Intel StrataFlash® Embedded Memory (P30) to Intel® IXP42X Network Processors and Intel® IXC1100 Control Plane Processor Design Guide

Application Note 832

October 2005

Order Number: 308294, Revision: 002
13 Oct 2005

Contents

1.0 Introduction 5

2.0 P30 Flash Memory Device 5

3.0 IXP42X Processor and IXC1100 Processor 6

4.0 Hardware Interface 7

4.1 Flash Memory Interface 7
4.2 Processor Memory Interfaces 8
4.3 Interface Considerations 10
4.4 Hardware Considerations 10
4.5 Reset Consideration 10

5.0 Register Settings 13

5.1 Flash Memory Register Settings 13
5.2 Processor Register Settings 13
5.3 Timing and Control Register 13
5.4 Configuration Register 0 15

6.0 Bus Operation and Timings 17

6.1 Asynchronous Single-Word Read 18
6.2 Asynchronous Write 19

7.0 Summary 19

Appendix A Additional Information 20

Appendix B Enabling P30 Flash on IXDPG425 21

B.1 RedBoot porting 21
B.1.1 Using Redboot to update RedBoot 25
B.1.2 Linux MTD 25
B.2 Using EPI ICE to Program redboot binary code to P30 26

Revision History

Date of Revision	Version	Description
29 Jun 2005	-001	Initial Release
13 Oct 2005	-002	Updated Processor Register Settings, added Redboot release note

1.0 Introduction

This application note describes the interface between the following:

- Intel StrataFlash® Embedded Memory (P30) device
- Intel® IXP42X Network Processor (IXP42X processor) and Intel® IXC1100 Control Plane Processor (IXC1100 processor)

This application note also discusses general concepts involved when interfacing the features and control signals of the P30 flash memory device.

This document was written based on information available at the time for the P30 flash memory device and the IXP42X Processor and IXC1100 Processor. Any changes in specifications to any of these devices might not be reflected in this document. Refer to the appropriate documents or sales personnel for the most current information before finalizing any design.

2.0 P30 Flash Memory Device

The P30 flash memory device brings two-bit-per-cell storage technology to the embedded flash market segment. It is available in 64-Mbit, 128-Mbit, 256-Mbit, and 512-Mbit densities in TSOP and EzBGA packages, and up to 1-Gbit in QUAD+ SCSP package.

Benefits of the two-bit-per-cell technology include:

- More density in less space
- High-speed interface
- Low cost-per-bit NOR device
- Support for code and data storage

Features include:

- Fast initial read access
- Synchronous burst reads
- Asynchronous page-mode reads
- Buffered writes
- Flexible security options

The P30 flash memory device is designed for applications that require high density and low cost, such as:

- networking
- wireless communication devices
- storage applications
- telecommunications
- digital set-top boxes
- audio recording
- digital imaging

The P30 flash memory product family is manufactured using Intel® 130nm ETOX™ VIII process technology.

3.0 IXP42X Processor and IXC1100 Processor

The IXP42X processor and IXC1100 processor contain an Intel® StrongARM* V5TE architecture compliant microprocessor referred to as Intel XScale® technology. These processors are designed for high performance (measured in W/MIPs) and low power.

The IXP42X processor and IXC1100 processor are designed with Intel® 0.18-µ production semiconductor process technology. This process technology helps the IXP42X processor and the IXC1100 processor to operate over a wide range of low-cost networking applications. Other factors that help these processors to operate with diverse network applications include the compactness of the Intel® StrongARM* RISC ISA, simultaneous processing using three integrated network processing engines, and numerous dedicated function peripheral interfaces.

The IXP42X processor and the IXC1100 processor provide the following:

- Two MII interfaces
- Intel® IXB8055 UTOPIA/POS Reference Design level 2 interface
- A USB v1.1 device controller with embedded transceiver
- A 32-bit, 33-MHz/66-MHz PCI bus
- A 32-bit, 133-MHz SDRAM interface
- Two Universal Asynchronous Receiver/Transmitters (UARTs)
- Two high-speed serial interfaces
- 16 General Purpose Input/Output interfaces (GPIOs) and a 16-bit expansion bus

The Expansion Bus Controller provides an interface from the internal South Advanced High Performance Bus (AHB) to the external flash, Host-Port Interfaces (HPI), and other devices that operate at slower speeds than the processor.

4.0 Hardware Interface

This section describes the hardware interface signals between the P30 flash memory device and the IXP42X processor or the IXC1100 processor. This document assumes that all other device signals and power supplies are properly connected for optimal device operation.

4.1 Flash Memory Interface

Table 1 lists the P30 flash memory device hardware Interface signals.

Table 1. Embedded Flash Memory (P30) Device Hardware Interface Signals (Sheet 1 of 2)

P30 Flash Memory Device Signal	Type	Description
A[MAX:1] or A[MAX:0]	Input	**ADDRESS INPUTS:** Device address inputs. • Easy BGA and TSOP package: – 64-Mbit: A[22:1] – 128-Mbit: A[23:1] – 256-Mbit: A[24:1] – 512-Mbit: A[25:1]. A1 is the least significant word (x16) address bit. • QUAD+ SCSP: – 64-Mbit: A[21:0] – 128-Mbit: A[22:0] – 256-Mbit: A[23:0] – 512-Mbit: A[24:0]. A0 is the least significant word (x16) address bit.
D[15:0]	Input/Output	**DATA INPUT/OUTPUTS:** • Inputs during write operations. • Outputs during read operations • High-Z (float) when CE# or OE# are deasserted.
CE#	Input	FLASH CHIP ENABLE: Active low input. • CE#-low enables the device. • CE#-high disables the device, placing it in standby mode. CE#-high also places the data and WAIT signals in High-Z.
OE#	Input	**OUTPUT ENABLE:** Active low input. • OE# low enables the device output data buffers during read cycles. • OE# high places the data outputs and WAIT in High-Z.
WE#	Input	**WRITE ENABLE.** Active low input. WE#-low enables the write buffers. Address and data are latched on the rising edge of WE#.
WP#	Input	**WRITE PROTECT.** Active low input. WP# enables and disables the lockdown mechanism. • WP#-low enables the lock-down mechanism. Blocks configured for lock-down cannot be unlocked using software commands. • WP#-high overrides the lock-down mechanism, enabling software to unlock those blocks.
CLK	Input	**CLOCK**. Configurable as valid rising edge or valid falling edge. CLK synchronizes the device to the system bus clock and increments the internal address generator in synchronous-burst read mode. • In synchronous-burst read mode, the initial address is latched on the rising edge of ADV# or the next valid clock edge when ADV# is low, whichever occurs first. • For the P30 flash device to be used in asynchronous mode, CLK must be tied to VCCQ or VSS (CLK is tied to the I/O supply in this example design).

