/O capabilities of the SA-IOP can be expanded in two ways:

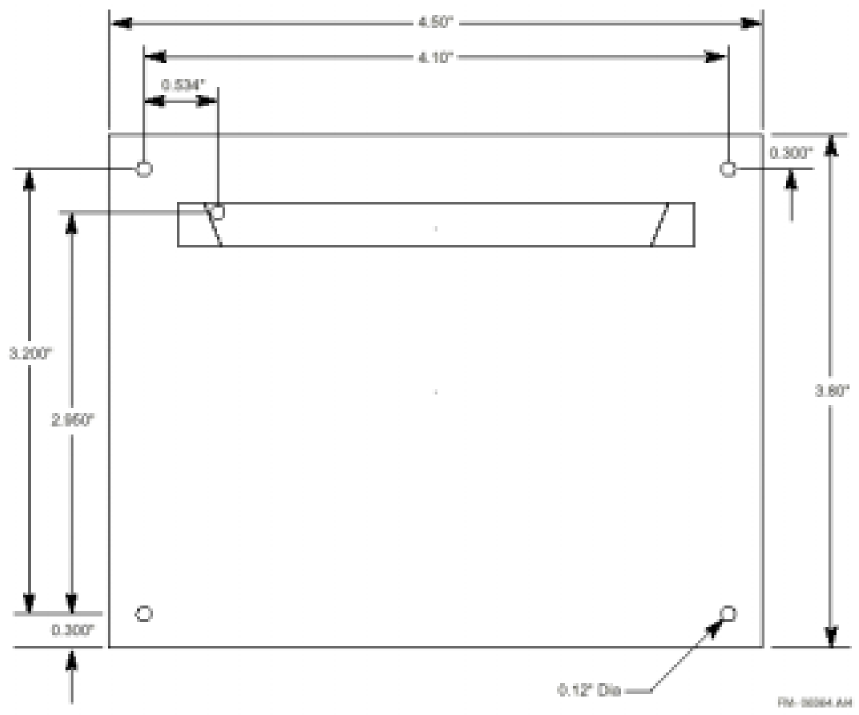
- Two PCI connectors which allow up to three additional PCI devices on the secondary bus.
- The expansion headers of the buffered 21285 X-Bus. These allow a small mezzanine PCB to be attached for connection to the X-Bus. The X-Bus provides a simple way of providing access to low-performance I/O peripherals.

3.3.4.1 PCI Connectors

The EB554 has two connectors on the secondary PCI bus for adding peripherals. All secondary devices are hidden from standard configuration accesses from the primary PCI bus by the 21554. The 21554 has one interrupt routed to the primary side. All devices on the secondary PCI bus have their interrupts routed to the IRQ inputs of the 21285.

The top edge connector is a standard 64-bit 5.0 V PCI bus option card slot. A 5.0 V or universal PCI option card can be plugged into this secondary slot. The Champ* connector is a 120-pin AMP connector (part number 176380-5) that enables a PCI daughter card to be placed on the EB554 with violating the PCI height requirements. It has two sets each of PCI_CLKS, IDSELs, INTs, REQs, and GNTs signals routed to it so that the daughter card can contain up to two PCI targets or masters. This connector is intended for users interested in developing modules for the EB554 that will fit into a closed system. The recommended physical dimensions of a Champ daughter card are shown in Figure 3-2. The pin connections to the Champ connector are listed in Table 3-3.

Figure 3-2. Physical Dimensions of Custom Champ Daughtercard



Note: This view is from the backside of the board. Components ree placed on the front side of the board so that the components from the daughter card and EB554 are sandwiched between the boards.

