XPedite8150

Intel® Atom™ E3800 Series Processor-Based Rugged COM Express® Module

XPedite8150 User's Manual

Revision C

X-ES
Extreme Engineering Solutions

...Always Fast

Extreme Engineering Solutions
3225 Deming Way, Suite 120 • Middleton, WI 53562
Phone: 608.833.1155 • Fax: 608.827.6171
sales@xes-inc.com • http://www.xes-inc.com

XPedite8150 User's Manual

Revision History

Revision	Date	Remark
A	June 6, 2014	Initial release
B	May 9, 2016	Section 1.5: added for firmware access Section 2.2: specified rail voltages Section 3.8: expanded GPIO descriptions Chapter 6: added for coreboot
C	June 2, 2016	Table 3.14: corrected note/reference for pins A67 and A85

Table of Contents

List of Figures

List of Tables

Safety Information

ESD Warning

Before handling any product described in this manual, please remember that electrostatic discharge (ESD) can easily damage electrical components and potentially result in product failure. The installer must be properly grounded at all times, else static charge will accumulate and may cause ESD damage.

Please adhere to the following guidelines to ensure the safety of your product:

- **Handle the product only when absolutely necessary.**
- **When handling the product, wear a grounding wrist strap at all times.**
- **Hold the product only by its edges. Do not touch any electrical components on the printed circuit board (PCB).**
- **Place the product only on a grounded ESD-dissipative mat. Simply placing the product on a static-shielding bag offers little to no ESD protection.**
- **Never place the product on metal or other conductive surfaces.**
- **If possible, store the product in an ESD-safe bag or clamshell when it is not in use.**

Manual Conventions

This manual uses the following conventions for typography and syntax:

`0x0`	Denotes a hexadecimal number.
`0b0`	Denotes a binary number.
`#`	When used as a postfix, denotes an active-low signal (e.g., RESET#). It also denotes the negative half of a signal pair (e.g., RX/RX# or TX/TX#).
`-`	Denotes an unused (unconnected) pin.
`n/a`	Denotes a pin or reference that is not applicable.
`TBD`	Denotes an item that is to-be-determined.

Overview

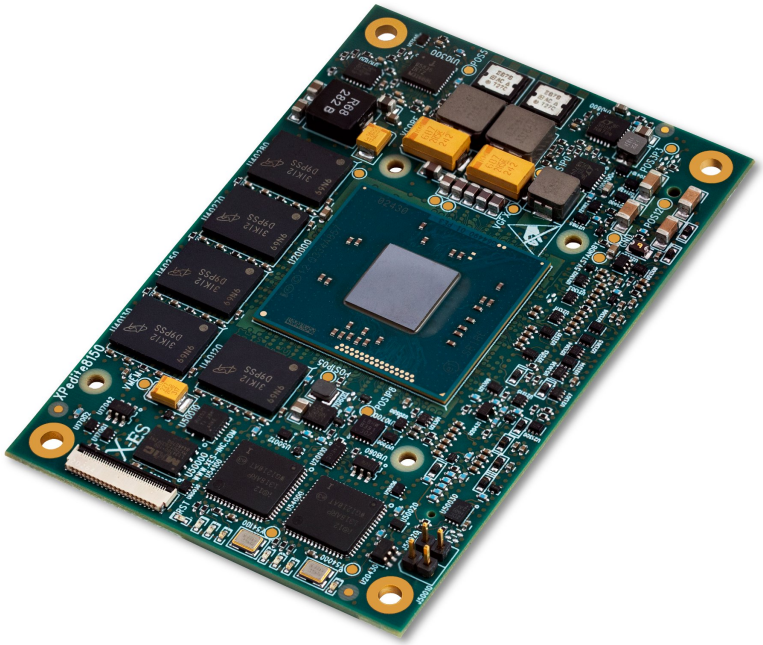
The XPedite8150 is a Rugged COM Express® module based on the Intel® Atom™ E3800 series of processors. It is compliant with the COM Express® Mini form factor (55 mm x 84 mm) and provides up to 4 GB of DDR3 ECC SDRAM. The XPedite8150 also supports an enhanced Type 10 pinout with one Dual-Mode DisplayPort, one Embedded DisplayPort, and two Gigabit Ethernet interfaces.

With built-in test (BIT) support, true configuration and obsolescence management, Class III PCB fabrication and assembly, environmental qualification per MIL-STD-810, as well as many other features, the XPedite8150 is designed and tested for maximum reliability in the most demanding environments and applications that require long life cycles.

Wind River VxWorks and Linux Board Support Packages (BSPs), as well as Microsoft Windows drivers, are available for the XPedite8150. It also supports the open source coreboot bootloader, powered by Intel®'s Firmware Support Package (FSP), to enable ultra-fast boot times and drastically simplify system security.

The Intel® Atom™ E3800 series processors are low-power system-on-chip (SoC) processors with integrated graphics and support for up to four cores operating at up to 1.91 GHz. Along with best-in-class performance-per-watt, the E3800 family supports extremely low operating temperatures, and its power-efficient 22 nm technology enables operation in the most demanding high-temperature environments. The XPedite8150 supports the E3845 processor in standard configurations and can be built to support the E3827, E3826, E3825, E3815, and E3805. The E3800 series is the 4th generation Atom™ processor from Intel® and was formerly known as the Bay Trail-I platform and Valleyview processor.

Figure 1.1: XPedite8150 Photo



1.1 Features

The following is a brief summary of the XPedite8150's features:

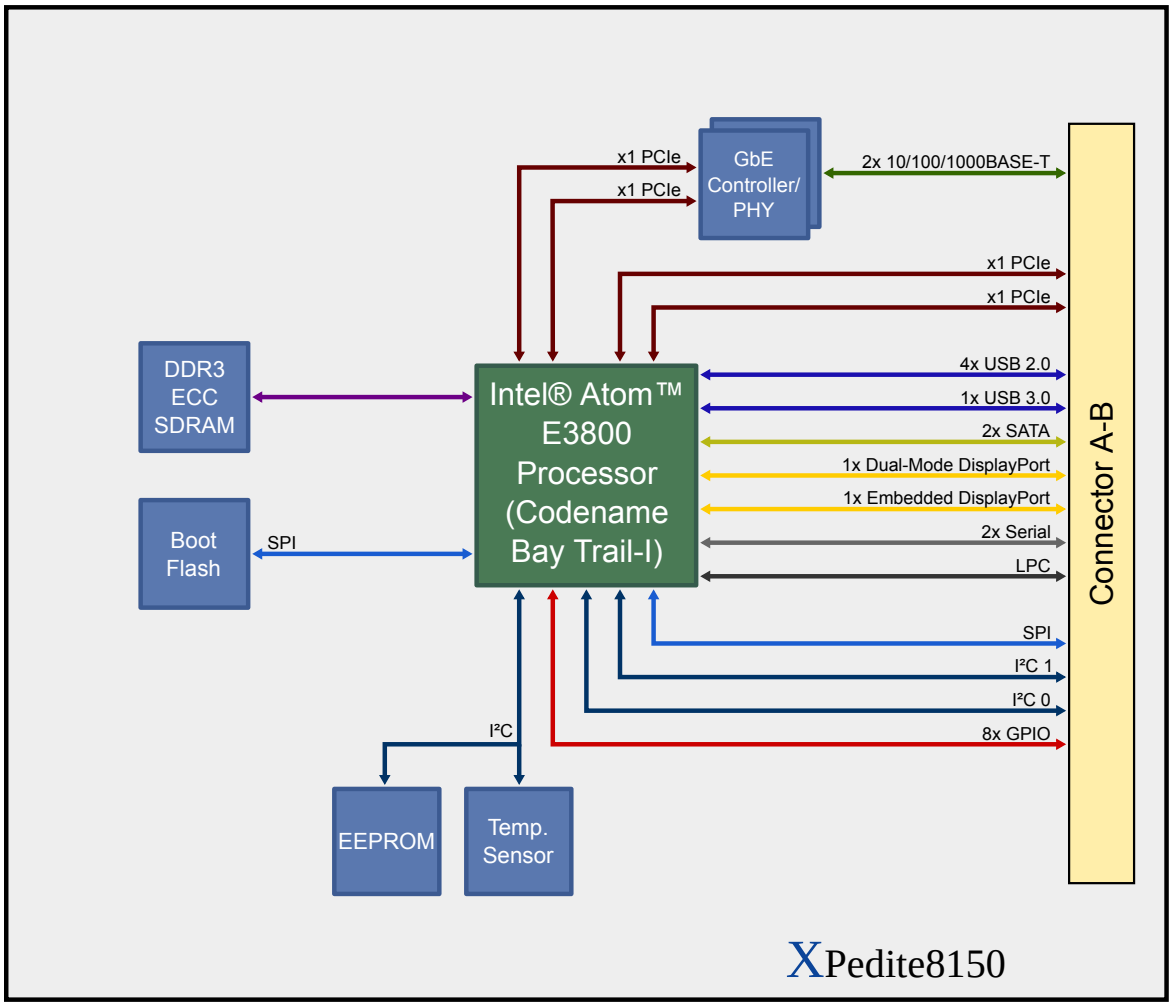
- Supports Intel® Atom™ E3800 family processors (formerly Bay Trail-I)
- COM Express® Mini form factor with ruggedization enhancements
- COM Express® enhanced Type 10 pinout
- Conduction- or air-cooled
- Extended shock and vibration tolerance
- Up to 4 GB of DDR3-1333 ECC SDRAM
- One Dual-Mode DisplayPort interface and one Embedded DisplayPort interface
- Two x1 PCI Express interfaces
- Two Gigabit Ethernet ports
- Two serial ports
- Four USB 2.0 ports
- One USB 3.0 port
- Two SATA ports

- Intel® Platform Trust Technology (PTT) providing optional Trusted Platform Module (TPM) support
- coreboot bootloader, powered by Intel®'s Firmware Support Package (FSP)
- Wind River VxWorks BSP
- Linux BSP
- Microsoft Windows drivers
- Contact factory for availability of Green Hills INTEGRITY, QNX Neutrino, and LynuxWorks LynxOS BSPs

1.2 Block Diagram

Figure 1.2 provides a high-level overview of the XPedite8150.

Figure 1.2: General Block Diagram



1.3 Available Accessories

Table 1.1 provides a list of X-ES accessories available for the XPedite8150.

Table 1.1: Available Accessories

Part Number	Description
90071685-2	**XTend209 Debug Module.** The XTend209 is an optional firmware programming/debug module for use with the XPedite8150.
90072175	**XPand1403.** COM Express Type 10 development platform for the XPedite8150.

1.4 Available Software

Table 1.2 describes software solutions offered by X-ES for the XPedite8150. Additional operating systems may be supported upon request. Please contact X-ES for availability.

Table 1.2: Available Software

Software*	Description
VxWorks Driver and BSP	Support for Wind River VxWorks
Windows Drivers	Support for Microsoft Windows
X-ES Enterprise Linux BSP	Support for Linux

*Contact X-ES for ordering details.

1.5 Accessing Firmware

Basic operational tests are available for the XPedite8150 by accessing the firmware via a console serial connection.

1. On your personal computer workstation, open a terminal emulator program, such as Tera-Term, HyperTerminal, or minicom, and connect to the COM port.

   The XPedite8150 defaults to using a baud rate of 115200, with 8-bits/character, one stop bit, no parity, no flow control, and a terminal size of 80x25 characters.

2. Power-up the XPedite8150. A command prompt or menu will appear on the console screen, allowing for basic access to the firmware.

3. Check for output on the serial console's screen.

   If you can see the output, compare it to the examples described in the XPedite8150's boot firmware documentation.

If you are unable to see the expected output or are experiencing other problems, check your connections and refer to Section 1.8, for assistance.

4. Once you have confirmed the serial console's output, see Section 1.6, to proceed with more advanced functions.

1.6 Booting an Operating System

X-ES provides various Board Support Packages (BSPs) to support different operating systems. Please consult the XPedite8150's firmware documentation and related BSP documentation (see Section 1.10.2) for details on booting the desired operating system. If you have questions about the availability of BSPs or need further assistance, please see Section 1.9.

1.7 Service Information

If you need to return the XPedite8150 to X-ES for service, please e-mail <support@xes-inc.com> or call 608-833-1155 and ask for a Return Merchandise Authorization (RMA) number. Be prepared to supply the XPedite8150 serial number, the reason for return, and your original purchase order number.

When returning hardware that is out of warranty, billing information is required as well. Contact X-ES for any warranty-related questions.

When returning the XPedite8150, be sure to enclose it in an anti-static bag or clamshell, such as the original shipping material. Send the XPedite8150, pre-paid, to:

Extreme Engineering Solutions, Inc.
3225 Deming Way, Suite 120
Middleton, WI 53562
USA

1.8 Basic Troubleshooting

In the event that the XPedite8150 does not seem to be working properly, follow these troubleshooting steps:

1. Check that the XPedite8150 is fully seated in the host.
2. Check that all required power cables are connected.
3. Check that the power supply is supplying power at the correct voltages. If possible, use an oscilloscope to look for excessive power supply ripple or noise.
4. Check that any serial cables or network cables in use are securely connected.

5. If the XPedite8150 does not boot, check the status of the red Reset LED on the XPedite8150 (see Section 2.4). When lit, it indicates the processor is in reset.

1.9 Technical Support

The X-ES SupportNet web site at https://support.xes-inc.com allows registered users to access product information, documentation updates, software downloads, and technical support. X-ES customers can request login access by following the New User Sign-Up link.

If you have followed the entire troubleshooting procedure in Section 1.8 and are still experiencing problems getting the XPedite8150 to work properly, contact X-ES using any of the following methods:

- SupportNet: https://support.xes-inc.com
- E-mail: <support@xes-inc.com>
- Phone: 608-833-1155

Before contacting X-ES support, please be prepared to supply the following information, where applicable:

- XPedite8150 serial number

Tip

To determine the serial number of your XPedite8150, look for the white or light-green sticker attached to the PCB. The serial number will be the eight-digit number directly below the part number (90010360).

- Manufacturer and model number of XPedite8150's host chassis or enclosure, as applicable
- Manufacturer and model number of any other boards in the system, as applicable
- Any custom modifications to the XPedite8150
- State of the red Reset LED (see Section 2.4), if the XPedite8150 is not booting
- Serial console screen output of the error, if applicable

1.10 Additional Information

Please refer to the documents listed within this section for additional information about the XPedite8150 and its components that extends beyond the scope of this manual.

1.10.1 Standards

The technical information in this manual describes the unique features of the XPedite8150. It is assumed that the reader has familiarity with the standard interfaces and devices incorporated into the XPedite8150 that are not described in this manual.

Table 1.3 lists industry standards that apply to the XPedite8150. Please see these documents for more information.

Table 1.3: Applicable Standards

Standard	Document	Author	Revision
-	*COM Express™ Carrier Design Guide*	PICMG	Rev. 2.0; December 6, 2013
PICMG COM.0	*COM Express® Module Base Specification*	PICMG	Rev. 2.1; May 14, 2012

1.10.2 Other X-ES Documents

Table 1.4 lists additional documents that are available from X-ES. Contact X-ES to obtain any of these documents.