Table 1. Embedded Flash Memory (P30) Device Hardware Interface Signals (Sheet 2 of 2)

P30 Flash Memory Device Signal	Type	Description
ADV#	Input	**ADDRESS VALID.** Active low input. • In synchronous-burst read mode, the initial address is latched on the rising edge of ADV# or the next valid clock edge when ADV# is low, whichever occurs first. • For the P30 flash device to be used in asynchronous mode, ADV# must be tied to VSS to allow addresses to flow through (ADV# is tied to VSS in this example design).
WAIT	Output	**WAIT**. Configurable for high-true or low-true. • In synchronous-burst read mode: – WAIT indicates invalid data when asserted. – WAIT indicates valid data when deasserted. – WAIT is High-Z whenever CE# or OE# is deasserted. • For the P30 flash device to be used in asynchronous mode, WAITmust be floated (WAIT is floated in this example design).
RST#	Input	**RESET**. Active low input. • RST#-low resets internal automation and inhibits write operations. • RST#-high enables normal operation.

4.2 Processor Memory Interfaces

The IXP42X processor and IXC1100 processor each contains an Expansion Bus Controller to interface to a variety of components that do not have PCI bus interfaces or do not optimally reside on the PCI Bus. Examples of such components include:

- Flash memory
- SRAM memory
- Simple displays
- Digital Signal Processors (DSPs) with host interface ports
- Digital/Analog converters
- Programmable logic
- Control interfaces to devices such as UTOPIA or other PHY components

See Figure 1, Processor System Block Diagram.

Figure 1. Processor System Block Diagram

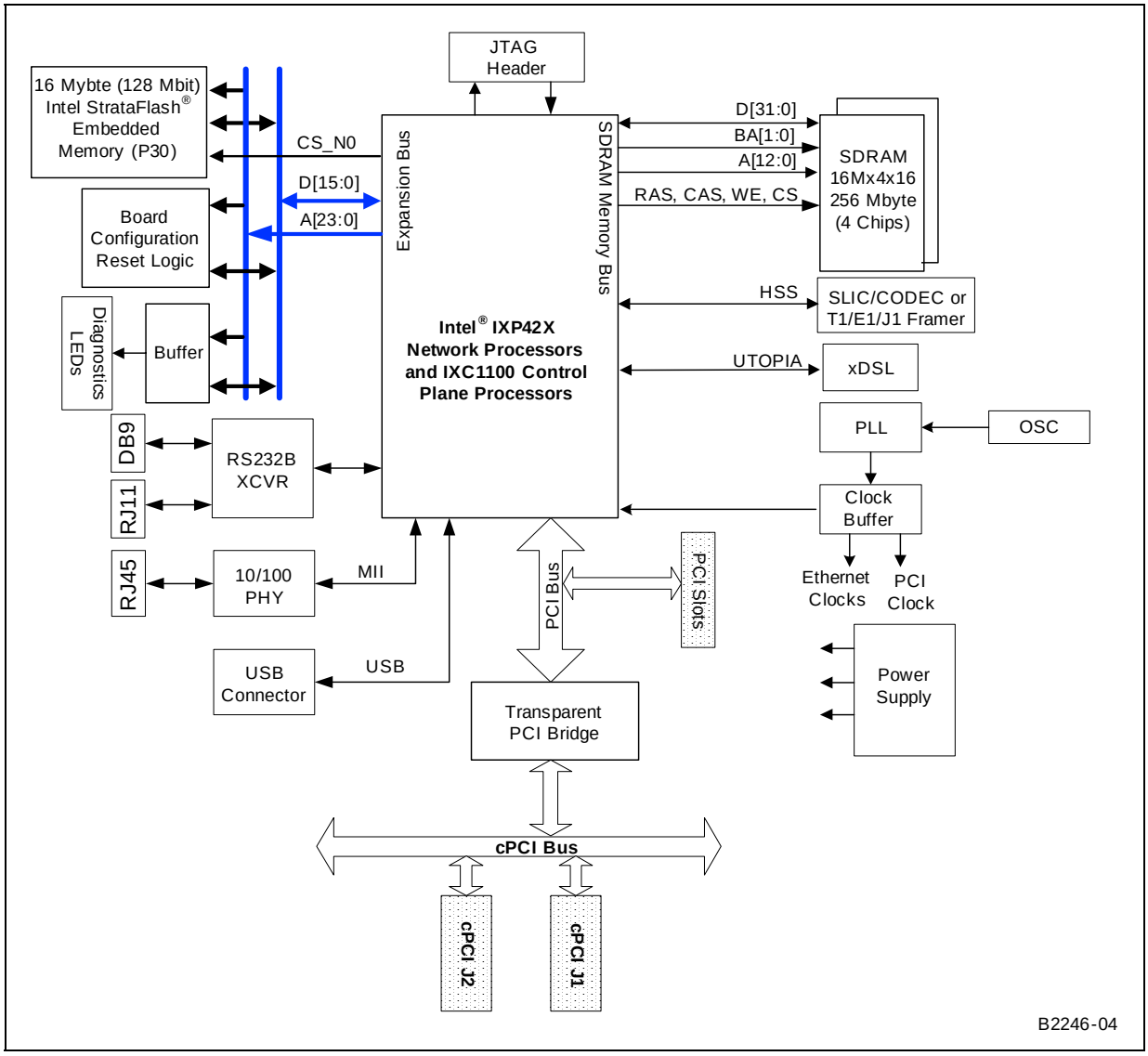


Table 2 describes the P30 flash memory device expansion bus interface signals.

Table 2. Embedded Flash Memory (P30) Interface Signals for the Expansion Bus

P30 Flash Memory Interface Signal	Description
EXP_ADDR[23:0]	Expansion Memory Address Bus (output): Provides the address used for memory accesses.
EXP_D[15:0] Expansion Data Bus (I/O)	• Inputs during read operations. • Outputs during write operations.
EXP_WR_N Write Strobe (output)	Intel mode write strobe / Motorola mode data strobe / TI mode data strobe.
EXP_RD_N Read Strobe (output)	Intel mode read strobe / Motorola mode read-not-write strobe / TI mode data strobe.
EXP_CS_N[0] Static Memory Chip Select (output)	Flash memory chip select for expansion bus.
RESET_N Reset (input)	Used as a reset input to the device after power up conditions are met.