Table 3-3. Champ Connector Pinout

Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
1	S_RST#	25	S_AD24	49	S_AD9	73	GND	97	S_DEVSEL#
2	+12 V	26	S_GNT#	50	S_C/BE0#	74	NC[1]	98	GND
3	S_TMS	27	3.3 V	51	3.3 V	75	GND	99	S_LOCK#
4	S_TDI	28	S_AD22	52	S_AD6	76	NC[1]	100	S_PERR#
5	+ 5 V	29	S_AD20	53	S_AD4	77	GND	101	3.3 V
6	NC[a]	30	GND	54	GND	78	NC[1]	102	S_SERR#
7	S_INTC#	31	S_AD18	55	S_AD2	79	+5 V	103	3.3 V
8	+5 V	32	S_AD16	56	S_AD0	80	S_AD31	104	S/CBE1#
9	SCLK_2	33	3.3 V	57	+5 V	81	S_AD29	105	S_AD14
10	+5 V	34	S_FRAME#	58	NC[1]	82	GND	106	GND
11	SCLK_3	35	GND	59	+5 V	83	S_AD27	107	S_AD12
12	GND	36	S_TRDY#	60	+5 V	84	S_AD25	108	S_AD10
13	GND	37	GND	61	-12 V	85	3.3 V	109	GND
14	IDSEL3	38	S_STOP#	62	S_TCK	86	S_C/BE3#	110	S_AD8
15	S_RST#	39	3.3 V	63	GND	87	S_AD23	111	S_AD7
16	+5 V	40	NC1	64	TDO	88	GND	112	3.3 V
17	IDSEL2	41	NC1	65	+5 V	89	S_AD21	113	S_AD5
18	GND	42	GND	66	+5 V	90	S_AD19	114	S_AD3
19	S_REQ2#	43	S_PAR	67	NC[1]	91	3.3 V	115	GND
20	S_AD30	44	S_AD15	68	S_INTD#	92	S_AD17	116	S_AD1
21	3.3 V	45	3.3 V	69	S_REQ3#	93	S/CBE2#	117	+5 V
22	S_AD28	46	S_AD13	70	NC[1]	94	GND	118	NC[1]
23	S_AD26	47	S_AD11	71	S_GNT3#	95	S_IRDY#	119	+5 V
24	GND	48	GND	72	GND	96	3.3 V	120	+5 V

a. NC defines no connection.

3.3.4.2 X-Bus Interface

The X-Bus interface allows the connection of low speed ISA style peripherals (that is, UARTs, nonvolatile RAM, I^{2}C, and real-time clock) to the EB554. It provides a 16-bit data bus, XBUF_BUF_D[15:0], and a 10-bit address bus, BUF_A[11:2]. Control signals are provided for reset, read and write strobes, buffer direction control, and chip selects. The bus is unclocked and does not provide a means to stall the bus.

3.3.4.3 X-Bus Expansion Headers

The X-Bus expansion provides the X-Bus data, address, and control signals to five 2x8, 0.1-inch pitch headers. It also provides 5.0 V and ground to the header. The signals provided are:

- BUF_A[10:2]
- BUF_BE[1:0]
- XBUF_BUF_D[15:0]
- FLASH_RST
- FLASH_RST_L
- XD_WREN_L
- FBG_XIOR_L
- FBG_XIOW_L
- FBG_XCS2
- XBUS_XCS1_L
- XBUS_XCS0_IRQ_L
- VDD (5.0 V)
- GND (0.0 V)

3.4 Clocks, Arbiter, and Interrupts

This section provides a description of the EB554 clocks, PCI bus arbiter, and interrupts.

3.4.1 Clocks

The SA-IOP uses three oscillators:

- **3.6864 oscillator**. This oscillator drives the SA-110 phase-locked loop (PLL).
- **33 MHz oscillator**. This oscillator can be used to provide the secondary PCI clock when the 21554 is operating in asynchronous mode.
- **60 MHz oscillator**. This oscillator provides the oscillator input clock for the 21285. The 21285 buffers and redrives this clock to generate the SA-110 bus clock, the SDRAM clocks, and the 21285 feedback clock (FCLK). The local buses and the majority of the 21285's internal logic run synchronously at this clock frequency.

3.4.2 PCI Bus Arbiter

The EB554 is configured so that the internal arbiter of the 21554 performs the secondary bus arbitration. This arbiter supports a programmable 2-level rotating algorithm. For more information on the PCI arbitration, refer to the *PCI-to-PCI Bridge for Embedded Applications 21554 Hardware Reference Manual*.

3.4.3 Interrupt and IDSEL Routing

The primary interrupt of the 21554 is routed to the INTA signal line on the 64-bit primary PCI connector. All of the subsystem interrupts are routed to the IRQ lines of the 21285. The interrupt routing and corresponding IDSEL lines are provided in Table 3-4.