Table 1.4: Additional X-ES Documents

Document	Author
XPedite8150 VxWorks Manual	X-ES
XPedite8150 Linux Manual	X-ES

Printed Circuit Board

This chapter provides detailed information about the XPedite8150 printed circuit board (PCB). This information includes physical characteristics, power and environmental requirements, component maps, jumper definitions, and LED descriptions.

2.1 Physical Specifications

Table 2.1 describes the physical specifications of the XPedite8150.

Table 2.1: Dimensions and Weight

Attribute	Value
Form Factor	COM Express (Mini)
Length	84 mm
Width	55 mm
Stacking Height	5 mm *
Weight	Contact X-ES†

*with build option to 8 mm

†with R1 heat sink

2.2 Power Requirements

Table 2.2 describes the operating voltage specifications for the XPedite8150.

Table 2.2: Input Power Specifications

Power Rail*	Minimum	Nominal	Maximum
+12V	11.4 V	12.0 V	12.6 V

*The XPedite8150 does not support the wide input voltage range on +12V introduced in PICMG COM.0 R2.1. If you require different input voltages, please contact X-ES for assistance.

The XPedite8150 does not use +5V_Standby as a power source. If supplied, it will be used only as an alternate source for RTC power.

Caution

Beginning with revision 2.0 of the COM.0 specification, some power pins are re-purposed. The Type 10 pinout re-maps nine pins that were VCC_12V supply pins on Type 1, 2, 3, 4, and 5 modules. To avoid damage and ensure compatibility, the XPedite8150 should be used only with an X-ES-specified carrier.

The XPedite8150 utilizes the COM Express VCC_RTC pin to provide power to the Real Time Clock (RTC) for clock retention. The XPedite8150 also employs a backup capacitor for approximately 30 seconds of backup. The estimated maximum power draw for the +3.3-volt Auxiliary power rail average current is 6 µA at 27°C.

2.3 Environmental Requirements

The XPedite8150's conformance to a particular level is determined by rigorous analysis and testing. All qualification testing may not be completed at the time of this writing. Specific product configurations may be required to meet high-temperature operational requirements for certain levels. The low temperature limits for certain products may be below the operating temperature limits defined by the manufacturers of components used within the product assembly. In conjunction with performing qualification testing at these lower levels, X-ES also screens every product that falls into this category to verify operation at these lower temperatures.

Please contact X-ES for the most current information regarding this product.

Level 1 ruggedization, described in Table 2.3, is for standard commercial applications.

Table 2.3: Standard Air-Cooled - Level 1

Category	Specification
Operating Temp.	0 to +55°C ambient, 300 LFM, per MIL-STD-810F Method 501.4 Procedure II and Method 502.4 Procedure II
Storage Temp.	-40 to +85°C ambient*
Operating Vibration	0.002 g²/Hz, 1 hour per axis from 5 to 2000 Hz
Operating Shock	20 g, 11 ms sawtooth, per MIL-STD-810F Method 516.5 Procedure I
Humidity	0 to 95% non-condensing, per MIL-STD-810F Method 507.4

*If the manufacturer's advertised storage temperature limit for any component used within this product does not meet these levels, these tests will be performed during qualification.

Level 3 ruggedization, described in Table 2.4, is for applications requiring an extended thermal range, as well as increased shock and vibration tolerances.

Table 2.4: Rugged Air-Cooled - Level 3

Category	Specification
Operating Temp.	-40 to +70°C ambient, 600 LFM, per MIL-STD-810F Method 501.4 Procedure II and Method 502.4 Procedure II
Storage Temp.	-55 to +105°C ambient per MIL-STD-810F Method 501.4 Procedure I and Method 502.4 Procedure I*
Operating Vibration	0.04 g²/Hz (maximum), 1 hour per axis from 5 to 2000 Hz†
Operating Shock	30 g, 11 ms sawtooth, per MIL-STD-810F Method 516.5 Procedure I
Humidity	0 to 95% non-condensing, per MIL-STD-810F Method 507.4

*If the manufacturer's advertised storage temperature limit for any component used within this product does not meet these levels, these tests will be performed during qualification.

†5-100 Hz, PSD increasing at 3 dB/octave; 100-1000 Hz, PSD 0.04 g²/Hz; 1000-2000 Hz, PSD decreasing at 6 dB/octave

Level 5 ruggedization, described in Table 2.5, is for applications requiring an extended thermal range beyond that of the previous levels.

Table 2.5: Conduction-Cooled - Level 5

Category	Specification
Operating Temp.	-40 to +85°C board rail surface temperature, per MIL-STD-810F Method 501.4 Procedure II and Method 502.4 Procedure II
Storage Temp.	-55 to +105°C ambient per MIL-STD-810F Method 501.4 Procedure I and Method 502.4 Procedure I*
Operating Vibration	0.1 g^2/Hz (maximum), 1 hour per axis from 5 to 2000 Hz†
Operating Shock	40 g, 11 ms sawtooth, per MIL-STD-810F Method 516.5 Procedure I
Humidity	0 to 95% non-condensing, per MIL-STD-810F Method 507.4

*If the manufacturer's advertised storage temperature limit for any component used within this product does not meet these levels, these tests will be performed during qualification.

†5-100 Hz, PSD increasing at 3 dB/octave; 100-1000 Hz, PSD 0.1 g^2/Hz; 1000-2000 Hz, PSD decreasing at 6 dB/octave

2.4 Component Maps

Figure 2.1: Front Component Map (Revision B)

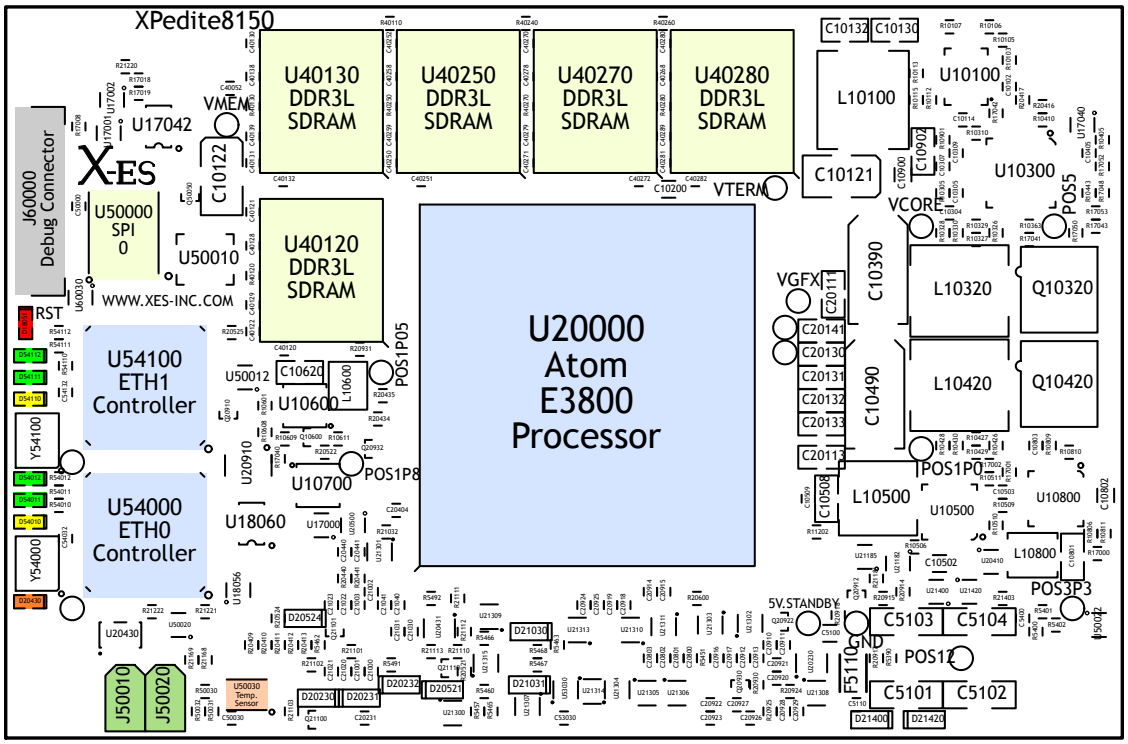
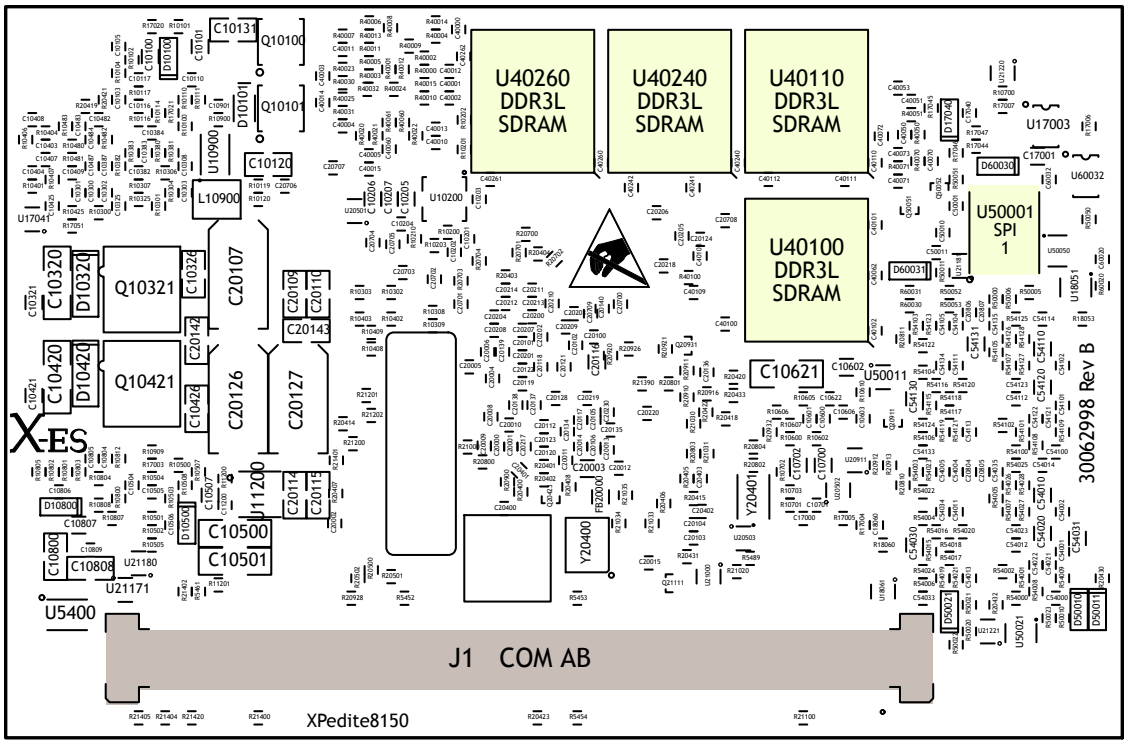


Figure 2.2: Back Component Map (Revision B)



2.5 Jumpers

Table 2.6 describes the jumpers used on the XPedite8150.

Table 2.6: Jumper Functions

Jumper	Description
J50010	**Boot Device Select.** By default, the XPedite8150 will boot from the primary SPI flash device. Installing this jumper will swap the chip selects going to the primary and secondary SPI flash devices, causing the board to boot from the secondary SPI flash device.
J50020	**Non-Volatile Memory Write Enable.** Removing this jumper while the NVM_WP input signal is high will write-protect all non-volatile memory on the XPedite8150 board. Placing this jumper will override the NVM_WP input signal.

2.6 LEDs

Table 2.7 describes the LEDs used on the XPedite8150.

Table 2.7: LED Definitions

LED	Color	Description
D18051	Red	**Reset.** When lit, indicates Intel E3800 is in reset.
D20430	Orange	**PROCHOT.** When lit, indicates E3800 has reached it's T_JMax temperature (110°C) and performance may be reduced.
D54010	Yellow	**ETH0 Link/Activity.** When lit, indicates Ethernet Port 0 is linked. Flashing indicates activity.
D54011	Green	**ETH0 100 Link.** When lit, indicates Ethernet Port 0 is linked at 100 Mbit/s.
D54012	Green	**ETH0 1000 Link.** When lit, indicates Ethernet Port 0 is linked at 1000 Mbit/s.
D54110	Yellow	**ETH1 Link/Activity.** When lit, indicates Ethernet Port 1 is linked. Flashing indicates activity.
D54111	Green	**ETH1 100 Link.** When lit, indicates Ethernet Port 1 is linked at 100 Mbit/s.
D54112	Green	**ETH1 1000 Link.** When lit, indicates Ethernet Port 1 is linked at 1000 Mbit/s.

2.7 Temperature Sensors

Table 2.8 describes the temperature sensors used on the XPedite8150. See Section 2.4 for physical locations of these sensors on the XPedite8150.

Table 2.8: Temperature Sensors

Sensor	Device / Function	Source	Address
U53030	TMP102; ambient temperature	CPU SMBus Controller or I²C Controller 7	0x48

COM Express Interface

This chapter describes details associated with the XPedite8150's COM Express® interface. The XPedite8150 implements an X-ES-specific Type 10 COM Express interface, accessible via a 220-pin socket connector. A number of signal groups are present on this connector, including SATA, PCI Express, Ethernet, USB, SPI, GPIO, serial, I²C, and other compatible Type 10 COM Express interfaces.

Caution

Only install the XPedite8150 on a carrier card that specifically supports the X-ES-enhanced Type 10 pinout. Installing the XPedite8150 on an unsupported carrier can result in electrical damage to the module and/or carrier.

3.1 SATA

The XPedite8150 provides two SATA 3Gb/s interfaces from the E3800 processor over the COM Express J1 (AB) connector. Table 3.1 describes the SATA signals.

Table 3.1: SATA Signals

Signal *	Description
SATA0_RX/ SATA0_RX#	**SATA 0 Receive.** Connected to E3800 port 0
SATA0_TX/ SATA0_TX#	**SATA 0 Transmit.** Connected to E3800 port 0
SATA1_RX/ SATA1_RX#	**SATA 1 Receive.** Connected to E3800 port 1

Table 3.1: SATA Signals *(continued)*

Signal *	Description
SATA1_TX/ SATA1_TX#	**SATA 1 Transmit.** Connected to E3800 port 1
SATA_ACT#	**SATA Activity.** Low represents activity on either SATA 0 or SATA 1

*If the SATA interface is unused, these signals can be left unconnected.

3.2 PCI Express

The XPedite8150 provides a PCI Express (PCIe) interface over the COM Express J1 (AB) connector utilizing the signals shown in Table 3.2. The default configuration provides two x1 PCIe links, though they can be combined into a single x2 link.

Table 3.2: PCI Express Signals

Signal *	Description
REFCLK/REFCLK#	**PCIe Reference Clock.** Reference clock output
PE*n*_TX/PE*n*_TX#	**PCIe Transmit.** Differential transmit pair *n*, connected to E3800 lane *n*.
PE*n*_RX/PE*n*_RX#	**PCIe Receive.** Differential receive pair *n*, connected to E3800 lane *n*.