4.3 Interface Considerations

The first design example shown in Figure 2 on page 11 uses one 128-Mbit P30 flash memory device interfaced with the IXP42X processor or the IXC1100 processor in a x16 data bus configuration. The expansion bus timing shown in this application note is running at 33 MHz.

A second design example shown in Figure 3 on page 12 uses one 256-Mbit P30 flash memory device interfaced with the IXP42X processor or the IXC1100 processor in a x16 data bus configuration, using a glue logic AND gate. Other bus speeds and memory sizes can be implemented by modifying the design.

These sample interfaces do not include information regarding system initialization, interrupt control, exception handling, or other peripheral device operations. Assert or negate the external device signals/pins as necessary for the desired device operation. Read all applicable documentation, such as datasheets, user's manuals, and errata before attempting to design using this interface. For information about additional references, see Appendix A, “Additional Information” on page 20.

4.4 Hardware Considerations

Figure 2 on page 11 shows the 128-Mbit P30 flash memory device connected to the IXP42X processor and IXC1100 processor. The BootROM address space supports up to 16-Mbyte (128-Mbit) of flash. The BootROM maps to the Intel XScale® core physical address 0x00. At startup (reset), the IXP42X processor and IXC1100 processor begin fetching and executing instructions from address 0x00.

4.5 Reset Consideration

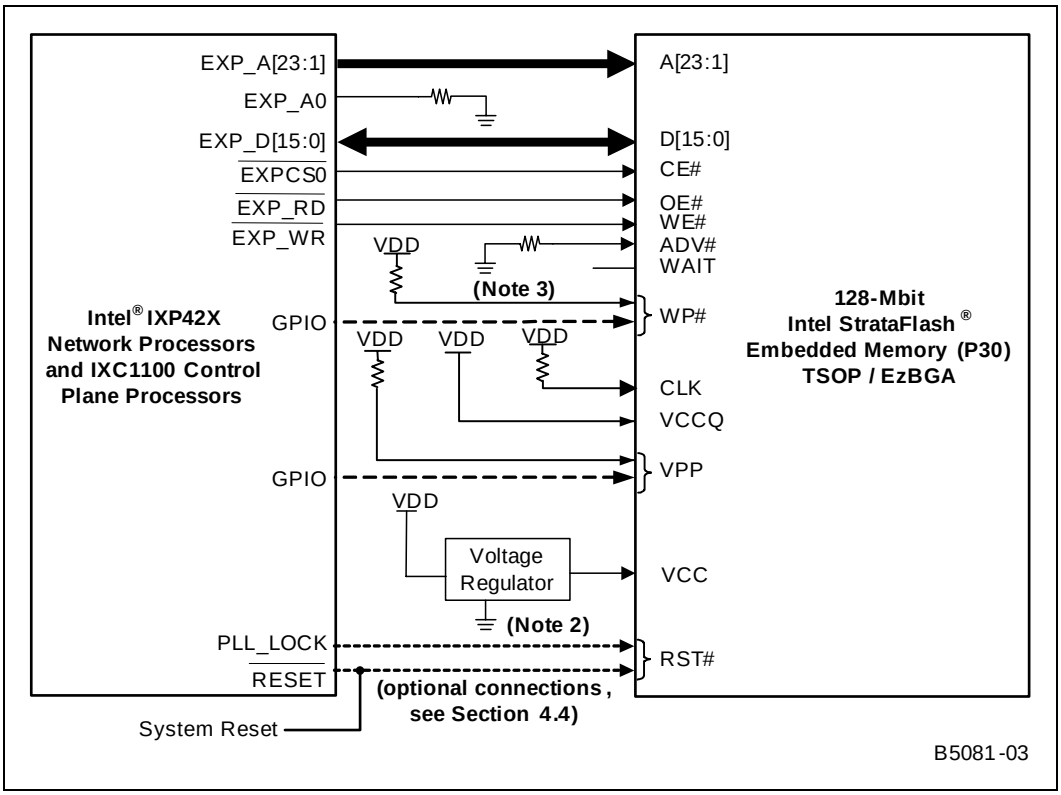
Consider how the flash reset input, RST#, is connected in the system. If the internal watch-dog timer (WDT) on the IXP42x is used, then use an external signal to reset the flash when a WDT time-out event occurs. The flash device enters Read Array mode when the processor exits from reset. Without this reset, the processor cannot read the boot code from the flash memory device, causing the system to hang.

Connect PLL_LOCK from the processor to RST# on the flash device. PLL_LOCK goes low whenever a WDT event occurs, or whenever RESET_N or PWRON_RST_N is asserted. PLL_LOCK deasserts for 24 * 1/33.33MHz = 720 ns, which is sufficient only if the P30 flash device is *not* performing a program or erase operation.

If the P30 flash device *is* performing a program or erase operation, 25 µs is the minimum required time for t_{PLPH}. In this case, the PLL_LOCK low pulse-width is not sufficient to meet t_{PLPH}. To meet this worst-case timing requirement, use external glue logic to stretch the PLL_LOCK low-time from 720 ns to 25 µsec or greater.

For applications that do not use the processor watch-dog timer, RST# can be driven directly by the system reset signal (see Figure 2).

Figure 2. Block Diagram Using a 128-Mbit Intel StrataFlash® Embedded Memory (P30) Device