Table 3-4. IRQ and IDSEL Routing

Device	21285	IDSEL Line
21554	IRQ_IN0	S_AD20
21143	IRQ_IN1	S_AD21
Champ Device No. 1	IRQ_IN1	S_AD25
Champ Device No. 2	IRQ_IN3	S_AD23
PCI Edge Connector	IRQ_IN3 (INTA)	S_AD27

In the EB554 default configuration, IRQ_IN2 of the 21285 is connected to the 3.6864 MHz oscillator to provide a clock for the timer. If R355 is removed, and R356 installed onto the board, this IRQ pin is connected to S_INTC.

3.5 Board Reset

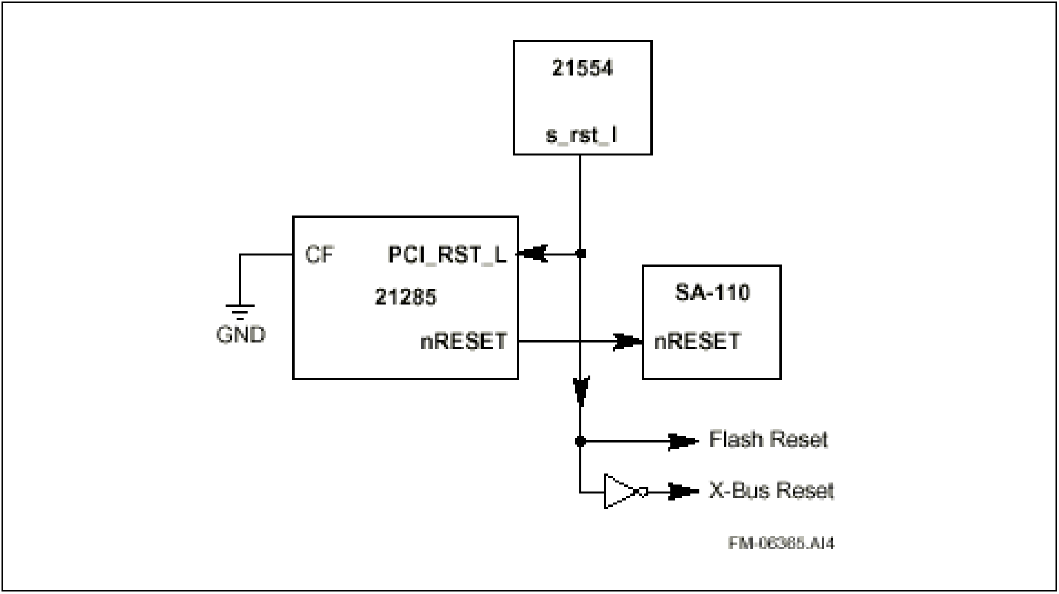
Reset on the EB554 is performed by a PCI reset only. S_RST_L is driven from the 21554 and acts as a PCI reset for the secondary bus. The 21554 asserts S_RST_L when one of the following conditions is met:

- Signal P_RST_L is asserted
- Write a 1 followed by a 0 to the secondary reset bit of the reset control register
- Write a 1 to the chip reset bit of the reset control register

The Flash ROM, X-Bus interface, and all secondary PCI add-in cards are reset directly by S_RST_L but the SA-110 microprocessor is reset by an nRESET signal generated by the 21285. This can be used to hold the CPU in reset while the Flash ROM is being programmed.

Figure 3-3 shows the routing of the reset signals.