*If the PCI Express interface is unused, these signals can be left unconnected.

3.3 Ethernet

The XPedite8150 provides two Ethernet ports via the COM Express J1 (AB) connector. Both function as analog (10/100/1000BASE-T) ports.

Table 3.3: Ethernet Signal Descriptions

Signal *	Description
ETH*n*_DA/ETH*n*_DA#	**Ethernet DA (10/100/1000BASE-T).** Port *n* bi-directional transmit and receive signals for 1000BASE-T operation. Transmit (output) for 10/100BASE-T operation.
ETH*n*_DB/ETH*n*_DB#	**Ethernet DB (10/100/1000BASE-T).** Port *n* bi-directional transmit and receive signals for 1000BASE-T operation. Receive (input) for 10/100BASE-T operation.

Table 3.3: Ethernet Signal Descriptions *(continued)*

Signal *	Description
ETH*n*_DC/ETH*n*_DC#	**Ethernet DC (10/100/1000BASE-T).** Port *n* bi-directional transmit and receive signals for 1000BASE-T operation. Not used for 10/100BASE-T operation.
ETH*n*_DD/ETH*n*_DD#	**Ethernet DD (10/100/1000BASE-T).** Port *n* bi-directional transmit and receive signals for 1000BASE-T operation. Not used for 10/100BASE-T operation.
ETH*n*_LINK100#	**Ethernet Link 100 Status.** Indicates Ethernet Port *n* is linked at 100 Mbit/s
ETH*n*_LINK1000#	**Ethernet Link 1000 Status.** Indicates Ethernet Port *n* is linked at 1000 Mbit/s
ETH*n*_ACT#	**Ethernet Activity Status.** Indicates Ethernet Port *n* is active.
ETH*n*LINK#	**Ethernet Link Status.** Indicates Ethernet Port *n* is linked.

*If the Ethernet interface is unused, these signals can be left unconnected.

3.4 USB

The XPedite8150's E3800 provides four USB 2.0 interfaces and one SuperSpeed USB 3.0 interface at the COM Express J1 (AB) connector. Table 3.4 describes the USB signals.

Table 3.4: USB Signals

Signal *	Description
USB01_OC#	**USB Over-Current.** USB over-current status (shared by ports 0 and 1)
USB23_OC#	**USB Over-Current.** USB over-current status (shared by ports 2 and 3)
USB45_OC#/ USB67_OC#	**Unused.** No connection
USBSS_RX0/ USBSS_RX0#	**SuperSpeed USB Data RX.** Port 0 SuperSpeed USB data receive
USBSS_TX0/ USBSS_TX0#	**SuperSpeed USB Data TX.** Port 0 SuperSpeed USB data transmit
USB*n*_D/USB*n*_D#	**USB Data.** Port *n* USB data

*If the USB interface is unused, these signals can be left unconnected.

3.5 Serial

The XPedite8150 provides two serial interfaces at the COM Express J1 (AB) connector. Both interfaces are LVTTL. The receive lines are 5-volt tolerant. For the transmit lines, the absolute maximum is 4.6 V. Figure 3.1 provides an overview of the serial interface. Table 3.5 describes the serial signals.

Figure 3.1: Serial Interface Block Diagram

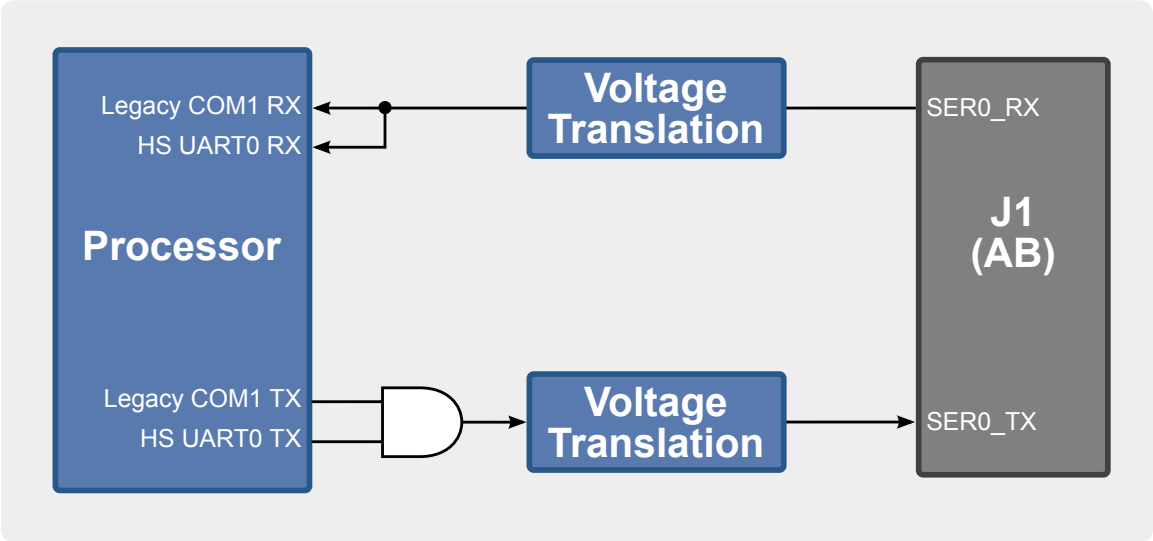


Table 3.5: Serial Signals

Signal *	Description
SER0_RX	**Serial 0 Receive.** Connected to E3800 HSUART port 0 and legacy COM1 UART
SER0_TX	**Serial 0 Transmit.** Connected to E3800 HSUART port 0 and legacy COM1 UART
SER1_RX	**Serial 1 Receive.** Connected to E3800 HSUART port 1
SER1_TX	**Serial 1 Transmit.** Connected to E3800 HSUART port 1

*If the Serial interface is unused, these signals can be left unconnected.

3.6 SPI

The Serial Peripheral Interface (SPI) connects to the E3800's general-purpose SIO_SPI controller, rather than the boot flash SPI interface, allowing it to be used with a variety of devices, such as microcontrollers, serial EEPROMs, and SPI-CAN transceivers. The XPedite8150 employs on-board voltage translation for the SPI interface for 3.3 V operation. Table 3.6 describes the SPI signals.

Table 3.6: SPI Signals

Signal *	Description
SPI_CLK	**SPI Clock.**
SPI_CS#	**SPI Chip Select 0.**
SPI_EN#	**SPI Enable.**
SPI_MISO	**SPI Master Input, Slave Output.**
SPI_MOSI	**SPI Master Output, Slave Input.**
SPI_PWR	**SPI Power.** Fused 3.3 V output from the XPedite8150

*If the SPI is unused, these signals can be left unconnected.

3.7 I²C and SMBus

The XPedite8150 provides two inter-integrated circuit (I²C) interfaces at the COM Express J1 (AB) connector. Figure 3.2 shows the I²C interfaces, and Table 3.7 describes the signals.

Figure 3.2: I²C and SMBus Interface Block Diagram

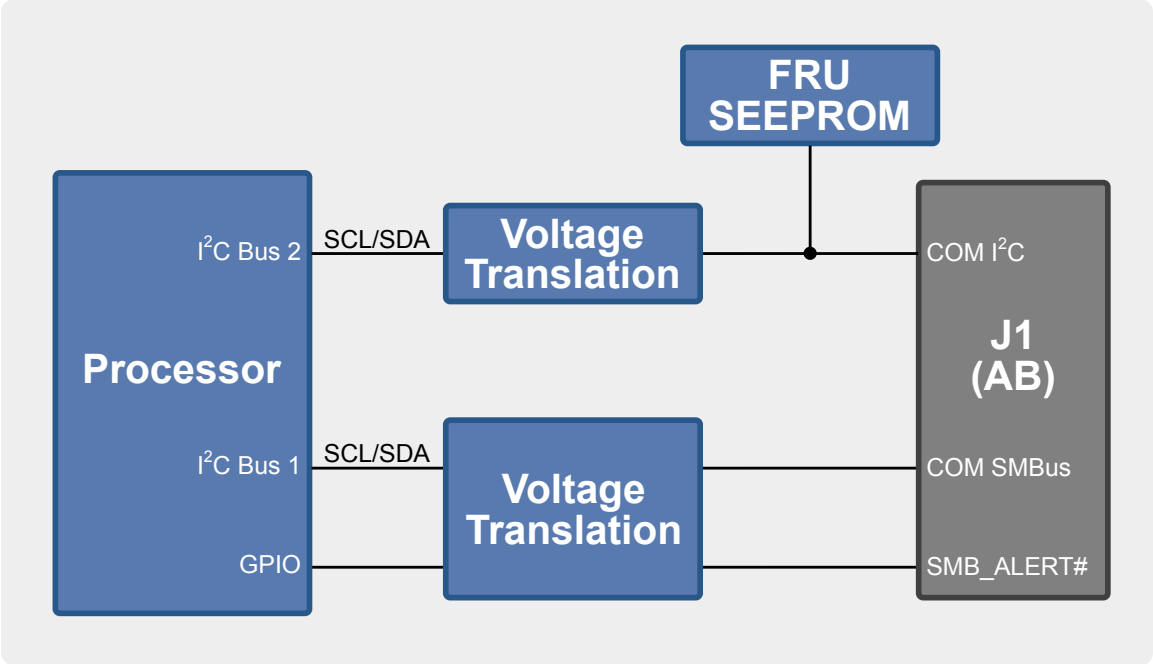


Table 3.7: I²C and SMBus Signals

Signal *	Description
I2C0_SCL	**Serial Clock.** Serial clock for E3800 I²C bus 1. Connected to E3800 through voltage translation
I2C0_SDA	**Serial Data.** Serial data for E3800 I²C bus 1. Connected to E3800 through voltage translation
I2C1_SCL	**Serial Clock.** Serial clock for E3800 I²C bus 2. Connected to E3800 through voltage translation
I2C1_SDA	**Serial Data.** Serial data for E3800 I²C bus 2. Connected to E3800 through voltage translation
SMB_ALERT#	**SMBus Alert.** Connected to GPIO on the E3800

*If the I²C interface is unused, these signals can be left unconnected.

3.8 GPIO

The General-Purpose Input/Output (GPIO) interface consists of eight pins. Table 3.8 describes the GPIO signals.

Caution

To avoid damaging the processor, do not exceed 3.3 V on any GPIO pin.

Table 3.8: GPIO Signals

Signal *	Description
GPI	**General-Purpose Inputs.** Connects to the E3800 processor through input-only voltage translation (see pinouts in Table 3.14)
GPIO	**General-Purpose Inputs/Outputs.** Connects to the E3800 processor through bidirectional voltage translation (see pinouts in Table 3.14)
GPO	**General-Purpose Outputs.** Connects to the E3800 processor through output-only voltage translation (see pinouts in Table 3.14)

*If the GPI, GPIO, or GPO signal is unused, it can be left unconnected.

Numerous pins are dedicated to GPIO, as well as standard control. The E3800 does not directly support 3.3 V signaling, so in most cases voltage translation is required. Three voltage translation schemes are supported:

Input-only voltage translation — Input-only pins use the Texas Instruments SN74AUPG97 logic gate as a buffer to drop the incoming signal to the appropriate voltage level. (See SN74AUPG97 web page.)

Output-only voltage translation — Output-only pins use the Texas Instruments SN74AUPG97 logic gate to translate the voltage to 3.3 V. (See SN74AUPG97 web page.)

Bidirectional voltage translation — Bidirectional pins use the Texas Instruments TXS0101 voltage translator. (See TXS0101 web page.)

Be careful when using bidirectional pins as outputs. Do not use pull-down resistors. Only use very weak (less than 10 kΩ), since the translators themselves include pull-up resistors (approximately 10 kΩ) to 3.3 V. The SoC and bidirectional voltage translator combination only can sink a very small amount of current (typically less than 0.5 mA). Always buffer signals before using them as an output of any sort.

3.9 Graphics

The XPedite8150 provides a dual-mode DisplayPort interface, as well as an embedded DisplayPort (eDisplayPort) interface. Table 3.9 shows the dual-mode DisplayPort signals, and Table 3.10 shows the eDisplayPort signals.

Table 3.9: Dual-Mode DisplayPort Signals

Signal *	Description
DDP0_AUX/ DDP0_AUX#	**AUX Data.** Auxiliary data pair for dual-mode DisplayPort interface
DDP0_HPD	**Hot Plug Detect.** Hot plug detection input for dual-mode DisplayPort interface
DDP0_[0:3]/ DDP0_[0:3]#	**Data Pairs.** Data ports 0-3 for dual-mode DisplayPort interface
DDP0_AUX_SEL	**AUX Select.** Auxiliary interface selection input for dual-mode DisplayPort interface

*If the dual-mode DisplayPort interface is unused, these signals can be left unconnected.

Table 3.10: eDisplayPort Signals

Signal *	Description
EDP0_AUX/ EDP0_AUX#	**AUX Data.** Auxiliary data pair for eDisplayPort interface
EDP0_HPD	**Hot Plug Detect.** Hot plug detection input for eDisplayPort interface
EDP0_[0:3]/ EDP0_[0:3]#	**Data Pairs.** Data ports 0-3 for eDisplayPort interface
EDP0_BKLTCTL	**Backlight Control.** Backlight control signal output for eDisplayPort interface
EDP0_BKLTEN	**Backlight Enable.** Backlight enable signal output for eDisplayPort interface
EDP0_VDDEN	**Power.** Power enable signal for eDisplayPort interface

*If the eDisplayPort interface is unused, these signals can be left unconnected.

3.10 LPC

The XPedite8150 provides a low pin count (LPC) interface to support low-bandwidth and legacy input/output devices. The LPC interface uses 3.3 V signaling. Table 3.11 describes the LPC signals.

Table 3.11: LPC Signals

Signal *	Description
LPC_AD[0:3]	**Address / Data.** Address and data ports 0-3, connected to the E3800's LPC bus
LPC_CLK0	**Clock.** LPC Clock 0 signal, connected to the E3800's LPC bus
LPC_FRAME#	**Frame.** LPC frame signal, connected to the E3800's LPC bus
LPC_SERIRQ	**Serial IRQ.** LPC serial interrupt signal, connected to the E3800's LPC bus

*If the LPC interface is unused, these signals can be left unconnected.

3.11 Power Control

The XPedite8150 provides access to various signals related power control, as described in Table 3.12.