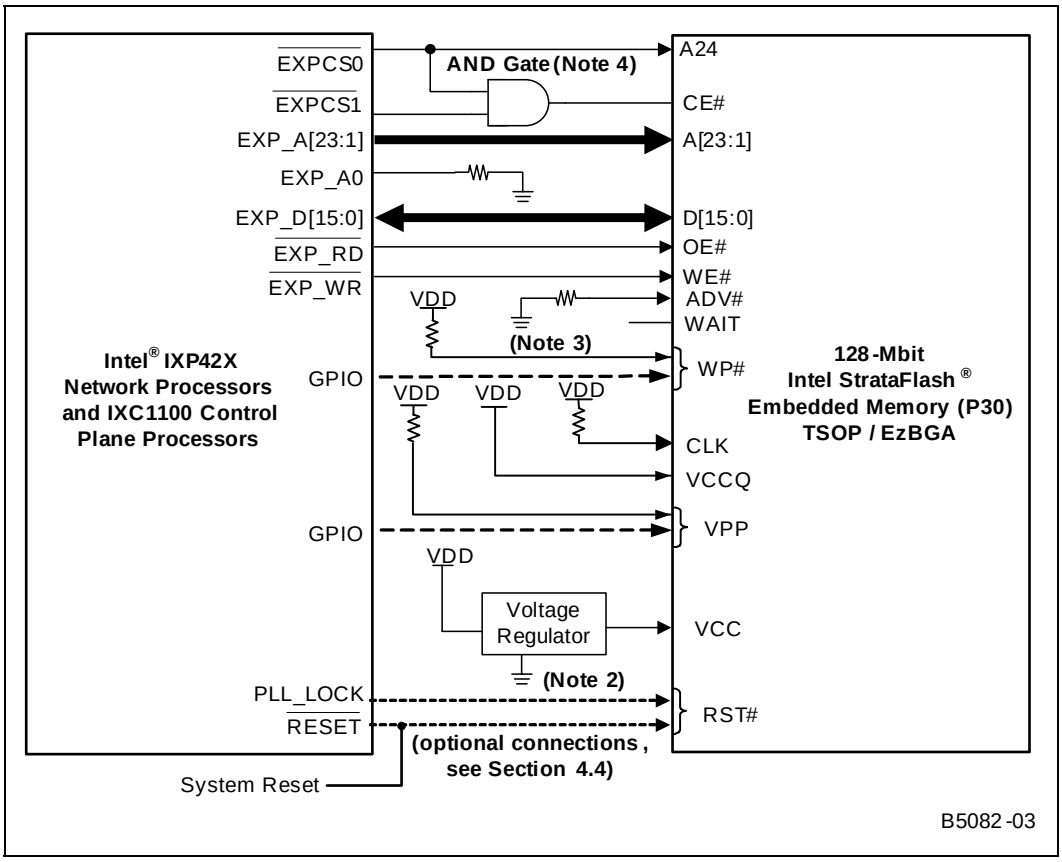


Notes:

1. Dotted lines are optional connections. See the *Intel StrataFlash® Embedded Memory (P30) 1-Gbit P30 Family Datasheet* for more details.
2. Consider the P30 flash memory device maximum current rating when choosing a voltage regulator.
3. Pull up WP# if not used.
4. EXP_A0 is a byte address of the processor. Because the P30 flash memory device already has a 16-bit bus width, ground only EXP_A0.
5. Use resistors to pull all control signals and unused inputs to an inactive state if built-in pull-up/ pull-downs are not available.

Figure 2 also shows a Low-Dropout (LDO) voltage regulator component used to convert the 3-volt range power supply of the IXP42X processor and the IXC1100 processor to 1.8-volt range.

Figure 3. Block Diagram Using a 256-Mbit Intel StrataFlash® Embedded Memory (P30) Device



Notes:

1. Dotted lines are optional connections. See the *Intel StrataFlash® Embedded Memory (P30) 1-Gbit P30 Family Datasheet* for more details.
2. Consider the P30 flash memory device maximum current rating when choosing a voltage regulator.
3. Pull up WP# if not used.
4. Consider the propagation delay of the AND gate affecting the A24 and CE# signals, for bus operation and timings and when setting the processor registers.
5. EXP_A0 is a byte address of the processor . Because the P30 flash memory device already has a 16-bit bus width, ground only EXP_A0.
6. Use resistors to pull all control signals and unused inputs to an inactive state if built-in pull-up/ pull-downs are not available.

Figure 3 shows the block diagram using a monolithic 256-Mbit P30 flash device and a glue logic AND gate. A monolithic 256-Mbit P30 flash device is typically desired, rather than using two 128-Mbit P30 flash devices, for a more cost-effective and optimal design. In Figure 3, one processor chip select connects to the 256Mbit P30 flash device A24 pin acting as a 128-Mbit bank select, in conjunction with the AND gate. This connection enables using a monolithic 256-Mbit P30 flash device, despite the 128-Mbit (16-MByte) maximum addressability of the IXP42X processor and the IXC1100 processor chip selects.

For this design, use a high-speed AND gate with the least propagation delay, to minimize impact to bus operations and timings.

5.0 Register Settings

After a system reset, the processor registers assume default values. The following sections describe the affected registers and their settings.

5.1 Flash Memory Register Settings

If you use the P30 flash memory device with the IXP42X processor and the IXC1100 processor, you do not need to configure any flash register settings. Following a device power-up or reset, asynchronous read mode is the default read mode for the P30 flash memory device, and the flash device is set to read array mode.

5.2 Processor Register Settings

The expansion bus controller registers configure the operation of the external memory interface. The settings for the Timing and Control Register (EXP_TIMING_CS0) and the Configuration Register 0 (EXP_CNFG0) affect the interface timings for external memory accesses. These registers are described in the following sections.

Note: For current information, see the *Intel® IXP42X Product Line of Network Processors and IXC1100 Control Plane Processors Specification Update* and the *Intel® IXP42X Product Line and IXC1100 Control Plane Processors Developer's Manual.*

Only the register bit fields that affect memory interface operations with the P30 flash memory device are discussed. Program all other bit fields as necessary for proper operation.

5.3 Timing and Control Register

Each chip select on the expansion bus has a Timing and Control Register. In this design example, the boot device, a P30 flash memory device, connects to Chip Select 0 (CS0):

- Bit 31 of the Timing and Control Register (EXP_TIMING_CS0) enables or disables the flash memory device.
- Bits [29:16] of the EXP_TIMING_CS0 configure read/write flash memory accesses.

For this design example, configure EXP_TIMING_CS0 [31;29:16;6;4:3;1:0] with the following values, assuming that the expansion bus is running at 33MHz in this example (and not 66MHz):

- EXP_TIMING_CS0 (CSx_EN [31]) = 1 (Chip Select x enabled)
- EXP_TIMING_CS0 (T1 - Addressing Timing [29:28]) = 01 (Extend address phase: 30ns)
- EXP_TIMING_CS0 (T2 - Setup / Chip Select Timing [27:26])= 01 (Extend setup phase: 30ns)
- EXP_TIMING_CS0 (T3 - Strobe Timing [25:22]) = 0011 (Extend strobe phase: 90ns). For 256P30 TSOP only, use 0100 (Extend strobe phase: 120ns).
- EXP_TIMING_CS0 (T4 - Hold Timing [21:20]) = 01 (Extend hold phase: 30ns)
- EXP_TIMING_CS0 (T5 - Recovery Timing [19:16]) = 01 (Extend recovery phase: 30ns)
- EXP_TIMING_CS0 (BYTE_RD16 [6]) = 1 (Byte access enabled, recommended setting)
- EXP_TIMING_CS0 (MUX_EN [4]) = 0 (Separate address and data buses)

- EXP_TIMING_CS0 (SPLT_EN [3]) = 1 (AHB split transfers enabled)
- EXP_TIMING_CS0 (WR_EN [1]) = 1 (Writes to CS region are enabled)
- EXP_TIMING_CS0 (BYTE_EN [0]) = 0 (= Expansion bus uses 16-bit-wide data bus)

Table 3 describes the Timing and Control register of CS0.