Figure 3-3. Board Reset Circuitry

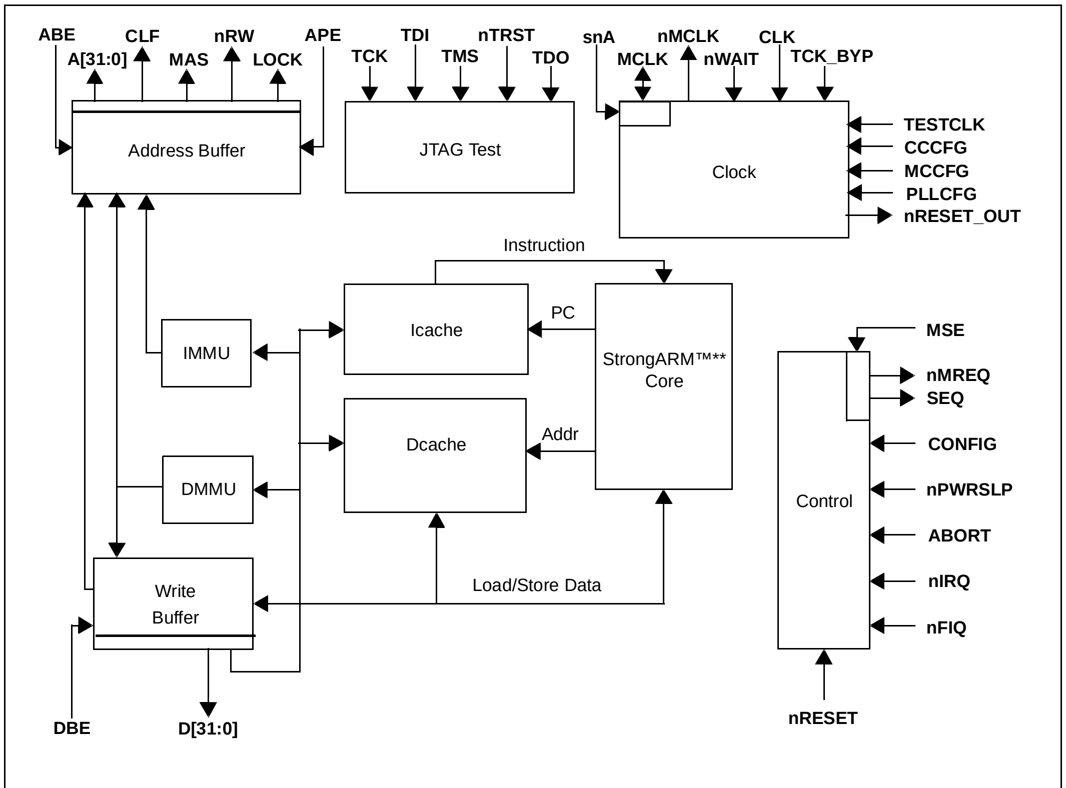


SA-110 Microprocessor Overview 4

This chapter describes the features and operation of the SA-110 microprocessor on the EB554. For more detail on the SA-110 microprocessor and 21285 refer to the *StrongARM™** SA-110 Technical Reference Manual* and the *21285 Core Logic for SA-110 Microprocessor Data Sheet*.

Figure 4-1 shows the block diagram of the SA-110 microprocessor.

Figure 4-1. SA-110 Block Diagram



** StrongARM™ is a trademark of Advanced RISC Machines, Ltd.

4.1 CPU Address Mapping

Table 4-1 describes the address map partitioning of the SA-110 microprocessor's 4GB address space. SDRAM and ROM regions are the only locations that the SA-110 microprocessor can mark as cacheable.

Table 4-1. SA-110 Microprocessor 4 GB Address Mapping

Function	Starting Address	Ending Address	Size
SDRAM	0000 0000h	0FFF FFFFh	256MB
Reserved	1000 0000h	3FFF FFFFh	—
SDRAM array 0 mode register	4000 0000h	4000 3FFFh	16KB
SDRAM array 1 mode register	4000 4000h	4000 7FFFh	16KB
SDRAM array 2 mode register	4000 8000h	4000 BFFFh	16KB
SDRAM array 3 mode register	4000 C000h	4000 FFFFh	16KB
X-Bus XCS0	4001 0000h	4001 0FFFh	4KB
X-Bus XCS1	4001 1000h	4001 1FFFh	4KB
X-Bus XCS2	4001 2000h	4001 2FFFh	4KB
X-Bus no CS	4001 3000h	4001 3FFFh	4KB
Reserved	4001 4000h	40FF FFFFh	—
ROM	4100 0000h	41FF FFFFh	16MB
CSR space	4200 0000h	420F FFFFh	1MB
Reserved	4210 0000h	4FFF FFFFh	—
SA-110 microprocessor cache flush	5000 0000h	50FF FFFFh	16MB
Reserved	5100 0000h	77FF FFFFh	—
Outbound write flush	7800 0000h	78FF FFFFh	16MB
PCI IACK/special trace	7900 0000h	79FF FFFFh	16MB
PCI type 1 configuration	7A00 0000h	7AFF FFFFh	16MB
PCI type 0 configuration	7B00 0000h	7BFF FFFFh	16MB
PCI I/O space	7C00 0000h	7C00 FFFFh	64KB
Reserved	7C01 0000h	7FFF FFFFh	—
PCI memory space	8000 0000h	FFFF FFFFh	2GB

4.2 Primary PCI Interface

The primary PCI interface of the EB554 is directly connected to the primary side of the 21554 PCI-PCI bridge. This bridge does not accept Type 1 configuration transactions. The host requires a special downstream configuration access to configure the 21285 or another device on the secondary side of the bridge.