Table 3.12: Power Control Signals

Signal *	Description
BATLOW#	**Battery Low.** Indicates insufficient input power and holds the E3800 in suspend state 5
PWRBTN#	**Power Button.** Connected to the E3800
PWR_IN_PGOOD	**Power Good.** Signals the XPedite8150 to begin its power-on sequence. Before assertion of PWR_IN_PGOOD, every on-card supply is turned off.
SLEEP#	**Sleep.** Connected to an E3800 GPIO signal
SLP_S3#	**Sleep Indicator 3.** Asserts when the E3800 enters suspend state 3
SLP_S4#	**Sleep Indicator 4.** Asserts when the E3800 enters suspend state 4

Table 3.12: Power Control Signals *(continued)*

Signal *	Description
SLP_S5#	**Sleep Indicator 5.** Asserts when the E3800 enters suspend state 5. (Not supported on XPedite8150. Signal is pulled up.)
SUS_STAT#	**Suspend Status.** Buffered version of the E3800's SUS_STAT# output
WAKE0#	**Wake 0.** Connected to an E3800 GPIO signal
WAKE1#	**Wake 1.** Connected to an E3800 GPIO signal

*If unused, these signals can be left unconnected.

3.12 Miscellaneous

The XPedite8150's COM Express interface includes a number of miscellaneous signals, as described in Table 3.13. These signals use standard COM Express 3.3 V signaling, except where noted otherwise.

Table 3.13: Miscellaneous Signals

Signal *	Description
FLSH_PASS_CS	**Alternate Flash Select.** When pulled low, forces the processor to boot from alternate flash. See Section 2.5 for more information.
NVM_WP	**Non-Volatile Memory Write Protect.** When allowed to float high, prevents any non-volatile storage on the XPedite8150 from being written. Installing J50020 overrides NVM_WP on the XPedite8150.
PWM_OUT	**PWM Out.** Connected to E3800's PWM0 output through a buffer
RESET_IN#	**Reset In.** Signals the XPedite8150 to reset. Asserting RESET_IN# causes RESET_OUT# to be asserted
RESET_OUT#	**Reset Out.** Asserted when resetting the XPedite8150. Also asserted when the XPedite8150 performs a local reset.
SPKR	**Speaker.** Connected to E3800's SPKR pin through a buffer. Can be used as GPO.

Table 3.13: Miscellaneous Signals *(continued)*

Signal *	Description
THERM#	**Thermal Interrupt.** Connected to E3800 PROCHOT circuitry, can be configured to cause E3800 to throttle performance and reduce power consumption when asserted.
THERMTRIP#	**Reserved.** The thermal trip signal is pulled up.

*If unused, these signals can be left unconnected.

3.13 Connector

The XPedite8150 utilizes a single 220-pin socket connector, J1 (AB), to implement an X-ES-specific, modified, Type 10 COM Express interface. Please refer to Section 2.4 for the location of this connector on the XPedite8150 printed circuit board. See the *PICMG COM Express Specification* for detailed signal descriptions.

Caution

Only install the XPedite8150 on a carrier card that specifically supports the X-ES-enhanced Type 10 pinout. Installing the XPedite8150 on an unsupported carrier can result in electrical damage to the module and/or carrier.

3.13.1 J1 (AB)

Table 3.14: Connector Pinout, J1 (AB)

Pin	COM Express (Type 10) Spec. Assignment	XPedite8150 Assignment*	Direction	Voltage	Note / Reference
A1	GND	GND	-	-	-
A2	GBE0_MDI3-	ETH0_DD#	In/Out	-	Section 3.3
A3	GBE0_MDI3+	ETH0_DD	In/Out	-	Section 3.3
A4	GBE0_LINK100#	ETH0_LINK100#	Out	-	Section 3.3
A5	GBE0_LINK1000#	ETH0_LINK1000#	Out	-	Section 3.3
A6	GBE0_MDI2-	ETH0_DC#	In/Out	-	Section 3.3
A7	GBE0_MDI2+	ETH0_DC	In/Out	-	Section 3.3
A8	GBE0_LINK#	ETH0_LINK#	Out	-	Section 3.3
A9	GBE0_MDI1-	ETH0_DB#	In/Out	-	Section 3.3

Table 3.14: Connector Pinout, J1 (AB) *(continued)*

Pin	COM Express (Type 10) Spec. Assignment	XPedite8150 Assignment*	Direction	Voltage	Note / Reference
A10	GBE0_MDI1+	ETH0_DB	In/Out	-	Section 3.3
A11	GND	GND	-	-	-
A12	GBE0_MDI0-	ETH0_DA#	In/Out	-	Section 3.3
A13	GBE0_MDI0+	ETH0_DA	In/Out	-	Section 3.3
A14	GBE0_CTREF	-	-	-	No connection. The XPedite8150 uses a voltage mode PHY, so no CTRef voltage is required on the Ethernet magnetics module.
A15	SUS_S3#	SLP_S3#	Out	-	Section 3.11
A16	SATA0_TX+	SATA0_TX	Out	-	Section 3.1
A17	SATA0_TX-	SATA0_TX#	Out	-	Section 3.1
A18	SUS_S4#	SLP_S4#	Out	-	Section 3.11
A19	SATA0_RX+	SATA0_RX	In	-	Section 3.1
A20	SATA0_RX-	SATA0_RX#	In	-	Section 3.1
A21	GND	GND	-	-	-
A22	USB_SSRX0-	USBSS_RX0#	In	-	Section 3.4
A23	USB_SSRX0+	USBSS_RX0	Out	-	Section 3.4
A24	SUS_S5#	Pull-up only	Out	-	The E3800 does not provide a SUS_S5 output. Section 3.11
A25	USB_SSRX1-	Reserved	-	-	-
A26	USB_SSRX1+	Reserved	-	-	-
A27	BATLOW#	BATLOW#	In	3.3 V	Section 3.11
A28	(S)ATA_ACT#	SATA_ACT#	Out	3.3 V	Section 3.1
A29	AC/HDA_SYNC	-	-	-	No connection
A30	AC/HDA_RST#	-	-	-	No connection
A31	GND	GND	-	-	-
A32	AC/HDA_BITCLK	-	-	-	No connection
A33	AC/HDA_SDOUT	-	-	-	No connection
A34	BIOS_DIS0#	GPIO	In/Out	3.3 V	Connected to E3800 pin N24, GPIO_S5[23] through bidirectional voltage translation. Section 3.8
A35	THRMTRIP#	Pull-up only	Out	3.3 V	The E3800 does not provide thermal trip output. Section 3.12
A36	USB6-	-	-	-	No connection
A37	USB6+	-	-	-	No connection
A38	USB_6_7_OC#	Pull-up only	In	3.3 V	Section 3.4
A39	USB4-	-	-	-	No connection

Table 3.14: Connector Pinout, J1 (AB) *(continued)*

Pin	COM Express (Type 10) Spec. Assignment	XPedite8150 Assignment*	Direction	Voltage	Note / Reference
A40	USB4+	-	-	-	No connection
A41	GND	GND	-	-	-
A42	USB2-	USB2_D#	In/Out	-	Section 3.4
A43	USB2+	USB2_D	In/Out	-	Section 3.4
A44	USB_2_3_OC#	USB23_OC#	In	3.3 V	Connected to E3800 pin B20, USB_OC1_N/GPIO_S5[20] through input-only voltage translation. Section 3.4
A45	USB0-	USB0_D#	In/Out	-	Section 3.4
A46	USB0+	USB0_D	In/Out	-	Section 3.4
A47	VCC_RTC	VCC_RTC	In	2-3.3 V	-
A48	EXCD0_PERST#	RESET_OUT#	Out	3.3 V	Section 3.12
A49	EXCD0_CPPE#	GPIO	In/Out	3.3 V	Connected to E3800 pin J3, GPIO_S5[40] through bidirectional voltage translation. Section 3.8
A50	LPC_SERIRQ	LPC_SERIRQ	In/Out	3.3 V	Section 3.10
A51	GND	GND	-	-	-
A52	Reserved	ETH1_DA	In/Out	-	Section 3.3
A53	Reserved	ETH1_DA#	In/Out	-	Section 3.3
A54	GPI0	GPIO	In/Out	3.3 V	Connected to E3800 pin K18, GPIO_S5[27] through bidirectional voltage translation. Section 3.8
A55	Reserved	ETH1_DC	In/Out	-	Section 3.3
A56	Reserved	ETH1_DC#	In/Out	-	Section 3.3
A57	GND	GND	-	-	-
A58	PCIE_TX3+	-	-	-	No connection
A59	PCIE_TX3-	-	-	-	No connection
A60	GND	GND	-	-	-
A61	PCIE_TX2+	-	-	-	No connection
A62	PCIE_TX2-	-	-	-	No connection
A63	GPI1	GPIO	In	3.3 V	Connected to E3800 pin K20, GPIO_S5[28] through bidirectional voltage translation. Section 3.8
A64	PCIE_TX1+	PE1_TX	Out	-	Section 3.2
A65	PCIE_TX1-	PE1_TX#	Out	-	Section 3.2
A66	GND	GND	-	-	-

Table 3.14: Connector Pinout, J1 (AB) *(continued)*

Pin	COM Express (Type 10) Spec. Assignment	XPedite8150 Assignment*	Direction	Voltage	Note / Reference
A67	GPI2	GPIO	In/Out	3.3 V	Connected to E3800 pin M22, GPIO_S5[29] through bidirectional voltage translation. Section 3.8
A68	PCIE_TX0+	PE0_TX	Out	-	Section 3.2
A69	PCIE_TX0-	PE0_TX#	Out	-	Section 3.2
A70	GND	GND	-	-	-
A71	LVDS_A0+	EDP0_2	Out	-	Alt. pinout in spec. (embedded DisplayPort instead of LVDS). Section 3.9
A72	LVDS_A0-	EDP0_2#	Out	-	Alt. pinout in spec. (embedded DisplayPort instead of LVDS). Section 3.9
A73	LVDS_A1+	EDP0_1	Out	-	Alt. pinout in spec. (embedded DisplayPort instead of LVDS). Section 3.9
A74	LVDS_A1-	EDP0_1#	Out	-	Alt. pinout in spec. (embedded DisplayPort instead of LVDS). Section 3.9
A75	LVDS_A2+	EDP0_0	Out	-	Alt. pinout in spec. (embedded DisplayPort instead of LVDS). Section 3.9
A76	LVDS_A2-	EDP0_0#	Out	-	Alt. pinout in spec. (embedded DisplayPort instead of LVDS). Section 3.9
A77	LVDS_VDD_EN	EDP0_VDDEN	Out	-	Section 3.9
A78	LVDS_A3+	-	-	-	No connection
A79	LVDS_A3-	-	-	-	No connection
A80	GND	GND	-	-	-
A81	LVDS_A_CK+	EDP0_3	Out	-	Alt. pinout in spec. (embedded DisplayPort instead of LVDS). Section 3.9
A82	LVDS_A_CK-	EDP0_3#	Out	-	Alt. pinout in spec. (embedded DisplayPort instead of LVDS). Section 3.9
A83	LVDS_I2C_CK	EDP0_AUX	In/Out	-	Alt. pinout in spec. (embedded DisplayPort instead of LVDS). Section 3.9
A84	LVDS_I2C_DAT	EDP0_AUX#	In/Out	-	Alt. pinout in spec. (embedded DisplayPort instead of LVDS). Section 3.9
A85	GPI3	GPIO	In/Out	3.3 V	Connected to E3800 pin M24, GPIO_S5[30] through bidirectional voltage translation. Section 3.8
A86	Reserved	-	-	-	No connection
A87	eDP_HPD	EDP0_HPD	In	3.3 V	Section 3.9
A88	PCIE_CLK_REF+	REFCLK	Out	-	Section 3.2
A89	PCIE_CLK_REF-	REFCLK#	Out	-	Section 3.2
A90	GND	GND	-	-	-
A91	SPI_POWER	SPI_PWR	Out	3.3 V	Section 3.6

Table 3.14: Connector Pinout, J1 (AB) *(continued)*

Pin	COM Express (Type 10) Spec. Assignment	XPedite8150 Assignment*	Direction	Voltage	Note / Reference
A92	SPI_MISO	SPI_MISO	In	3.3 V	Section 3.6
A93	GPO0	GPIO	In/Out	3.3 V	Connected to E3800 pin M3, GPIO_S5[32] through bidirectional voltage translation. Section 3.8
A94	SPI_CLK	SPI_CLK	Out	3.3 V	Section 3.6
A95	SPI_MOSI	SPI_MOSI	Out	3.3 V	Section 3.6
A96	TPM_PP	-	-	-	No connection
A97	TYPE10#	TYPE10#	Out	0 V	Connected to ground on Type 10 modules, per the COM Express spec.
A98	SER0_TX	SER0_TX	Out	3.3 V	Section 3.5
A99	SER0_RX	SER0_RX	In	3.3 V	Section 3.5
A100	GND	GND	-	-	-
A101	SER1_TX	SER1_TX	Out	3.3 V	Section 3.5
A102	SER1_RX	SER1_RX	In	3.3 V	Section 3.5
A103	LID#	NVM_WP	In	3.3 V	Non-volatile memory write protect. Section 3.12
A104	+12V	+12V	-	-	-
A105	+12V	+12V	-	-	-
A106	+12V	+12V	-	-	-
A107	+12V	+12V	-	-	-
A108	+12V	+12V	-	-	-
A109	+12V	+12V	-	-	-
A110	GND	GND	-	-	-
B1	GND	GND	-	-	-
B2	GBE0_ACT#	ETH0_ACT#	Out	3.3 V	Section 3.3
B3	LPC_FRAME#	LPC_FRAME#	Out	3.3 V	Section 3.10
B4	LPC_AD0	LPC_AD0	In/Out	3.3 V	Section 3.10
B5	LPC_AD1	LPC_AD1	In/Out	3.3 V	Section 3.10
B6	LPC_AD2	LPC_AD2	In/Out	3.3 V	Section 3.10
B7	LPC_AD3	LPC_AD3	In/Out	3.3 V	Section 3.10
B8	LPC_DRQ0#	-	-	-	No connection
B9	LPC_DRQ1#	-	-	-	No connection
B10	LPC_CLK	LPC_CLK	Out	3.3 V	Section 3.10
B11	GND	GND	-	-	-
B12	PWRBTN#	PWRBTN#	In	3.3 V	Section 3.11