Table 4 describes the bit level description of the Timing and Control register.

Table 3. Timing and Control Register for Chip Select 0

Register Name:	EXP_TIMING_CS0																		
Hex Offset Address:	0XC4000000						Reset Hex Value:	0xBFFF3C4x											
Register Description:	Timing and Control Registers																		
Access: Read/Write																			
31	30	29	28	27	26	25... 22		21	20	19... 16		15	14	13... 10	9. 7	6	5	4	3	2	1	0
CSx_EN	(Rsvd)	T1		T2		T3		T4		T5		CYCLE_ TYPE		CNFG[3:0]	(Rsvd)	BYTE_RD16	HRDY_POL	MUX_EN	SPLT_EN	(Rsvd)	WR_EN	BYTE_EN

Table 4. Bit Level Definition for each of the Timing and Control Registers (Sheet 1 of 2)

Bits	Name	Description
31	CSx_EN	0 = Chip Select x disabled 1 = Chip Select x enabled
30		(Reserved)
29:28	T1 – Address timing	00 = Generate normal address phase timing 01 - 11 = Extend address phase by 1 - 3 clocks
27:26	T2 – Setup / Chip Select Timing	00 = Generate normal setup phase timing 01 - 11 = Extend setup phase by 1 - 3 clocks
25:22	T3 – Strobe Timing	0000 = Generate normal strobe phase timing 0001-1111 = Extend strobe phase by 1 - 15 clocks
21:20	T4 – Hold Timing	00 = Generate normal hold phase timing 01 - 11 = Extend hold phase by 1 - 3 clocks
19:16	T5 – Recovery Timing	0000 = Generate normal recovery phase timing 0001-1111 = Extend recovery phase by 1 - 15 clocks
15:14	CYC_TYPE	00 = Configures the expansion bus for Intel cycles. 01 = Configures the expansion bus for Motorola* cycles. 10 = Configures the expansion bus for HPI cycles. (HPI is reserved for chip selects [7:4] only) 11 = Reserved

Table 4. Bit Level Definition for each of the Timing and Control Registers (Sheet 2 of 2)

Bits	Name	Description
13:10	CNFG[3:0]	Device Configuration Size. Calculated using the formula: SIZE OF ADDR SPACE = $2^{(9+CNFG[3:0])}$ For Example: 0000 = Address space of 2^9 = 512 Bytes … 1000 = Address space of 2^{17} = 128 Kbytes … 1111 = Address space of 2^{24} = 16 Mbytes
9:7		(Reserved)
6	BYTE_RD16	Byte read access to Half Word device 0 = Byte access disabled. 1 = Byte access enabled.
5	HRDY_POL	HPI HRDY polarity (reserved for exp_cs_n[7:4] only) 0 = Polarity low true. 1 = Polarity high true.
4	MUX_EN	0 = Separate address and data buses. 1 = Multiplexed address / data on data bus.
3	SPLT_EN	0 = AHB split transfers disabled. 1 = AHB split transfers enabled.
2		(Reserved)
1	WR_EN	0 = Writes to CS region are disabled. 1 = Writes to CS region are enabled.
0	BYTE_EN	0 = Expansion bus uses 16-bit-wide data bus. 1 = Expansion bus uses only 8-bit data bus.

5.4 Configuration Register 0

The general-purpose Configuration Register 0 (EX_CNFG0) of the expansion bus is loaded at reset from the pull-ups or pull-downs on the expansion bus corresponding address lines. See the *Intel® IXP42X Product Line of Network Processors and IXC1100 Control Plane Processors Developer's Manual*. Bit 0 of the EX_CNFG0 configures the data bus width of the flash memory device.

For this design example, configure EX_CNFG0 [0] with EX_CNFGO (0) = 0. This setting configures the flash data bus width to 16-bits.

- Table 5, "Expansion Bus Configuration Register 0" on page 16 defines the Configuration Register 0.
- Table 6, "Expansion Bus Configuration Register 0 Description" on page 16 lists the pin-level description for Configuration Register 0.

Note: At system reset, after the boot sequence completes, bit 31 is written to a 0. This bit setting switches the default system memory map to place the SDRAM controller at address 0x00000000 to 0x0FFFFFFF.

Table 5. Expansion Bus Configuration Register 0

Register Name:	EXP_CNFG0																															
Hex Offset Address:	0XC4000020								Reset Hex Value:	0x8XXXXXXX																						
Register Description:	Configuration Register #0																															
Access: Read/Write.																																
31	30						24	23	22	21	20			17	16											5	4	3	2	1	0		
MEM_MAP	(Reserved)							CLK bit 2	CLK Bit 1	CLK Bit 0	User-configurable				(Reserved)												PCI_CLK	RES	PCI_ARB	PCI_HOST	8/16		

Table 6. Expansion Bus Configuration Register 0 Description

Bit	Name	Description
31	MEM_MAP	Location of EXPBus in memory map space: 0 = Located at 50000000 (normal mode) 1 = Located at 00000000 (boot mode)
30:24		(Reserved)
23:21	Intel XScale core Clock Set[2:0]	Allow a slower Intel XScale core clock speed to override the device fuse settings. However, cannot be used to over clock core speed. Refer to the “Setting The Intel XScale® Core Operation Speed” section in the *Intel® IXP42X Product Line of Network Processors and IXC1100 Control Plane Processor Developer’s Manual* for additional details.
20:17		User-configurable. See the *Intel® IXP42X Product Line of Network Processors and IXC1100 Control Plane Processor Developer’s Manual* for additional comments.
16:6		(Reserved)
5		(Reserved)
4	PCI_CLK	Sets the clock speed of the PCI Interface: 0 = 33 MHz 1 = 66 MHz
3	—	(Reserved)
2	PCI_ARB	Enables the PCI Controller Arbiter: 0 = PCI arbiter disabled 1 = PCI arbiter enabled
1	PCI_HOST	Configures the PCI Controller as PCI Bus Host: 0 = PCI as non-host 1 = PCI as host
0	8/16 FLASH	Specifies the data bus width of the flash memory device: 0 = 16-bit data bus 1 = 8-bit data bus

6.0 Bus Operation and Timings

For the following discussions, consult the appropriate IXP42X processor, IXC1100 processor, and P30 embedded flash memory documents for specific timing information about the individual components presented in this design example. In the following sections, flash memory timings are prefixed with either R (read) or W (write).