4.3 Secondary PCI Interface

All devices on the secondary PCI interface are *hidden* from the host processor. The 21554 creates a new 4GB address space on the secondary side which is completely separate from the host address space. In most cases, the devices on the secondary PCI interface will be configured directly by the SA-110 microprocessor while the host system is being locked out from accessing devices downstream. Once the subsystem is configured, the SA-110 microprocessor can clear the 21554 lockout bit and the host will have access to devices on the secondary PCI interface. Address translation is used for both upstream and downstream transactions.

In addition, the secondary PCI interface can run either synchronously or asynchronously with the primary PCI clock. In synchronous mode, the latency through the bridge is reduced by one clock cycle and the secondary PCI clocks must be generated by the 21554. If using the Y3 external oscillator for asynchronous mode, resistor R234 must be removed and R241 installed. The 21554 supports asynchronous mode only if the ratio between the two PCI clocks is two or less.

4.4 DMA Channels

There are two DMA channels in the 21285. Either channel can move data from SDRAM-to-PCI or PCI-to-SDRAM. The DMA channels read parameters from a list of descriptors in memory, perform data movement, and stop when the list is exhausted. There is no restriction on byte alignment of either the source or destination address. The DMA reads always have all the byte enables asserted.

4.5 System Reset State

After system reset, the 21285 SDRAM controller and the X-Bus are disabled. The 21285 decodes the flash ROM at two locations; at its normal base address and also at an alias of 0. Since the 21285 SDRAM controller is disabled by reset, the SDRAM contents are UNDEFINED after reset.

Software Initialization and Configuration 5

This chapter describes the software initialization and configuration of the system hardware after a power-up or reset. The software can be run directly from ROM or copied to RAM and executed there. Typically this is an image loader that will find an image in ROM, copy it to RAM, and then branch to the image. If an image executes completely from ROM, the access timings can be modified in the 21285 to provide faster access to the ROM.

The correct sequence for the initialization steps required after reset is as follows:

1. Disable the flash ROM alias at address 0.
2. Determine the board configuration.
3. Initialize the X-Bus using the arbiter in the PCI-to-PCI bridge.
4. Size and initialize the SDRAM.
5. Configure the 21285 UART (optional).
6. Configure the PCI interface.
7. Configure the SA-110 microprocessor memory management unit (MMU) to turn on virtual memory, clock switching, and caches (optional).

5.1 Disable Flash ROM Alias

After reset, the 21285 decodes the flash ROM at two locations; at its normal base address of 4100 0000h and also at an alias of 0. Decoding this alias allows the SA-110 microprocessor to start executing from the normal reset vector address of 0. When the first write operation has been executed by the SA-110 microprocessor, the alias is disabled. Software should branch to the high-order alias of the flash ROM before executing the first write operation.

5.2 Access Flash ROM

In normal operation after reset, the flash ROM appears as a 32-bit read-only device. It is actually implemented using four, 8-bit devices that are accessed in parallel. Flash operations other than simple ROM operations are selected by writing specific commands into the command user interface. Refer to the *Intel 28F008SA Data Sheet* for details of the commands.

The four flash ROM devices can be accessed individually or simultaneously, depending on whether commands are sent to all parts simultaneously, or to a subset of the parts. For example, programming a flash ROM location is achieved by writing 40h to a byte location followed by writing a data byte to the same location. A write of 4040 4040h to any SA-IOP flash ROM location followed by a write of ABCD EF01h to the same location will result in ABCD EF01h being programmed into the location (one byte programmed into each of the four flash ROM devices). However, a write of 4000 0040 to an SA-IOP flash ROM location followed by a write of ABCD EF01h to the same location will result in the ABh and 01h bytes being programmed and the other two bytes of the Dword being unchanged.

5.2.1 Programming Flash ROM from SA-110 Microprocessor

The SA-110 microprocessor must not be executing from flash ROM when it executes a flash ROM programming algorithm because flash ROM programming requires a defined sequence of reads and writes to the flash ROM. Code fetches would disrupt this sequence.