Table 3.14: Connector Pinout, J1 (AB) *(continued)*

Pin	COM Express (Type 10) Spec. Assignment	XPedite8150 Assignment*	Direction	Voltage	Note / Reference
B13	SMB_CK	I2C1_SCL	In/Out	3.3 V	Connected to E3800 pin BG23, SIO_I²C0_CLK/GPIO_S0_SC[79] through bidirectional voltage translation. Section 3.7
B14	SMB_DAT	I2C1_SDA	In/Out	3.3 V	Connected to E3800 pin BH22, SIO_I²C0_DATA/GPIO_S0_SC[78] through bidirectional voltage translation. Section 3.7
B15	SMB_ALERT#	GPIO	In/Out	3.3 V	Connected to E3800 pin M20, GPIO_S5[24] through bidirectional voltage translation. Section 3.7
B16	SATA1_TX+	SATA1_TX	Out	-	Section 3.1
B17	SATA1_TX-	SATA1_TX#	Out	-	Section 3.1
B18	SUS_STAT#	SUS_STAT#	Out	3.3 V	Section 3.11
B19	SATA1_RX+	SATA1_RX	In	-	Section 3.1
B20	SATA1_RX-	SATA1_RX#	In	-	Section 3.1
B21	GND	GND	-	-	-
B22	USB_SSTX0-	USBSS_TX0#	Out	-	Section 3.4
B23	USB_SSTX0+	USBSS_TX0	Out	-	Section 3.4
B24	PWR_OK	PWR_IN_PGOOD	In	3.3 V	Section 3.11
B25	USB_SSTX1-	Reserved	-	-	-
B26	USB_SSTX1+	Reserved	-	-	-
B27	WDT	-	-	-	No connection
B28	AC/HDA_SDIN2	-	-	-	No connection
B29	AC/HDA_SDIN1	-	-	-	No connection
B30	AC/HDA_SDIN0	-	-	-	No connection
B31	GND	GND	-	-	-
B32	SPKR	SPKR	Out	3.3 V	Connected to E3800 pin BH12, SPKR/GPIO_S0_SC[54] through output-only voltage translation. Section 3.12
B33	I2C_CK	I2C2_SCL	In/Out	3.3 V	Connected to E3800 pin BH24, SIO_I²C1_CLK/GPIO_S0_SC[81] through bidirectional voltage translation. Section 3.7
B34	I2C_DAT	I2C2_SDA	In/Out	3.3 V	Connected to E3800 pin BG24, SIO_I²C1_DATA/GPIO_S0_SC[80] through bidirectional voltage translation. Section 3.7
B35	THRM#	THERM#	In	3.3 V	Section 3.12
B36	USB7-	-	-	-	No connection
B37	USB7+	-	-	-	No connection
B38	USB_4_5_OC#	Pull-up only	In	3.3 V	Section 3.4

Table 3.14: Connector Pinout, J1 (AB) *(continued)*

Pin	COM Express (Type 10) Spec. Assignment	XPedite8150 Assignment*	Direction	Voltage	Note / Reference
B39	USB5-	-	-	-	No connection
B40	USB5+	-	-	-	No connection
B41	GND	GND	-	-	-
B42	USB3-	USB3_D#	In/Out	-	Section 3.4
B43	USB3+	USB3_D	In/Out	-	Section 3.4
B44	USB_0_1_OC#	USB01_OC#	In	3.3 V	Connected to E3800 pin C20, USB_OC0_N/GPIO_S5[19] through input-only voltage translation. Section 3.4
B45	USB1-	USB1_D#	In/Out	-	Section 3.4
B46	USB1+	USB1_D	In/Out	-	Section 3.4
B47	EXCD1_PERST#	RESET_OUT#	Out	3.3 V	Section 3.12
B48	EXCD1_CPPE#	GPIO	In/Out	3.3 V	Connected to E3800 pin H3, GPIO_S5[42] through bidirectional voltage translation. Section 3.8
B49	SYS_RESET#	RESET_IN#	In	3.3 V	Section 3.12
B50	CB_RESET#	RESET_OUT#	Out	3.3 V	Section 3.12
B51	GND	GND	-	-	-
B52	Reserved	ETH1_DB	In/Out	-	Section 3.3
B53	Reserved	ETH1_DB#	In/Out	-	Section 3.3
B54	GPO1	GPIO	In/Out	3.3 V	Connected to E3800 pin L1, GPIO_S5[33] through bidirectional voltage translation. Section 3.8
B55	Reserved	ETH1_DD	In/Out	-	Section 3.3
B56	Reserved	ETH1_DD#	In/Out	-	Section 3.3
B57	GPO2	GPIO	In/Out	3.3 V	Connected to E3800 pin K2, GPIO_S5[34] through bidirectional voltage translation. Section 3.8
B58	PCIE_RX3+	-	-	-	No connection
B59	PCIE_RX3-	-	-	-	No connection
B60	GND	GND	-	-	-
B61	PCIE_RX2+	-	-	-	No connection
B62	PCIE_RX2-	-	-	-	No connection
B63	GPO3	GPIO	In/Out	3.3 V	Connected to E3800 pin K3, GPIO_S5[35] through bidirectional voltage translation. Section 3.8
B64	PCIE_RX1+	PE1_RX	In	-	Section 3.2
B65	PCIE_RX1-	PE1_RX#	In	-	Section 3.2

Table 3.14: Connector Pinout, J1 (AB) *(continued)*

Pin	COM Express (Type 10) Spec. Assignment	XPedite8150 Assignment*	Direction	Voltage	Note / Reference
B66	WAKE0#	GPIO	In/Out	3.3 V	Connected to E3800 pin J18, GPIO_S5[25] through bidirectional voltage translation. Section 3.11
B67	WAKE1#	GPIO	In/Out	3.3 V	Connected to E3800 pin M18, GPIO_S5[26] through bidirectional voltage translation. Section 3.11
B68	PCIE_RX0+	PE0_RX	In	-	Section 3.2
B69	PCIE_RX0-	PE0_RX#	In	-	Section 3.2
B70	GND	GND	-	-	-
B71	DDI0_PAIR0+	DDP0_0	Out	-	Section 3.9
B72	DDI0_PAIR0-	DDP0_0#	Out	-	Section 3.9
B73	DDI0_PAIR1+	DDP0_1	Out	-	Section 3.9
B74	DDI0_PAIR1-	DDP0_1#	Out	-	Section 3.9
B75	DDI0_PAIR2+	DDP0_2	Out	-	Section 3.9
B76	DDI0_PAIR2-	DDP0_2#	Out	-	Section 3.9
B77	DDI0_PAIR4+	-	-	-	No connection
B78	DDI0_PAIR4-	-	-	-	No connection
B79	LVDS_BKLT_EN	EDP0_BKLTEN	Out	3.3 V	Section 3.9
B80	GND	-	-	-	No connection
B81	DDI0_PAIR3+	DDP0_3	Out	-	Section 3.9
B82	DDI0_PAIR3-	DDP0_3#	Out	-	Section 3.9
B83	LVDS_BKLT_CTRL	EDP0_BKLTCTRL	Out	3.3 V	Section 3.9
B84	+5V_STANDBY	+5V_STANDBY	-	-	Not required, only used as alt. source for RTC power
B85	+5V_STANDBY	+5V_STANDBY	-	-	Not required, only used as alt. source for RTC power
B86	+5V_STANDBY	+5V_STANDBY	-	-	Not required, only used as alt. source for RTC power
B87	+5V_STANDBY	+5V_STANDBY	-	-	Not required, only used as alt. source for RTC power
B88	BIOS_DIS1#	-	-	-	No connection
B89	DD0_HPD	DDP0_HPD	In	3.3 V	Section 3.9
B90	GND	GND	-	-	-
B91	DDI0_PAIR5+	-	-	-	No connection
B92	DDI0_PAIR5-	-	-	-	No connection
B93	DDI0_PAIR6+	-	-	-	No connection
B94	DDI0_PAIR6-	-	-	-	No connection

Table 3.14: Connector Pinout, J1 (AB) *(continued)*

Pin	COM Express (Type 10) Spec. Assignment	XPedite8150 Assignment*	Direction	Voltage	Note / Reference
B95	DDI0_DDC_AUX_SEL	DDP0_AUX_SEL	In	3.3 V	Section 3.9
B96	USB_HOST_PRSNT	FLSH_PASS_CS	In	3.3 V	Section 3.12
B97	SPI_CS#	SPI_CS#	Out	3.3 V	Section 3.6
B98	DDI0_CTRLCLK_AUX+	DDP0_AUX	In/Out	-	Section 3.9
B99	DDI0_CTRLDATA_AUX-	DDP0_AUX#	In/Out	-	Section 3.9
B100	GND	GND	-	-	-
B101	FAN_PWMOUT	PWM_OUT	Out	3.3 V	Connected to E3800 pin AU322, SIO_PWM0/GPIO_S0_SC[94] through output-only voltage translation. Section 3.12
B102	FAN_TACHIN	-	Out	-	No connection
B103	SLEEP#	GPIO	In/Out	3.3 V	Connected to E3800 pin N3, GPIO_S5[37] through bidirectional voltage translation. Section 3.11
B104	+12V	+12V	-	-	-
B105	+12V	+12V	-	-	-
B106	+12V	+12V	-	-	-
B107	+12V	+12V	-	-	-
B108	+12V	+12V	-	-	-
B109	+12V	+12V	-	-	-
B110	GND	GND	-	-	-

*Shaded items indicate an X-ES enhancement to the COM Express (Type 10) pinout.

Accessory Connectors

This chapter describes internal connectors on the XPedite8150 that provide an interface for accessories from X-ES.

4.1 Overview

Table 4.1 gives a brief description of the XPedite8150's accessory connectors.

Table 4.1: Accessory Connector Descriptions

Name	Reference Designator	Description
XTend209 Debug Connector	J60000	XTend209 Debug Module/Cable for Intel SBC, 2x RS-232, JTAG and Dediprog headers, and Port 80 debug output.

4.2 Pinouts

All signal directions are in reference to the XPedite8150. Out implies the XPedite8150 drives the signal; In implies the XTend209 drives the signal.

Table 4.2: XTend209 Debug Connector Pinout, J60000

Pin	Direction	Signal	Pin	Direction	Signal
1	-	GND	2	-	-
3	-	GND	4	-	-
5	In	PRG_SPI_CS_EN#	6	-	-
7	In	PRG_SPI_CS1_3P3#	8	-	-
9	In	PRG_SPI_CS0_3P3#	10	In/Out*	C0_FLSH_PASS_CS
11	In	PRG_SPI_CLK	12	-	-

Table 4.2: XTend209 Debug Connector Pinout, J60000 *(continued)*

Pin	Direction	Signal	Pin	Direction	Signal
13	Out	PRG_SPI_MOSI	14	-	-
15	In	C0_SPI_MISO_3P3	16	In/Out	C0_LPC_AD3
17	In	EM_PRESENT#	18	In/Out	C0_LPC_AD2
19	Out	C0_SPI_CS1#	20	In/Out	C0_LPC_AD1
21	Out	C0_SPI_CS0#	22	In/Out	C0_LPC_AD0
23	Out	C0_SPI_CLK	24	In	C0_PLT_RST#
25	Out	C0_SPI_MOSI	26	Out	C0_LPC_FRAME#
27	-	-	28	Out	C0_LPC_CLK1
29	-	-	30	-	-
31	-	-	32	-	-
33	-	GND	34	-	-
35	-	GND	36	-	-
37	-	XTEND209_PWR	38	-	-
39	-	XTEND209_PWR			

*Open Drain

Processor

This chapter describes details associated with the XPedite8150's Intel Atom processor.

5.1 Introduction

The Atom processor provides the following features for the XPedite8150.

- Up to four processor cores, four threads
- Up to 2 MB L2 Cache
- Integrated DDR3L-1066/1333 SDRAM Memory Controller with ECC support
- Integrated Intel HD Graphics Controller
- Integrated Graphics Interface with DVI/HDMI/DisplayPort and Integrated DisplayPort/HDMI Audio
- Integrated PCI Express Gen2 (1x4; 2x2; 1x2 and 2x1; or 4x1)
- Digital Thermal Sensor
- Intel Thermal Monitor
- Intel Virtualization Technology (VT-x)
- Intel Trusted Execution Engine (TXE)
- AES New Instructions
- Power Management Support

5.2 DDR3 SDRAM Controller

The Intel Atom processor incorporates a DDR3 SDRAM controller that supports ECC and operates at up to 1333 MT/s. Table 5.1 gives the chip select usage for the XPedite8150.

Table 5.1: Chip Select Assignments

Chip Select	Device Assignment
Channel A : 0	Soldered Rank 0
Channel A : 1	Unused
Channel B : 0	Unused
Channel B : 1	Unused

The XPedite8150 maintains memory integrity by utilizing the Intel Atom memory controller's hardware managed Single Error Correction - Double Error Detection (SEC-DED) Error Correction Code (ECC) capabilities. This allows the XPedite8150 to correct all single-bit errors and to detect all double-bit errors.

5.3 PCI Express Controller

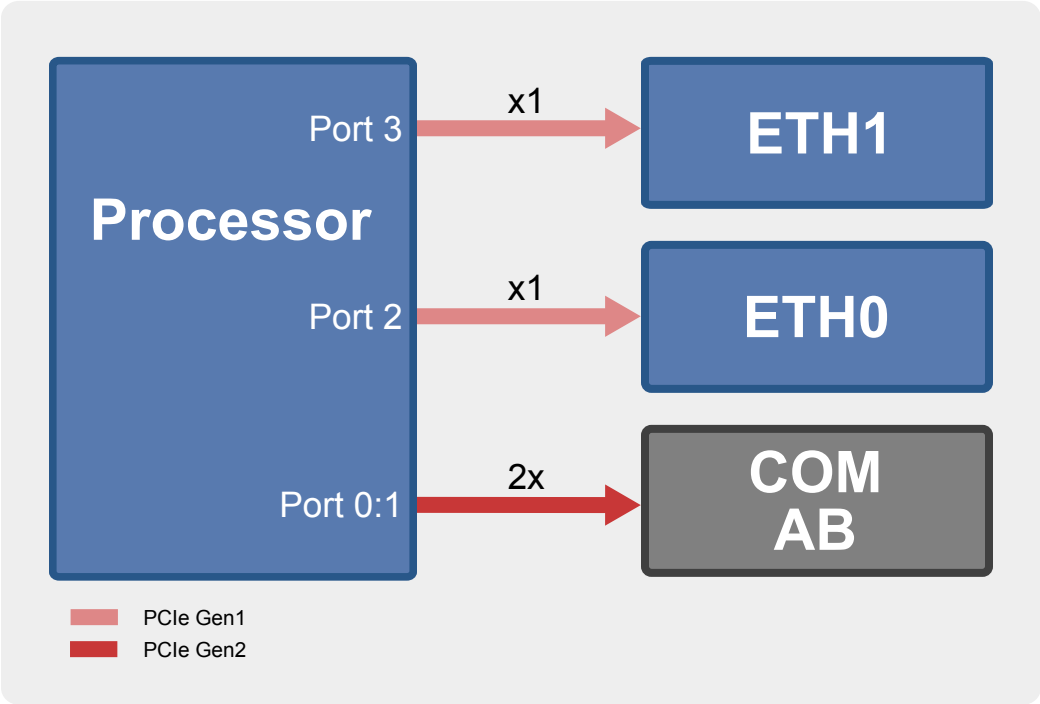
The Intel Atom processor incorporates four lanes of PCI Express 2.0 (5.0 GT/s) which can be bifurcated as a single x4, two x2, a single x2 and two x1, or four x1 ports. When connected to an endpoint device that only supports PCI Express 1.x (2.5 GT/s), the PCI Express controller will automatically train down to those speeds. Each port operates in root complex mode and cannot operate as an endpoint. The ports are assigned as described in Table 5.2.

Table 5.2: Atom PCI Express Port Assignment

PCI Express Port	Device Assignment
Port 0, Lane 0	COM port
Port 1, Lane 1	COM port
Port 2, Lane 2	Intel I210 Ethernet MAC/PHY
Port 3, Lane 3	Intel I210 Ethernet MAC/PHY

Figure 5.1 provides an overview of the XPedite8150's PCI Express connectivity.