Because the IXP42X processor and the IXC1100 processor timings are register dependent, the associated register field names are shown where applicable.

For the following discussion, refer to:

- Table 7, "Read Timing Definitions" for bus operations and timings.
- Table 8, "Write Timing Definitions" for a description of the flash memory timings shown in the waveforms.

For a description of all processor timings, refer to the following tables:

- Table 3, "Timing and Control Register for Chip Select 0" on page 14
- Table 4, "Bit Level Definition for each of the Timing and Control Registers" on page 14
- Table 5, "Expansion Bus Configuration Register 0" on page 16
- Table 6, "Expansion Bus Configuration Register 0 Description" on page 16

Table 7. Read Timing Definitions

#	Symbol	Parameter Definition
R1	t_{AVAV}	Read cycle time
R2	t_{AVQV}	Address to output valid
R3	t_{ELQV}	CE# to output valid
R4	t_{GLQV}	OE# to output valid
R5	t_{PHQV}	RST# high to output valid
R6	t_{ELQX}	CE# to output in low-Z
R7	t_{GLQX}	OE# to output in low Z
R8	t_{EHQZ}	CE# high to output in high Z
R9	t_{GHQZ}	OE# high to output in high Z
R10	t_{OH}	Output hold from first occurring address, CE#, or OE# change

Table 8. Write Timing Definitions (Sheet 1 of 2)

#	Symbol	Parameter Definition
W1	t_{PHWL}	RST# high recovery to WE# low
W2	t_{ELWL}	CE# setup to WE# low
W3	t_{WLWH}	WE# write pulse width low
W4	t_{DVWH}	Data setup to WE# high
W5	t_{AVWH}	Address setup to WE# high
W6	t_{WHEH}	CE# hold from WE# high

Table 8. Write Timing Definitions (Sheet 2 of 2)

#	Symbol	Parameter Definition
W7	t_{WHDX}	Data hold from WE# high
W8	t_{WHAX}	Address Hold from WE# high
W9	t_{WHWL}	WE# pulse width high
W10	t_{VPWH}	VPP setup to WE# high

Note: For additional information regarding the individual components presented in this design example, consult the appropriate documents as listed in Appendix A, "Additional Information" on page 20.

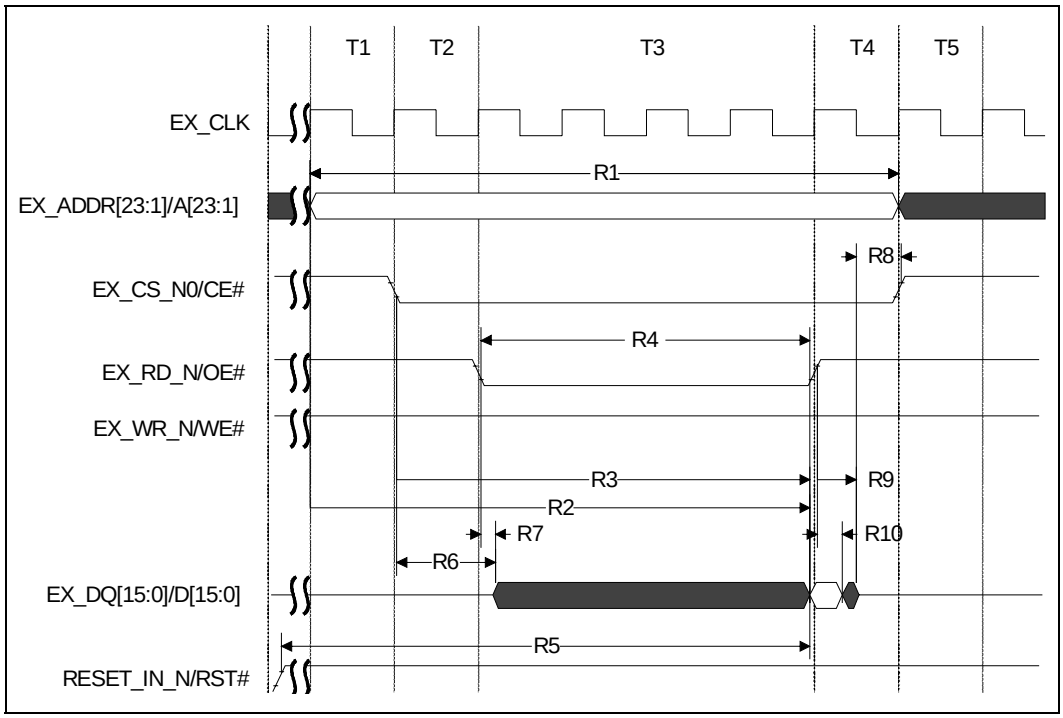
- Memory timings start with R (Read) or W (Write).
- Parameters in parentheses represent the field names of the SRAM Unit register settings, which determine the clock delays require.

6.1 Asynchronous Single-Word Read

Figure 4 shows the bus timings for an initial read following a reset.

Note: The IXP42X processor defaults to asynchronous read, so read accesses after a reset are asynchronous reads. Non-array (register) reads are also performed in this mode.