5.2.2 Programming Flash ROM from PCI Interface

Before programming the flash ROM from the PCI interface, the 21285 ROM write byte address register must be set to 0. The 21285 allows the flash ROM to be reprogrammed from the PCI interface while the SA-110 microprocessor is running (the SA-110 microprocessor must not be accessing the flash ROM). However, the 21285 also provides a special mechanism for reprogramming the flash ROM called blank program mode. In this mode, the SA-110 microprocessor is held in reset by the 21285. On the SA-IOP, this mode is selected by moving a jumper on J9 or a switch on J11 (Section 2.2.2 or Section 2.2.3).

5.3 Initializing the X-Bus

The X-Bus should be configured by writing to the 21285 X-Bus cycle/arbiter register and X-Bus I/O strobe mask register. The X-Bus I/O strobe mask register is also used to set the polarity of some of the 21285 interrupt inputs.

A suggested initialization sequence for the SA-IOP is:

1. Write 1000 16DBh to the 21285 X-Bus cycle/arbiter register.
2. Write FCFC FCFCh to the 21285 X-Bus I/O strobe mask register.
3. Write 6000 0000h to the 21285 SA-110 microprocessor control register (to enable X-Bus chip selects 2 and 1).

5.4 Initializing SDRAM

Two sets of operations are required to allow access to the SDRAM:

- Configure the 21285 memory controller registers.
- Configure the mode registers in the SDRAM arrays.

This section provides a sample configuration, coded in the SA-110 microprocessor assembler. The code assumes that all four SDRAM arrays are populated with 2Mx8 parts that can run with a latency of 2. For this operation, use the following procedures.

1. Start with a set of equates for registers and register values.

Registers	Values
CSR_BASE	*EQU &42000000*
SDRAM_TIMING	*EQU &10C*
SDRAM_ADDR_SIZE_0	*EQU &110*
SDRAM_ADDR_SIZE_1	*EQU &114*
SDRAM_ADDR_SIZE_2	*EQU &118*
SDRAM_ADDR_SIZE_3	*EQU &11C*
CMD_DRIVE	*EQU &800*
PARITY_ENABLE	*EQU &1000*
T_{ref_min}	*EQU &010000*
T_{ref_norm}	*EQU &1A0000*
T_{rp}	*EQU &1*
T_{dal}	*EQU &4*
T_{rcd}	*EQU &20*
T_{cas}	*EQU &80*
T_{rc}	*EQU &300*

2. After reset, the SDRAM arrays are in an unknown state. To put them into a known state, force an all-banks precharge to each of the four possible arrays. All four arrays must be accessed, even if there are less than four arrays installed. This is necessary because the 21285 counts these precharge accesses, and inhibits access to the SDRAM until all four have been completed. Failure to perform four precharge accesses will result in unpredictable operation. An all-banks precharge is initiated by a read from any address in the mode register address space.

```
ldr   r0,=&40000008    ;   SDRAM array 0
ldr    r0,   [r0]
ldr    r0,=&40004008    ;   SDRAM array 1
ldr    r0,   [r0]
ldr    r0,=&40008008    ;   SDRAM array 2
ldr    r0,   [r0]
ldr    r0,=&4000C008    ;   SDRAM array 3
ldr    r0,   [r0]
```

3. Write to the SDRAM Mode Register in the SDRAMs. This requires one write operation for each SDRAM array. The address is important, not the data. The offset from the start of the mode space for each SDRAM array controls what data is written to the SDRAM mode register. The mode register should be configured for a burst size of four and for linear addressing.

```
ldr    r0,=&40000008:OR:Tcas
str    r0,   [r0]
ldr    r0,=&40004008:OR:Tcas
str    r0,   [r0]
ldr    r0,=&40008008:OR:Tcas
str    r0,   [r0]
ldr    r0,=&4000C008:OR:Tcas
str    r0,   [r0]
```

4. Write to the SDRAM Timing Register in the 21285. Set the refresh interval to the minimum because we have to wait for eight refresh cycles to complete before we can rely on the SDRAMs operating normally.

```
ldr    r1,=CSR_BASE

ldr    r0,=Trp:OR:Tdal:OR:Trcd:OR:Tcas:OR:Trc:OR:
CMD_DRIVE:OR:Tref_min

str    r0,  [r1,#SDRAM_TIMING]
```