Figure 5.1: PCI Express Connectivity



5.4 Graphics Controller

The Intel Atom processor incorporates a high performance graphics controller. See Table 5.3 for specific port assignments.

Table 5.3: Graphics Port Assignments

Graphics Port	Mode	Assignment*
0	Dual-mode DisplayPort	COM AB, DDI0
1	Embedded DisplayPort	COM AB, eDP

*See Section 3.9 for pinout information.

5.5 GPIO

The XPedite8150 incorporates General Purpose I/O (GPIO) pins that connect directly to the E3800 processor. Table 5.4 describes the assignments for these pins.

Table 5.4: Processor GPIO Pin Assignment

Pin	Name	Signal	Assignment
B18	GPIO_S5_0	Flash Select Swap Sense	Reading a zero indicates the alternate NOR flash is selected as the boot device. Reading a one indicates the primary flash is selected.
J24	GPIO_S5_17	Non-Volatile Memory Write Protect Indicator	Reading a zero indicates that the XPedite8150's non-volatile memory is writable. Reading a one indicates that XPedite8150's non-volatile memory is write protected.
F18	GPIO_S5_13	PCIe Mux Control	Controls the PCIe multiplexer that selects between the PCIe-to-PCI bridge for the PMC interface and the auxiliary PCIe link on XMC P16. A low value selects the P16 PCIe port. A high value selects the PCIe to PCI Bridge.
C19	GPIO_S5_10	Reset Output Mask	This signal can be used to mask the XMC Reset Output signal. A low value will allow the reset output to be asserted.
K20	GPIO_S5_28	Serial Port 0 Transmit Enable	In RS-422/485 mode, writing a one enables the transmitter. Writing a zero disables the transmitter. In RS-232 mode, the RS-422/485 Transmitter is disabled.
K18	GPIO_S5_27	Serial Port 0 Mode	Writing a one enables RS-422/485 mode. Writing a zero enables RS-232 mode. By default RS-232 mode is enabled.
BH18	GPIO_S0_SC14	Serial Port 0 Termination Enable	Writing a one enables a 120-ohm terminating resistor between the A and B lines of the RS-422/485 Receiver. This signal must be low in RS-232 mode.
M24	GPIO_S5_30	Serial Port 1 Transmit Enable	In RS-422/485 mode, writing a one enables the transmitter. Writing a zero disables the transmitter. In RS-232 mode, the RS-422/485 Transmitter is disabled.

Table 5.4: Processor GPIO Pin Assignment *(continued)*

Pin	Name	Signal	Assignment
M22	GPIO_S5_29	Serial Port 1 Mode	Writing a one enables RS-422/485 mode. Writing a zero enables RS-232 mode. By default RS-232 mode is enabled.
BF28	GPIO_S0_SC62	Serial Port 1 Termination Enable	Writing a one enables a 120-ohm terminating resistor between the A and B lines of the RS-422/485 Receiver. This signal must be low in RS-232 mode.

X-ES coreboot

This chapter provides details about the X-ES coreboot firmware employed by the XPedite8150.

6.1 Introduction

The XPedite8150 utilizes X-ES coreboot firmware with the Intel Firmware Support Package (FSP). coreboot is a corporate- and community-developed open-source firmware alternative to UEFI and legacy BIOS boot firmware. The X-ES coreboot distribution is customized, verified, and supported by X-ES to provide a small, simple, consistent, open code base with a Linux pre-boot debug environment that is customized for the XPedite8150. It retains support for disk and peripheral devices, as well as support for network-based booting of off-the-shelf x86 operating systems.

coreboot performs minimal hardware initialization required for booting, then passes control to a "payload" to finish the boot procedure. Various payloads can be built into the boot firmware flash image to perform various tasks, such as boot a specific operating system, run a specific test, or load services expected from a typical BIOS. The X-ES coreboot distribution includes SeaBIOS, an open-source, community-developed, coreboot payload customized for the XPedite8150. SeaBIOS provides legacy BIOS compatibility services and loads the user's operating system from standard storage devices. X-ES coreboot also includes built-in iPXE, an open-source network boot firmware with advanced, standards-based, network boot options.

Power-on initialization for modern processors is a complex task that requires detailed information about the processor and its register sets. Intel's Firmware Support Package (FSP) enables open-source projects such as coreboot to support cutting-edge Intel processors. X-ES coreboot includes an Intel FSP that has been customized and optimized for the XPedite8150. Further customization can be performed using the Intel Binary Configuration Tool.

X-ES coreboot includes source code, the Intel FSP binary customized for the XPedite8150, and a 16 MB coreboot ROM binary that is configured to balance speed and compatibility. The ROM binary can be programmed to the XPedite8150's SPI NOR flash.

6.1.1 Distribution Package

The X-ES coreboot distribution package includes and allows for the building of a 16 MB coreboot ROM Image file to be programmed to SPI NOR flash on the XPedite8150.

Table 6.1: X-ES coreboot Distribution Package Contents

File	Description
XPedite8150-FIRMWARE_COREBOOT_C-V.V.V.V.rom	Binary 16 MB X-ES coreboot ROM Image
XPedite8150-FIRMWARE_COREBOOT_DEBUG_C-V.V.V.V.rom	Binary 16 MB X-ES coreboot Debug/Verbose ROM Image
install_intel.sh	Intel FSP, Intel Video ROM, and Intel CPU microcode binaries extraction shell script
xes	Folder containing the X-ES SPI Linux CLI and VxWorks bootrom binaries
intel	Folder containing Intel Binaries and scripts to extract Intel Binaries after license agreement acceptance.
coreboot/src	Folder containing X-ES coreboot source code.
coreboot/util	Folder containing tools used in the X-ES coreboot build process.
coreboot/configs	X-ES coreboot board specific configuration files.
coreboot/payloads/seabios	X-ES coreboot distribution SeaBIOS.
coreboot/payloads/ipxe	X-ES coreboot distribution iPXE network boot loader.

The X-ES coreboot 16 MB ROM Image contains not only coreboot, but also a number of supporting firmware components.

X-ES coreboot 16 MB ROM Image example contents:

- **coreboot.** Open-source bootloader
- **SeaBIOS.** Open-source coreboot payload, provides Legacy BIOS boot functionality
- **Intel FSP.** X-ES-customized Intel binary that performs core chipset initialization. This can be modified with Intel's Binary Configuration Tool (BCT).
- **Intel VBIOS.** Binary from Intel that provides graphical output during SeaBIOS. This can be modified with Intel's Binary Manipulation Program (BMP).
- **Intel Microcode.** Binary from Intel that provides processor microcode
- **SPI Linux CLI.** SPI Linux Command Line Interface for troubleshooting

- **VxWorks bootrom.** Integrated X-ES VxWorks bootrom for fast coreboot to VxWorks booting, if available
- **Intel TXE.** Intel-provided Trusted Execution Engine (TXE) binary for chipset management and configuration functions. This can be modified with Intel's Flash Image Tool (FITC).
- **Intel Flash Descriptor.** Small section at the lowest address in the ROM that describes its contents and sets up soft-straps and runtime flash security.

X-ES coreboot releases include a pre-compiled `.rom` file, which is ready to flash to the target board. However, coreboot must be built (see Section 6.4) in order to re-configure features such as:

- Boot device ordering
- Boot-time optimization (removing unnecessary features)
- VxWorks bootrom

6.1.2 Software Licenses

Applicable software licenses, organized by each top-level directory of the coreboot release kit:

`coreboot/*` **coreboot and its payloads.** coreboot and payloads, such as SeaBIOS, are licensed under GPLv2, unless the individual source file's header indicates otherwise.

`intel/*` **Intel materials: binaries, microcode.** Intel materials are re-distributed in self-extracting `*.se` installers to preserve applicable software licenses. These materials include binaries, such as the Firmware Support Package (FSP), as well as microcode. The license for each Intel software component is displayed when running the `install_intel.sh` script, as detailed in Section 6.4.

`xes/*` **X-ES materials: Linux, VxWorks, and other images.** Includes X-ES provided OS images, such as the SPI Linux CLI or, if available, a VxWorks bootrom. SPI Linux is licensed under GPLv2. The VxWorks bootrom is licensed under a proprietary license from WindRiver.

6.2 X-ES coreboot Tutorials

This section provides basic tutorials for using the XPedite8150's X-ES coreboot with FSP distribution.

6.2.1 Boot Devices

X-ES coreboot includes SeaBIOS as a payload for coreboot. SeaBIOS provides legacy runtime interrupt services for operating systems and allows booting off-the-shelf x86 operating systems, such as Microsoft Windows and RedHat Linux from onboard and off-board media.

6.2.1.1 X-ES coreboot Default Payload

Depending on the selected build-time configuration of X-ES coreboot, either SeaBIOS or VxWorks will be loaded as an X-ES coreboot payload, if no key is pressed.

6.2.1.2 Booting to SeaBIOS

SeaBIOS's primary purpose is to provide legacy BIOS functionality, such as:

- Initializing video
- Master Boot Record Legacy booting from disk devices
- PXE network booting from Ethernet devices

X-ES coreboot will print out the following message early in the boot process:

```
Press or hold lowercase 's' to boot SeaBIOS.
```

Hold the lowercase **s** key to ensure the XPedite8150 boots to SeaBIOS.

Once SeaBIOS boots, continue holding the lowercase **s** key through the following message, in order ensure the XPedite8150 boots to the SeaBIOS boot device selection menu:

```
Press or hold lowercase 's' for SeaBIOS boot menu.
```

The SeaBIOS menu shows a numerical list of available boot devices in their boot order.

Example boot menu:

```
Select boot device:

1. AHCI/0: 8GB NANDrive ATA-8 Hard-Disk (7641 MiBytes)
2. USB MSC Drive JetFlash Transcend 4GB 1100
3. iPXE (PCI 00:00.0)
4. Payload [SPI Linux Command Shell]
5. Payload [bootrom]
```

In this example, press 1 or do not press the **s** key on subsequent boots to boot from the onboard SATA drive, press 2 to boot from an off-board USB flash drive, or press 3 to boot to iPXE, which allows network booting using the PXE protocol.

The boot order can be configured by modifying `src/mainboard/xes/xpedite8150/bootorder` at compile time. If the lowercase **s** is not pressed, SeaBIOS boots from first present and bootable device in the boot order.

SeaBIOS will show devices, such as:

- Onboard SATA-based MBR legacy boot devices, where present (e.g., NANDrive ATA-8 Hard-Disk).
- USB or SATA based MBR Legacy boot devices
- Integrated iPXE (http://www.ipxe.org/) PXE Network Boot Loader
- coreboot.rom embedded payloads, such as:
  - VxWorks bootrom
  - SPI Linux CLI

X-ES coreboot and SeaBIOS must be compiled with support for a given device class in order for SeaBIOS to recognize it. Standard releases generally aim to enable as many boot device types as possible. However, when optimizing boot time, certain boot device types are commonly disabled.

6.3 SPI Linux Tutorials

X-ES coreboot includes a built-in SPI Linux command line interface (CLI) maintenance and diagnostics environment that is accessible from the serial console.

6.3.1 Booting to the X-ES SPI Linux CLI

coreboot prints out the following message early in the boot process:

```
Press or hold lowercase 'l' to boot SPI Linux.
```

Once your key press is detected, it is acknowledged with:

```
Lowercase 'l' found, selecting SPI Linux.
```

The SPI Linux kernel will boot and load an initramfs filesystem. Once the login prompt is reached, you can login as root:

```
login: root
```

No password is needed when logging in as root.

To view the available commands, press the **Tab** key twice, consecutively, while at the # prompt. X-ES value-add commands for interacting with the board have an "xes" prefix, for example **xes-flashrom**, **xes-gpio**, and **xes-gpio**.

Two key coreboot maintenance-related features of X-ES SPI Linux CLI include:

- Firmware Update

  xes-flashrom is used to update coreboot. See Section 6.5.2.2 for information on how to use **xes-flashrom** to update coreboot.

- Board Information Programming

  The **xescfg** utility is used by X-ES manufacturing to program the serial number, as well as other board-specific information. To view usage information, run **xescfg** with no arguments. To view the data programmed, run:

```
# xescfg dump
```

6.4 Building coreboot

6.4.1 Build Environment Setup

Ubuntu Linux 12.04 is currently the only supported build environment. It can be installed on a dedicated computer or as a Virtual Machine (VM).

6.4.1.1 VirtualBox Setup (Optional)

X-ES uses VirtualBox internally to host X-ES coreboot development environments. Using a VM allows for installation of a fresh operating system without a dedicated computer. VirtualBox also includes useful features, such as Settings -> Shared Folders, to allow for easy file transfers between the host machine and guest VM.

- Latest VirtualBox download (host machine) - https://www.virtualbox.org/wiki/Downloads
- Ubuntu VirtualBox documentation - https://help.ubuntu.com/community/VirtualBox
- Ubuntu as a guest OS - https://help.ubuntu.com/community/Ubuntu_as_Guest_OS

To set up VirtualBox Guest Additions 4.2 (see https://www.virtualbox.org/manual/ch04.html for benefits):

- Download command: **wget http://download.virtualbox.org/virtualbox/4.2.0/virtualbox-4.2_4.2.0-80737~Ubuntu~precise_i386.deb**

- Install command: **dpkg --install virtualbox-4.2_4.2.0-80737~Ubuntu~precise_i386.deb**

6.4.1.2 Target Machine Setup

For production builds of coreboot images, the following build environment (used by X-ES for internal testing) is highly recommended. For development builds of coreboot, x86_64 Linux environments and distributions other than Ubuntu may be used.

- Ubuntu server 12.04.1 i386 stack, kernel 3.2.0-23-generic - http://old-releases.ubuntu.com/releases/12.04.1/ubuntu-12.04-server-i386.iso

Run the following command to install all necessary packages on your target machine:

- **apt-get install python python-pycurl dmidecode nfs-common ldap-utils smbfs dnsutils bash-completion emacs vim nano screen dos2unix subversion git man build-essential gcc-multilib gcc libxtst6 texi2html chrpath diffstat texinfo gawk tcl-dev pciutils-dev xz-utils expect telnet curl bc zip x11-common x11-utils libncurses5-dev openjdk-7-jre automake autoconf libtool binutils bison flex iasl librpc-xml-perl perl libsvn-perl libsoap-lite-perl libdbi-perl libdate-calc-perl libarchive-any-perl libarchive-zip-perl libdate-manip-perl libnet-ldapapi-perl libnet-ldap-server-perl libterm-readkey-perl libdbd-mysql-perl libclass-dbi-lite-perl libpdf-api2-perl liblog-dispatch-perl liblogfile-rotate-perl libexporter-easy-perl librpc-xml-perl**

6.4.1.3 Sage EDK

Sage EDK is the baseline compile toolchain. The free compile tools are all that is required to build X-ES coreboot releases. Optionally, the full Sage EDK license can be purchased from Sage Electronic Engineering. This provides access to a GUI-based development environment.