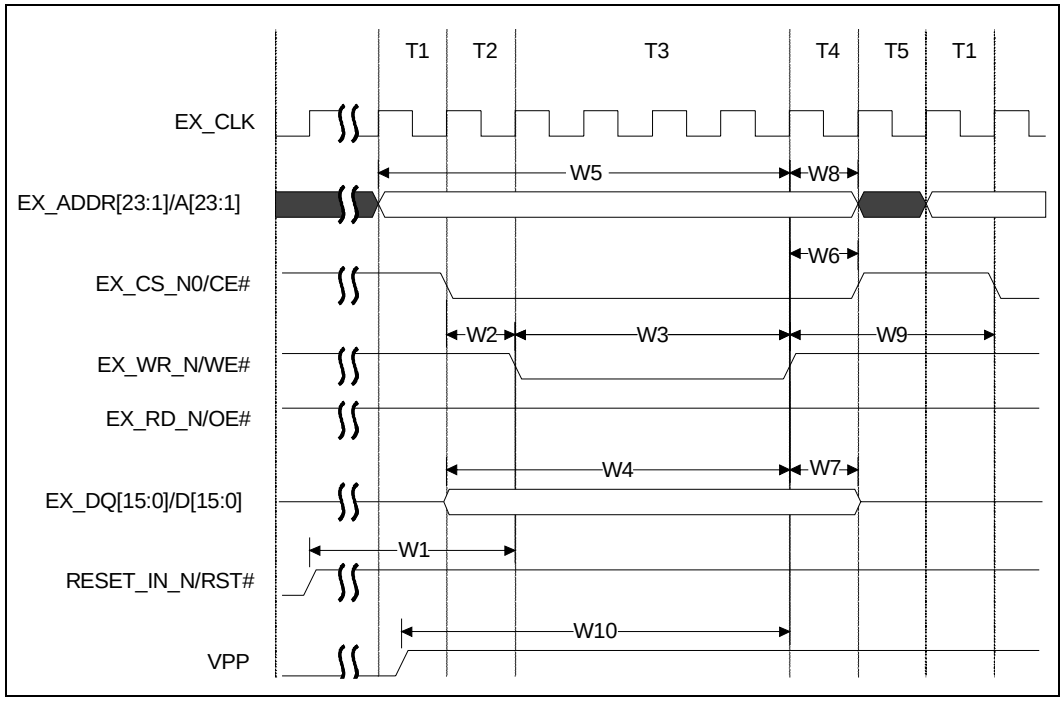
Figure 4. Asynchronous Single Word Read



6.2 Asynchronous Write

Figure 5 shows the bus timings for an asynchronous write to the P30 flash memory device. Parameters in parentheses represent the register field names of the expansion bus register settings, which determine the clock delays required.

Figure 5. Asynchronous Write



7.0 Summary

The Intel StrataFlash® Embedded Memory (P30) uses two-bit-per-cell storage technology to provide 2X the bits in 1X the space, compared to traditional single-bit-per-cell memory devices. The P30 flash device is available in a variety of packages and densities for increased flexibility, and can be used for both code and data applications where high density and low cost are required.

The P30 flash memory device provides the system designer with a flexible interface to processors, such as the IXP42X processor and the IXC1100 processor. Using these processor interfaces with the P30 flash memory device can help to decrease system costs, help to increase overall system reliability, and help to hasten product development and time-to-market.

Appendix A Additional Information

Document Number	Document/Tool
306666	*Intel StrataFlash® Embedded Memory (P30) 1-Gbit P30 Family Datasheet*
306667	*Migration Guide for Intel StrataFlash® Memory (J3) to Intel StrataFlash® Embedded Memory (P30) Application Note 812*
252479	*Intel® IXP42X Product Line of Network Processors and IXC1100 Control Plane Processor Datasheet*
252480	*Intel® IXP42X Product Line of Network Processors and IXC1100 Control Plane Processor Developer's Manual*
252817	*Intel® IXP42X Product Line of Network Processors and IXC1100 Control Plane Network Processor Hardware Design Guidelines*
292286	*Reduce Manufacturing Costs with Intel® Flash Memory Enhanced Factory Programming Application Note 738*
250260	*System Design Considerations When Designing with Intel® Flash Application Note 751*
292172	*AP-617 Application Note: Additional Flash Data Protection Using VPP, RST#, and WP#*
252702	*Intel® IXP42X Product Line of Network Processors and IXC1100 Control Plane Processor Specification Update*

Notes:

1. Call the Intel Literature Center at (800) 548-4725 to request Intel documentation. International customers should contact their local Intel or distribution sales office.
2. Visit the Intel World Wide Web home page http://www.Intel.com for technical documentation and tools.
3. For the most current information on Intel flash products, visit http://developer.intel.com/design/flash/.
4. See also the IXP42X Processor documents at http://developer.intel.com/design/network/products/npfamily/docs/ixp4xx.htm.

Appendix B Enabling P30 Flash on IXDPG425

B.1 RedBoot porting

From the software perspective, the differences between J3 and P30 are very few. The expansion bus transaction timing of P30 is faster than J3. The lock-unlock mechanism is different, P30 locks all blocks after system reset. So the security for data is better than J3.

Refer to "IXP400 RedBoot 1.94 release note" and "RedBoot 1.94 for IXDPG425",get source code and build image.

To enable functions of P30 on IXDPG425, you need to modify RedBoot by following steps:

1. Modify the expansion bus timing to support P30: In file "/packages/hal/arm/xscale/grg/current/include/grg.h"

```
#define IXP425_EXP_CS0_INIT \

 (EXP_ADDR_T(1) | EXP_SETUP_T(1) | EXP_STROBE_T(3) | EXP_HOLD_T(1) | \

  EXP_RECOVERY_T(1) | EXP_SZ_16M | EXP_WR_EN | EXP_BYTE_RD16 | EXP_CS_EN)
```