5. Wait for eight refresh cycles to complete. The minimum refresh interval is 32 cycles, and with the Icache off, the complete process takes 256 cycles.

```
        ldr    r0,=&100
wait     subs   r0,ro,#1
        bgt    wait
```

6. Write to the four 21285 SDRAM Address and Size Registers. This simple code assumes four arrays are installed and that are all the same size and type. A more sophisticated code would automatically detect and size each array.

```
ldr    r0,=&14
str    r0,  [r1,#SDRAM_ADDR_SIZE_0]
ldr    r0,=&800014
str    r0, [r1,#SDRAM_ADDR_SIZE_1]
ldr    r0,=&1000014
str    r0,[  r1,#SDRAM_ADDR_SIZE_2]
ldr    r0,=&1c00014
str    r0,  [r1,#SDRAM_ADDR_SIZE_3]
```

7. Finally, reset the refresh interval to a sensible value. Continuing to run with a very short interval would waste memory bandwidth. The refresh interval is calculated to refresh 4096 rows in 64 ms.

```
ldr     r0,=Trp:OR:Tdal:OR:Trcd:OR:Tcas:OR:Trc:OR:CMD_DRIVE:OR:
Tref_norm

str    r0,  [r1,#SDRAM_TIMING]
```

5.5 Reinitialize SDRAM

The 21285 only allows SDRAM mode writes to be performed when refresh is disabled. Therefore, the initialization sequence described in Section 5.4 cannot be executed again. Either of the following two modifications will allow the code to be executed again:

- At the start of the sequence, read the 21285 SDRAM timing register and check whether refresh is enabled (that is, the refresh interval is set to a non-zero value). If it is enabled, the SDRAM initialization has already been performed and no further action is needed. If refresh is not enabled, execute the initialization sequence described in Section 5.4.
- At the start of the sequence, disable refresh by setting the refresh interval to 0. Wait for 15 bus clock cycles to ensure that any pending or in-progress refresh complete successfully. Then, execute the initialization sequence described in Section 5.4.

5.6 Initialize Secondary PCI Interface

This section describes the minimum set of PCI registers that must be configured to allow the PCI interface to be used.

1. Set the following registers to a known state.
   a. Disable outbound interrupts in the 21285 outbound interrupt mask register (30h).
   b. Clear doorbell interrupts to the PCI in the 21285 doorbell PCI mask register (150h).
   c. Clear doorbell interrupts to the SA-110 microprocessor in the 21285 doorbell SA-110 microprocessor mask register (154h).
   d. Set the PCI not reset bit in the SA-110 microprocessor control register (13Ch).
   e. Request an interrupt by writing a 1 to interrupt pin register (3Dh).

Note: If the SA-110 microprocessor is required to directly address host memory, then the PCI extension register must be set accordingly.

   f. For now, set the PCI extension register to 0. After a successful boot, read bit 31 of the assigned address in BAR0 (10h) of the 21285. This will have the correct PCI memory space bank (upper or lower) address. Write the state of this bit into bit 15 of the PCI extension register (140h).

2. Open up a window from PCI memory space into the SA-IOP SDRAM address space. The PCI memory space should be aligned with a downstream translated address in the 21554 so that the host PC will be able to access the SDRAM local memory. The following example creates an 8MB window:
   a. Write 007C 0000h to the 21285 CSR base address mask register (F8h).
   b. Write the offset from BAR0 requested in the CSR base address offset register (FCh).
3. Finally, set bit 0, INITIALIZE_COMPLETE, in the 21285 SA-110 microprocessor control register. This allows the 21285 to respond to PCI configuration cycles, as described in Section 5.5.

5.7 Initialize 21285 UART

The SA-IOP runs with an FCLK_IN frequency of 60 MHz. The frequency of FCLK_IN determines the appropriate divisors when configuring the 21285 internal UART. Table 5-1 lists the baud rate divisors for the 60 MHz bus frequency FCLK_IN signal.