To install the Sage EDK Development tools:

- Acquire the Sage EDK for Linux version 3.00.00_33 installer.
- Install the Sage EDK with this command: **./SageEDK_Linux-x86-Install_3.00.00_33 --mode silent --prefix /opt/sage_edk/**
- Create this file: `/etc/profile.d/sage_edk.sh`
  With these contents: **`export SAGE_HOME=/opt/sage_edk`**
- Set a Symlink from `/bin/sh` to `/bin/bash` with this command: **ln -s /bin/sh /bin/bash**

6.4.2 Building coreboot

To build a 16 MB coreboot.rom image:

Procedure 6.1. Building coreboot

1. Download the X-ES coreboot distribution kit (see Section 6.5.1).
2. Extract the X-ES coreboot distribution kit tar.gz file as follows:

```
tar xzvf XPedite8150-FIRMWARE_COREBOOT_KIT-C-V.V.V.V.tar.gz
```

3. Change to the extracted directory:

```
cd XPedite8150-FIRMWARE_COREBOOT-C-V.V.V.V
```

4. Run the `install_intel.sh` script to extract the Intel Binary and processor microcode materials:

```
./install_intel.sh
```

   - Use the arrow keys to scroll through the license text for each of the materials.
   - Accept all license agreements for the extraction to take place:

   a. Press **q**.

   b. At the prompt, type **y** to accept.

   c. Press **enter**.

5. Change to the coreboot directory:

```
cd coreboot
```

6. Configure X-ES coreboot to build for your target board:

```
./configure.sh xpedite8150
```

7. Optionally, configure X-ES coreboot build options using the build configuration tool:

```
make menuconfig
```

   See Section 6.7 for available build options.

8. Optionally, edit the bootorder file by modifying `src/mainboard/xes/xpedite8150/bootorder` with a text editor.
9. Optionally, configure SeaBIOS options using the configuration tool:

```
cd payloads/seabios/
make menuconfig
```

After configuring SeaBIOS options using the build configuration tool, switch back to the `coreboot` directory:

```
cd ../..
```

10. Type **`make`** to build the X-ES coreboot 16 MB ROM file.
11. Flash the 16 MB ROM file located at `coreboot/build/coreboot.rom` according to Section 6.5.

6.5 Flash Update

The XPedite8150 is shipped with programmed X-ES coreboot with Intel FSP firmware. This section describes how to update the XPedite8150's firmware in the field.

6.5.1 X-ES coreboot Kit

The primary method for delivering the X-ES coreboot distribution is via the SupportNet website at (https://support.xes-inc.com/). Customers are encouraged to create a user account on SupportNet because it provides access to important updates. SupportNet also allows you to set alerts for selected products, and it conveniently sends a notification e-mail when an update occurs.

Refer to the following documents for each X-ES coreboot distribution release on SupportNet:

- **release_notes.txt** contains important release information, including any dependencies and known issues and limitations.
- **changes.txt** contains a log of changes between releases with the same major version.

Section 6.1.1 describes the X-ES coreboot distribution contents. After familiarizing yourself with the distribution contents, proceed to Section 6.5.2 to determine your preferred method of updating.

6.5.2 Update Procedures

6.5.2.1 Flashing coreboot with Dediprog SF100 Programmer

Table 6.2 describes programming accessories for flashing an X-ES coreboot 16 MB ROM image:

Table 6.2: Flash Update Programming Accessories

Part Number	Description
90071685-2	**XTend209 Debug Module.** The XTend209 is a firmware programming/debug module for use with the XPedite8150.
90081064	**Dediprog SF100 Programmer.** The DediProg SF100 is an in-circuit SPI NOR Flash programmer.
90081065	**Dediprog SF100-to-XTend209 Adapter Cable.** The Dediprog SF100-to-XTend209 Adapter Cable allows the DediProg SF100 to interface with the XTend209 to program the SPI NOR Flash on the XPedite8150.

1. Connect these accessories as follows:

   XPedite8150 -> XTend209 -> Adapter Cable -> Programmer

2. Install the DediProg Engineering GUI software from the DediProg website on a Windows PC: http://www.dediprog.com/pd/spi-flash-solution/sf100.

3. The XPedite8150's flash device runs at 1.8 V. However, the flash programming interface is converted to 3.3 V. Therefore, it is important to not allow the DediProg to auto-detect the voltage.

   To stop the DediProg from incorrectly auto-detecting the flash chip voltage:

   a. In the DediProg Engineering main user interface, click the **`Config`** button.

   b. Click the **`Miscellaneous Settings`** button in the windows that appears.

   c. Check the **`Manual select Vcc`** checkbox.

   d. Select **`Using fixed Vcc`** and then select **`3.5V`**.

4. Apply power to the XPedite8150 setup and click the **`Detect`** button on the DediProg Engineering GUI. Select **`MX25U12835F`** as the SPI NOR flash device.

5. Click the **`File`** button to select the `XPedite8150-FIRMWARE_COREBOOT-<Version>.rom` or `coreboot.rom` programming file.

6. Click **`OK`** and then start programming by clicking the **`Batch`** button.

6.5.2.2 Flashing coreboot from Linux

It is recommended to maintain the same coreboot version on both Primary and Recovery SPI NOR flash devices to ensure recoverability in the event of firmware corruption.

Note

Firmware updates are always applied to the inactive SPI flash to avoid interference with the active coreboot image or the Trusted Execution Engine (TXE). Therefore, if you have booted from the Primary SPI NOR Flash, you must update the Recovery SPI NOR flash, and vice versa.

1. Boot to Linux with X-ES **flashrom** support, logging on as root.
   - X-ES SPI Linux CLI has integrated X-ES **flashrom** support. See Section 6.3 to boot to X-ES SPI Linux.
   - Recent versions of the following Linux packages may have X-ES **flashrom** support added:
     - X-ES Enterprise Linux (XEL)
     - X-ES Gentoo Linux
2. Copy the `XPedite8150-FIRMWARE_COREBOOT-<Version>.rom` from the X-ES coreboot distribution or your built `coreboot.rom` image to a Linux-accessible location in one of the following ways:
   - Network-based transfer
   - Mount the appropriate network drive (for example, NFS)
   - Download the file to the local filesystem (for example, tftp, ftp, scp, wget)
   - Physical disk-based transfer
   - Mount USB or SATA-based disk device
3. Program the inactive SPI NOR flash with **flashrom** The inactive SPI NOR flash is mapped to the Intel SPI controller's Platform Data region. To write the `.rom` file, run the command: **xes-flashrom -w XPedite8150-FIRMWARE_coreboot-<Version>.rom**
4. Cleanly power-down the system.

   Run the **poweroff** command.

   Power-off the system, once Linux has completed its shutdown.
5. Change the state of Boot Device Select Jumper (see Section 2.5).

6. Boot from the newly flashed image and verify the version, which is always printed on a separate line early in the boot process. For example:

```
X-ES XPedite8150 coreboot Firmware Version C-V.V.V.V
```

7. Repeat steps 1 through 6 in order to program the SPI NOR flash originally booted from in step 1.

6.6 Memory Maps

This section provides memory maps to convey where resources are mapped in memory and how they are utilized by X-ES coreboot and operating systems.

6.6.1 I²C SEEPROM

The I²C SEEPROM memory map shows locations for information storage by end-user applications, BSPs, and critical hardware configuration data used by the UEFI BIOS. The I²C SEEPROM is also termed, System EEPROM, to reflect its central role in onboard non-volatile information storage.

Table 6.3: I²C SEEPROM

Address Range	Size	Description
0x0000 - 0x37FF	14 kB	BSP non-volatile storage / End-user data storage
0x3800 - 0x382F	48 bytes	Reserved
0x3830 - 0x3BFF	976 bytes	Non-volatile configuration data[1]
0x3C00 - 0x3CFF	256 bytes	DIMM A SPD data[1]
0x3D00 - 0x3EFF	512 bytes	Reserved
0x3F00 - 0x3FFF	256 bytes	Reserved for OS-level BIT tests

[1] The BSP or user application should never modify this data section, except within the context of performing a de-classification / re-classification operation.

Each of these I²C SEEPROM partitions are used as follows:

BSP non-volatile storage

- Some board support packages need a non-volatile storage region for parameters, such as network settings. BSPs generally will store these parameters starting at offset 0x0 and leave any unused space available to the end user. Use of this area is operating system-specific, so please see the BSP user manual for details.

End-user data storage

- The region between the BSP non-volatile storage and Reserved region is available for data storage by user applications.

Non-volatile configuration data

- X-ES coreboot uses this region to access non-volatile, board-specific information, such as the board's serial number and PCB revision. The **xescfg** Linux command described in Section 6.3 is used to access data within this region.

DIMM A SPD data

- X-ES programs the SPD data regions with information about the sizes, organization, and timings of onboard memory components. The data structures for this information follow the appropriate JEDEC industry standards.

Reserved for OS-level BIT tests

- This region is used by X-ES OS-level BIT Tests (e.g., X-ES CBIT and X-ES IBIT) to verify the SEEPROM and store BIT test results.

6.7 Build Options

The X-ES coreboot distribution has various build options that are accessed through the build configuration tool.

Note

Many of these options are generic to coreboot, and not all apply to the XPedite8150.

6.7.1 General Setup Menu

6.7.1.1 EXPERT

Prompt: Expert mode

Allows you to select certain advanced configuration options.

> **Warning**
>
> **Only enable this option if you really know what you are doing! You have been warned!**

6.7.1.2 LOCALVERSION

Prompt: Local version string

Append an extra string to the end of the coreboot version.

This can be useful if, for instance, you want to append the respective board's hostname or some other identifying string to the coreboot version number, so that you can easily distinguish boot logs of different boards from each other.

6.7.1.3 CBFS_PREFIX

Prompt: CBFS prefix to use

Select the prefix to all files put into the image. The default is "fallback", and "normal" is a common alternative.

6.7.1.4 ALT_CBFS_LOAD_PAYLOAD

Prompt: Use alternative **cbfs_load_payload()** implementation.

Either board or southbridge provide an alternative **cbfs_load_payload()** implementation. This may be used, for example, if accessing the ROM through memory-mapped I/O is slow and a faster alternative can be provided. This option must be enabled for the "Press 'key' to boot payload" feature.

6.7.1.5 COMPILER_GCC

Prompt: GCC

Use the GNU Compiler Collection (GCC) to build coreboot.

For details, see http://gcc.gnu.org.

6.7.1.6 COMPILER_LLVM_CLANG

Prompt: LLVM/clang

Use LLVM/clang to build coreboot.

For details, see http://clang.llvm.org.

6.7.1.7 SCANBUILD_ENABLE

Prompt: Build with scan-build for static code analysis.

Changes the build process to use scan-build (a utility for running the clang static code analyzer from the command line).

Requires the scan-build utility in your system **$PATH**.

For details, see http://clang-analyzer.llvm.org/scan-build.html.

6.7.1.8 SCANBUILD_REPORT_LOCATION

Prompt: Directory for the scan-build report(s)

Directory where the scan-build reports should be stored. The reports are stored in subdirectories named as `yyyy-mm-dd-*` in the specified directory.

If this setting is left empty, the coreboot top-level directory will be used to store the report subdirectories.

6.7.1.9 CCACHE

Prompt: Use ccache to speed up (re)compilation.

Enables the use of **ccache** for faster builds.

Requires the **ccache** utility in your system **$PATH**.

For details, see https://ccache.samba.org.

6.7.1.10 SCONFIG_GENPARSER

Prompt: Generate SCONFIG parser using flex and bison

Enable this option if you are working on the **sconfig** device tree parser and made changes to `sconfig.l` and `sconfig.y`.

Otherwise, type **N**.

6.7.1.11 USE_OPTION_TABLE

Prompt: Use CMOS for configuration values.

Enable this option if coreboot shall read options from the "CMOS" NVRAM instead of using hard-coded values.

6.7.1.12 COMPRESS_RAMSTAGE

Prompt: Compress ramstage with LZMA.

Compress **ramstage** to save memory in the flash image.

Note

Decompression might slow down booting, if the boot flash is connected through a slow link (e.g., SPI).

6.7.1.13 INCLUDE_CONFIG_FILE

Prompt: Include the coreboot `.config` file in the ROM image.

Include the `.config` file that was used to compile coreboot in the (CBFS) ROM image. This is useful if you want to know which options were used to build a specific `coreboot.rom` image.

Typing **Y** here will increase the image size by 2-3 kB.

You can use the following command to easily list the options: **grep -a CONFIG_ coreboot.rom**

Alternatively, you can use **cbfstool** to print the image contents (including the raw **config** item we are looking for).

Example:

```
$ cbfstool coreboot.rom print
coreboot.rom: 4096 kB, bootblocksize 1008, romsize 4194304,
```

```
offset 0x0
Alignment: 64 bytes
Name Offset Type Size
cmos_layout.bin 0x0 cmos layout 1159
fallback/romstage 0x4c0 stage 339756
fallback/coreboot_ram 0x53440 stage 186664
fallback/payload 0x80dc0 payload 51526
config 0x8d740 raw 3324
(empty) 0x8e480 null 3610440
```

6.7.1.14 DYNAMIC_CBMEM

Prompt: The CBMEM space is dynamically grown.

Instead of reserving a static amount of CBMEM space, the CBMEM area grows dynamically. CBMEM can be used both in romstage (after memory initialization) and ramstage.

6.7.1.15 COLLECT_TIMESTAMPS

Prompt: Create a table of timestamps collected during boot.

Make coreboot create a table of timer-ID/timer-value pairs to allow measuring time spent at different phases of the boot. process.

6.7.1.16 TIMESTAMP_PRINT_LEVEL

Prompt: Configure timestamp log level 1-8.

Set the loglevel that the timestamp values will be displayed with. If this is set above the current loglevel, the timestamp values will not be displayed. This configuration is set so that the timestamps may be displayed when almost nothing else is or just as part of the regular output.

6.7.1.17 SAVE_EARLY_TIMESTAMPS_TO_CMOS

Prompt: Save early TSC values to CMOS.

On platforms that do not have access to early **cbmem**, save the early TSC values into CMOS. This avoids having to find different register locations for each different chipset to stuff the values into.

6.7.1.18 USE_BLOBS

Prompt: Allow use of binary-only repository.

This draws in the blobs repository, which contains binary files that might be required for some chipsets or boards. This flag ensures that a "Free" option remains available for users.

6.7.1.19 COVERAGE

Prompt: Code coverage support

Add code coverage support for coreboot. This will store code coverage information in CBMEM for extraction from user space. If unsure, type **N**.

6.7.2 Mainboard Menu

6.7.2.1 COREBOOT_ROMSIZE_KB_64

Prompt: 64 KB

Choose this option if you have a 64 KB ROM chip.

6.7.2.2 COREBOOT_ROMSIZE_KB_128

Prompt: 128 KB

Choose this option if you have a 128 KB ROM chip.