2. Modify the unlock-block mechanism in the file "/packages/devs/flash/intel/strata/current/src/flash_unlock_block.c"

Original:

```
int

flash_unlock_block(volatile flash_t *block, int block_size, int blocks)

{

    volatile flash_t *ROM;

    flash_t stat;

    int timeout = 5000000;

    int cache_on;

#ifndef CYGOPT_FLASH_IS_SYNCHRONOUS

    int i;

    volatile flash_t *bp, *bpv;

    unsigned char is_locked[MAX_FLASH_BLOCKS];

#endif


    HAL_DCACHE_IS_ENABLED(cache_on);

    if (cache_on) {

        HAL_DCACHE_SYNC();

        HAL_DCACHE_DISABLE();
```

```
    }


    // Get base address and map addresses to virtual addresses
    ROM = FLASH_P2V( CYGNUM_FLASH_BASE_MASK & (unsigned int)block );
    block = FLASH_P2V(block);


    // Clear any error conditions
    ROM[0] = FLASH_Clear_Status;


#ifdef CYGOPT_FLASH_IS_SYNCHRONOUS
    // Clear lock bit
    block[0] = FLASH_Clear_Locks;
    block[0] = FLASH_Clear_Locks_Confirm;  // Confirmation
    while(((stat = ROM[0]) & FLASH_Status_Ready) != FLASH_Status_Ready) {
        if (--timeout == 0) break;
    }
#else
    // Get current block lock state. This needs to access each
    // block on the device so currently locked blocks can be
    // re-locked.
    bp = ROM;
    for (i = 0;  i < blocks;  i++) {
        bpv = FLASH_P2V( bp );
        *bpv = FLASH_Read_Query;
        if (bpv == block) {
            is_locked[i] = 0;
        } else {
#if 8 == CYGNUM_FLASH_WIDTH
            is_locked[i] = bpv[4] & FLASH_Lock_Bit;
#else
            is_locked[i] = bpv[2] & FLASH_Lock_Bit;
# endif
        }
        bp += block_size / sizeof(*bp);
    }
```

```
        // Clears all lock bits
        ROM[0] = FLASH_Clear_Locks;
        ROM[0] = FLASH_Clear_Locks_Confirm;  // Confirmation
        timeout = 5000000;
        while(((stat = ROM[0]) & FLASH_Status_Ready) != FLASH_Status_Ready) {
            if (--timeout == 0) break;
        }

        // Restore the lock state
        bp = ROM;
        for (i = 0;  i < blocks;  i++) {
            bpv = FLASH_P2V( bp );
            if (is_locked[i]) {
                *bpv = FLASH_Set_Lock;
                *bpv = FLASH_Set_Lock_Confirm;  // Confirmation
                timeout = 5000000;
                while(((stat = ROM[0]) & FLASH_Status_Ready) != FLASH_Status_Ready) {
                    if (--timeout == 0) break;
                }
            }
            bp += block_size / sizeof(*bp);
        }
    #endif  // CYGOPT_FLASH_IS_SYNCHRONOUS

        // Restore ROM to "normal" mode
        ROM[0] = FLASH_Reset;

        if (cache_on) {
            HAL_DCACHE_ENABLE();
        }

        return stat;
    }
```

Change to:

```
int
flash_unlock_block(volatile flash_t *block, int block_size, int blocks)
{
    volatile flash_t *ROM;
    flash_t stat;
    int timeout = 5000000;
    int cache_on;
#ifndef CYGOPT_FLASH_IS_SYNCHRONOUS
    int i;
    volatile flash_t *bp, *bpv;
    unsigned char is_locked[MAX_FLASH_BLOCKS];
#endif

    HAL_DCACHE_IS_ENABLED(cache_on);
    if (cache_on) {
        HAL_DCACHE_SYNC();
        HAL_DCACHE_DISABLE();
    }
    // Get base address and map addresses to virtual addresses
    ROM = FLASH_P2V( CYGNUM_FLASH_BASE_MASK & (unsigned int)block );
    block = FLASH_P2V(block);

    // Clear any error conditions
    ROM[0] = FLASH_Clear_Status;

    // Clear lock bit
    *block = FLASH_Clear_Locks;
    *block = FLASH_Clear_Locks_Confirm;  // Confirmation
while(((stat = *block) & FLASH_Status_Ready) != FLASH_Status_Ready) {
        if (--timeout == 0) break;
    }
    // Restore ROM to "normal" mode
    ROM[0] = FLASH_Reset;
```

```
    if (cache_on) {
        HAL_DCACHE_ENABLE();
    }

    return stat;
}
```

B.1.1 Using Redboot to update RedBoot

Because P30 locks itself after system reset, we need to change the "Using RedBoot to update RedBoot" procedure.

In currently design, RedBoot only locks "RedBoot config" partition and does not lock other partitions of the flash. If you want to lock contents of the flash, you need to issue the command "fis lock –f <flash address> -l <length> -n <name>". To unlock the flash contents, you need to issue the command "fis unlock –f <flash address> –l <length> -n <name>".

Each time P30 locks itself after system reset. So if you want to program contents to P30, you need to manually do an unlock command.

Procedure for "RedBoot update RedBoot" on P30:

1. load –v redbootRam.srec; go // load a RAM-version RedBoot and run it.
2. fis unlock –f 0x50000000 -l 0x40000 or <redboot> // unlock flash blocks
3. load –r –v –b <address> <file name> // load ROM-version RedBoot to a place of DRAM.
4. fis create redboot -b <address> // program RedBoot to flash.

B.1.2 Linux MTD

In current validation platform, we use a 32MB flash on a 16MB chip select space. After linux OS boots up, MTD driver uses CFI to get the flash size information. Then it will try to get the partition information. Thus cause the system hung because MTD reads a 32MB flash and will try to get partition information on address 0x1fe0000. To solve this special case on our platform, we hard-coded the flash size into 16MB. In file "linux/drivers/mtd/redboot.c", function parse_redboot_partitions(…). Add the following code in the beginning of the function:

```
master->size = 0x1000000;
```

This makes the partition-parse function treat 32 MB flash as a 16 MB flash. Then the flash partition can be successfully recognized by MTD and file system.

For more detail about MTD, please see http://www.linux-mtd.infradead.org/ and http://www.enseirb.fr/~kadionik/embedded/uclinux/mtd/howto_mtd.html.

B.2 Using EPI ICE to Program redboot binary code to P30

1. Get Flash Programming Utility from MAJIC, the version is “0380-0197-10 Rev 2.1.0, March 10, 2003”.
2. Use the IXDP425 as the “EPI ICE target” to start ICE.
3. Add the following settings are in ixdp425.cmd:

```
ew c15_15 = 1           // Enable access to Coprocessor 0

ew c15_1  = 0x1078// Enable instruction cache, mmu off
```

4. Load MonteJade expansion bus timing configuration value.

```
fr m c:\chipselect.bin 0xc4000000
```

5. Load flash program *flash.axf*, press “GO” to start program. Choosing the following options in the program menu:

   (1) Flash Type \ 31: 28F128J3 x16 16M

   (2) Image File Name \ Change to RedBoot file name which you want to use.

   (3) Base Address \ 0x50000000

   (5) Image size\(5) Use File Length : 0x0003d2d0

   (7) Program Menu\(2) Erase Image, Program Image, Verify

Please Select an Option: 2

You Selected ERASE Image, PROGRAM, Verify, Are You SURE? (Y/N):Y

Erasing Flash Image Sectors...

Verifying Image Erased...

Image Erased : PASSED

Programming Device...

Done!

Verifying Flash Image...

Flash Image : PASSED