Table 5-1. 21285 Baud Rate Divisors for 60 MHz FCLK_IN Signal

Baud Rate	Divisor	Error (%)
300	3124	0.00
600	1561	–0.03
1200	780	–0.03
1800	520	0.03
2000	468	0.05
2400	390	0.10
3600	259	– 0.16
4800	194	– 0.16
7200	129	– 0.16
9600	97	0.35
19200	48	0.36
38400	23	–1.80
56000	16	1.62
128000	6	–5.40

The onchip UART initialization sequence is as follows:

1. Initialize the UARTCON (174h) register to 0.
2. Read the RXSTAT (164h) to drain the FIFO contents.
3. Set the baud rate, stop bits, parity, and so on, in the H_UBRLCR, M_UBRLCR, L_UBRLCR (164h to 16Ch). There is a specific sequence required to program these registers.

For detailed information on the 21285 UART functionality, refer to the *21285 Core Logic for SA-110 Microprocessor Data Sheet*.

5.8 Configuring Cacheable/Non-Cacheable Space

To get maximum performance from the SA-110 microprocessor, enable clock switching and turn on the internal Icache and Dcache. The Dcache can only be enabled when the MMU is enabled. The page-tables used by the MMU determine, on a page-by-page basis, if the contents of the memory page is Dcacheable. These page-tables also determine if writes to the page can use the SA-110 microprocessor write buffer. Refer to the *StrongARM*™** *SA-110 Microprocessor Technical Reference Manual* for more details.

For correct operation, the following rules must be followed:

- The SDRAM address region may be marked Icacheable, Dcacheable, and/or buffered.

- The flash ROM address region may be marked Icacheable and Dcacheable for reads. During writes to flash ROM (reprogramming) the Dcache and write buffer should be disabled.
- Some regions of I/O space can be marked as buffered, non-cacheable to improve the performance of write operations. The PCI memory space, in particular 8000 0000h to FFFF FFFFh should be marked as buffered, non-cacheable. This allows SA-110 microprocessor writes to this region to be merged within the 21285, resulting in write bursts on the PCI whenever possible.
- The 21285 CSR address region should be marked non-cacheable, non-buffered.

If the software Dcache flush algorithm described in the *StrongARM*™** *SA-110 Microprocessor Technical Reference Manual* is implemented, the 21285's SA-110 microprocessor cache flush region (5000 0000h to 50FF FFFFh) can be accessed. Read accesses to this region complete in the minimum amount of time and return dummy data. This minimizes the time it takes to flush the caches.

Schematics A

Table A-1 lists the SA-IOP schematics included in this appendix.

Table A-1. SA-IOP Schematics

Sheet Number	Description
A-2	Block diagram
A-3	SA-110 microprocessor
A-4	X-Bus expansion headers
A-5	21285
A-6	Address and data buffers
A-7	X-Bus/SDRAM buffers and soft I/O
A-8	Flash and emulator socket
A-9	SDRAM DIMM socket
A-10	Power regulation
A-11	Bulk decoupling
A-12	Configuration jumpers
A-13	JTAG port
A-14	CPU and 21285 decoupling
A-15	Series termination resistors
A-16	21285 debug serial port and spare gates
A-17	PCI connector
A-18	21554 decoupling and pullups
A-19	Champ and edge connectors
A-20	LED schematic
A-21	21143
A-22	100BaseT PHY and magnetics
A-23	RJ45 jack and chokes
A-24	Link and Activity LEDs
A-25	21143 boot and serial ROM
A-26	21554
A-27	Secondary clock generation and hot swap test
A-28	21554 parallel and serial ROM
A-29	21554 pullup resistors

For a “B” size PDF of the schematics, go to this site: (http://developer.intel.com/design/strong/schems/sa-iop.htm)

Support, Products, and Documentation

If you need technical support, a *Product Catalog*, or help deciding which documentation best meets your needs, visit the Intel World Wide Web Internet site:

http://www.intel.com

Copies of documents that have an ordering number and are referenced in this document, or other Intel literature may be obtained by calling **1-800-332-2717** or by visiting Intel's website for developers at:

http://developer.intel.com

You can also contact the Intel Massachusetts Information Line or the Intel Massachusetts Customer Technology Center. Please use the following information lines for support:

For documentation and general information:	
Intel Massachusetts Information Line	
United States:	1–800–332–2717
Outside United States:	1–303-675-2148
Electronic mail address:	techdoc@intel.com

For technical support:	
Intel Massachusetts Customer Technology Center	
Phone (U.S. and international):	1–978–568–7474
Fax:	1–978–568–6698
Electronic mail address:	techsup@intel.com