6.7.2.3 COREBOOT_ROMSIZE_KB_256

Prompt: 256 KB

Choose this option if you have a 256 KB ROM chip.

6.7.2.4 COREBOOT_ROMSIZE_KB_512

Prompt: 512 KB

Choose this option if you have a 512 KB ROM chip.

6.7.2.5 COREBOOT_ROMSIZE_KB_1024

Prompt: 1024 KB (1 MB)

Choose this option if you have a 1024 KB (1 MB) ROM chip.

6.7.2.6 COREBOOT_ROMSIZE_KB_2048

Prompt: 2048 KB (2 MB)

Choose this option if you have a 2048 KB (2 MB) ROM chip.

6.7.2.7 COREBOOT_ROMSIZE_KB_4096

Prompt: 4096 KB (4 MB)

Choose this option if you have a 4096 KB (4 MB) ROM chip.

6.7.2.8 COREBOOT_ROMSIZE_KB_8192

Prompt: 8192 KB (8 MB)

Choose this option if you have a 8192 KB (8 MB) ROM chip.

6.7.2.9 COREBOOT_ROMSIZE_KB_12288

Prompt: 12288 KB (12 MB)

Choose this option if you have a 12288 KB (12 MB) ROM chip.

6.7.2.10 COREBOOT_ROMSIZE_KB_16384

Prompt: 16384 KB (16 MB)

Choose this option if you have a 16384 KB (16 MB) ROM chip.

6.7.2.11 MAINBOARD_SERIAL_NUMBER

Prompt: SMBIOS Serial Number

The Serial Number to store in SMBIOS structures.

6.7.2.12 MAINBOARD_VERSION

Prompt: SMBIOS Version Number

The Version Number to store in SMBIOS structures.

6.7.2.13 MAINBOARD_SMBIOS_MANUFACTURER

Prompt: SMBIOS Manufacturer

Override the default Manufacturer stored in SMBIOS structures.

6.7.2.14 MAINBOARD_SMBIOS_PRODUCT_NAME

Prompt: SMBIOS Product name

Override the default Product name stored in SMBIOS structures.

6.7.3 Architecture (x86) Menu

6.7.3.1 MARK_GRAPHICS_MEM_WRCOMB

Prompt: Mark graphics memory as write-combining.

The graphics performance may increase if the graphics memory is set as write-combining cache type. This option enables marking the graphics memory as write-combining.

6.7.3.2 UPDATE_IMAGE

Prompt: Update existing `coreboot.rom` image

If this option is enabled, no new `coreboot.rom` file is created. Instead, it is expected that there already is a suitable file for further processing. The bootblock will not be modified.

6.7.4 Chipset Menu

6.7.4.1 RESET_ON_INVALID_RAMSTAGE_CACHE

Prompt: Reset the system on S3 wake when ramstage cache invalid.

The Haswell romstage code caches the loaded ramstage program in SMM space. On S3 wake, the romstage will copy over a fresh ramstage that was cached in the SMM space. This option determines the action to take when the ramstage cache is invalid. If selected, the system will reset; otherwise, the ramstage will be reloaded from cbfs.

6.7.4.2 CACHE_ROM

Prompt: Allow for caching system ROM.

When selected, a variable range MTRR is allocated for coreboot, and the bootloader enables caching of the system ROM for faster access.

6.7.4.3 CPU_MICROCODE_CBFS_GENERATE

Prompt: Generate from tree.

Select this option if you want microcode updates to be assembled when building coreboot and included in the final image as a separate CBFS file. Microcode will not be hard-coded into ramstage.

The microcode file may be removed from the ROM image at a later time with cbfstool, if desired.

If unsure, select this option.

6.7.4.4 CPU_MICROCODE_CBFS_EXTERNAL

Prompt: Include external microcode file.

Select this option if you want to include an external file containing the CPU microcode. This will be included as a separate file in CBFS. *Only select this option if you are sure the microcode that you have is newer than the microcode shipping with coreboot.*

The microcode file may be removed from the ROM image at a later time with cbfstool, if desired.

If unsure, select "Generate from tree".

6.7.4.5 CPU_MICROCODE_CBFS_NONE

Prompt: Do not include microcode updates.

Select this option if you do not want CPU microcode included in CBFS.

Note

For some CPUs, the microcode is hard-coded into the source tree and is not loaded from CBFS. In this case, microcode will still be updated. There is a push to move all microcode to CBFS, but this change is not implemented for all CPUs.

This option currently applies to:

- Intel SandyBridge/IvyBridge
- VIA Nano

Microcode may be added to the ROM image at a later time with **cbfstool**, if desired. If unsure, select "Generate from tree".

Microcode updates intend to solve issues that have been discovered after CPU production. The expected effect is that systems work as intended with the updated microcode, but we have also seen cases where issues were solved by not applying microcode updates.

Note

Some operating system include these same microcode patches, so you may need to also disable microcode updates in your operating system for this option to have an effect.

Caution

Some CPUs depend on microcode updates to function correctly. Not updating the microcode may leave the CPU operating at less than optimal performance, or it may cause outright hangups. There are some CPUs that coreboot cannot properly initialize without microcode updates.

For example, if running with the factory microcode, some Intel SandyBridge CPUs may hang when enabling CAR, or some VIA Nano CPUs will hang when changing the frequency.

Make sure you have a way of flashing the ROM externally before selecting this option.

6.7.4.6 CPU_MICROCODE_FILE

Prompt: Path and filename of CPU microcode

The path and filename of the file containing the CPU microcode.

6.7.4.7 CPU_MICROCODE_CBFS_LEN

Prompt: Microcode length in CBFS

The microcode needs a specific length to get correctly detected and loaded by all CPUs.

6.7.4.8 HAVE_MRC

Prompt: Add a Memory Reference Code binary

Select this option to add a blob containing memory reference code.

Note

Without this binary, coreboot will not work.

6.7.4.9 MRC_FILE

Prompt: Intel System Agent path and filename

The path and filename of the file to use as System Agent binary.

6.7.4.10 CBFS_SIZE

Prompt: Size of CBFS filesystem in ROM

On Bay Trail systems the firmware image has to store a lot more than just coreboot, including:

- a firmware descriptor
- Intel Management Engine firmware
- MRC cache information

This option allows to limit the size of the CBFS portion in the firmware image.

6.7.4.11 LOCK_MANAGEMENT_ENGINE

Prompt: Lock Management Engine section

The Intel Management Engine supports preventing write accesses from the host to the Management Engine section in the firmware descriptor. If the ME section is locked, it can only be overwritten with an external SPI flash programmer. You will want this if you want to increase security of your ROM image once you are sure that the ME firmware is no longer going to change.

If unsure, type **N**.

6.7.4.12 ENABLE_BUILTIN_COM1

Prompt: Enable built-in COM1 Serial Port

The PMC has a legacy COM1 serial port. Choose this option to configure the pads and enable it. This serial port can be used for the debug console.

6.7.4.13 INCLUDE_BAYTRAIL_T_MICROCODE

Prompt: Include Microcode for Baytrail T processors

Include the microcode for the Baytrail T processor SKU

6.7.4.14 INCLUDE_BAYTRAIL_I_MICROCODE

Prompt: Include Microcode for Baytrail I processors

Include the microcode for the Baytrail I processor SKU

6.7.4.15 INCLUDE_BAYTRAIL_MD_MICROCODE

Prompt: Include Microcode for Baytrail M & D processors

Include the microcode for the Baytrail M & D processor SKUs

6.7.4.16 INCLUDE_ME

Prompt: Include the TXE

Build the txe and descriptor.bin into the ROM image. If you want to use a descriptor.bin and txe file from the previous ROM image, you may not want to build it in here.

6.7.4.17 HAVE_FSP_BIN

Prompt: Use Intel Firmware Support Package

Select this option to add an Intel FSP binary to the resulting coreboot image.

Note

Without this binary, coreboot builds relying on the FSP will not boot.

6.7.4.18 FSP_FILE

Prompt: Intel FSP binary path and filename

The path and filename of the Intel FSP binary for this platform.

6.7.4.19 FSP_LOC

Prompt: Intel FSP Binary location in CBFS

The location in CBFS that the FSP is located. This must match the value that is set in the FSP binary. If the FSP needs to be moved, rebase the FSP with the Intel's BCT (tool).

6.7.4.20 ENABLE_FAST_BOOT

Prompt: Enable Fast Boot

Enabling this feature will cause MRC data to be cached in NV storage, which will speed up boot time on future reboots and/or power cycles.

6.7.4.21 MRC_CACHE_SIZE

Prompt: Fastboot Data Cache Size

This is the amount of space in NV storage that is reserved for the fastboot data cache storage.

> **Warning**
>
> **Because this area will be erased and re-written, the size should be a full sector of the BIOS ROM, and nothing else should be included in CBFS in any sector that the Fastboot cache data is in.**

6.7.4.22 MRC_CACHE_LOC

Prompt: Fastboot Data Cache location in CBFS

The location in CBFS for the MRC data to be cached.

> **Warning**
>
> **This should be on a sector boundary of the BIOS ROM chip and nothing else should be included in that sector, or IT WILL BE ERASED.**

6.7.4.23 VIRTUAL_ROM_SIZE

Prompt: Virtual ROM Size

This is used to calculate the offset of the MRC data cache in NV Storage for "Fast Boot". If in doubt, leave this set to the default which sets the virtual size equal to the ROM size.

Example: Bakersport has 2 8 MB SPI ROMs. When the SPI ROMs are loaded with a 4 MB coreboot image, the virtual ROM size is 8 MB. When the SPI ROMs are loaded with an 8 MB coreboot image, the virtual ROM size is 16 MB.

6.7.4.24 CACHE_ROM_SIZE_OVERRIDE

Prompt: Cache ROM Size

This is the size of the cachable area that is passed into the FSP in the early initialization. Typically, this should be the size of the CBFS area, but the size must be a power of two, whereas the CBFS size does not have this limitation.

6.7.5 Devices Menu

6.7.5.1 VGA_ROM_RUN

Prompt: Run VGA Option ROMs

Execute VGA Option ROMs in coreboot if found. This is required to enable PCI/AGP/PCI-E video cards when not using a SeaBIOS payload.

When using a SeaBIOS payload it runs all option ROMs with much more complete BIOS interrupt services available than coreboot, which some option ROMs require in order to function correctly.

If unsure, type **N** when using SeaBIOS as payload, **Y** otherwise.

6.7.5.2 PCI_ROM_RUN

Prompt: Run non-VGA Option ROMs

Execute non-VGA PCI Option ROMs in coreboot if found.

Examples include IDE/SATA controller Option ROMs and Option ROMs for network cards (NICs).

When using a SeaBIOS payload it runs all option ROMs with much more complete BIOS interrupt services available than coreboot, which some option ROMs require in order to function correctly.

If unsure, type **N**when using SeaBIOS as payload, **Y** otherwise.

6.7.5.3 ON_DEVICE_ROM_RUN

Prompt: Run Option ROMs on PCI devices

Execute Option ROMs stored on PCI/PCIe/AGP devices in coreboot.

If disabled, only Option ROMs stored in CBFS will be executed by coreboot. If you are concerned about security, you might want to disable this option, but it might leave your system in a state of degraded functionality.

When using a SeaBIOS payload it runs all option ROMs with much more complete BIOS interrupt services available than coreboot, which some option ROMs require in order to function correctly.

If unsure, type **N** when using SeaBIOS as payload, **Y** otherwise.

6.7.5.4 PCI_OPTION_ROM_RUN_REALMODE

Prompt: Native mode

If you select this option, PCI Option ROMs will be executed natively on the CPU in real mode. No CPU emulation is involved, so this is the fastest, but also the least secure option. (only works on x86/x64 systems)

6.7.5.5 PCI_OPTION_ROM_RUN_YABEL

Prompt: Secure mode

If you select this option, the x86emu CPU emulator will be used to execute PCI Option ROMs.

This option prevents Option ROMs from doing dirty tricks with the system (such as installing SMM modules or hypervisors), but it is also significantly slower than the native Option ROM initialization method.

This is the default choice for non-x86 systems.

6.7.5.6 YABEL_PCI_ACCESS_OTHER_DEVICES

Prompt: Allow Option ROMs to access other devices

Per default, YABEL only allows Option ROMs to access the PCI device that they are associated with. However, this causes trouble for some onboard graphics chips whose Option ROM needs to reconfigure the northbridge.

6.7.5.7 YABEL_PCI_FAKE_WRITING_OTHER_DEVICES_CONFIG

Prompt: Fake success on writing other device's configuration space

By default, YABEL aborts when the Option ROM tries to write to other devices' configuration spaces. With this option enabled, the write doesn't follow through, but the Option ROM is allowed to go on. This can create issues such as hanging Option ROMs (if it depends on that other register changing to the written value), so test for impact before using this option.

6.7.5.8 YABEL_VIRTMEM_LOCATION

Prompt: Location of YABEL's virtual memory

YABEL requires 1 MB memory for its CPU emulation. This memory is normally located at 16 MB.

6.7.5.9 YABEL_DIRECTHW

Prompt: Direct hardware access

YABEL consists of two parts: It uses x86emu for the CPU emulation and additionally provides a PC system emulation that filters bad device and memory access (such as PCI configuration space access to other devices than the initialized one).

When choosing this option, x86emu will pass through all hardware accesses to memory and I/O devices to the underlying memory and I/O addresses. While this option prevents Option ROMs from doing dirty tricks with the CPU (such as installing SMM modules or hypervisors), they can still access all devices in the system. Enable this option for a good compromise between security and speed.

6.7.5.10 PCIEXP_COMMON_CLOCK

Prompt: Enable PCIe Common Clock

Detect and enable Common Clock on PCIe links.

6.7.5.11 PCIEXP_ASPM

Prompt: Enable PCIe ASPM

Detect and enable ASPM on PCIe links.

6.7.6 VGA BIOS Menu

6.7.6.1 VGA_BIOS

Prompt: Add a VGA BIOS image

Select this option if you have a VGA BIOS image that you would like to add to your ROM.

You will be able to specify the location and file name of the image later.

6.7.6.2 VGA_BIOS_FILE

Prompt: VGA BIOS path and filename

The path and filename of the file to use as VGA BIOS.

6.7.6.3 ONBOARD_VGA_IS_PRIMARY

Prompt: Force onboard VGA as primary video device

If not selected, the last adapter found will be used, allowing for PCI/PCIe graphics cards to be primary. Typically this should be set to **n**.

6.7.6.4 VGA_BIOS_PUT_IN_VGAROMS

Prompt: Put the VGA BIOS in CBFS vgaroms directory

All option ROMs in the CBFS vgaroms directory are loaded. The VGA BIOS vendor and device IDs are ignored for the purpose of determining which option ROMs to load when this option is selected.

6.7.6.5 VGA_BIOS_ADDRESS

Prompt: Optional VGA BIOS address in CBFS

Place the VGA BIOS at a specific address. If left blank, the VGA BIOS will be placed at an unused address range in CBFS. If an address is specified, it must point to an unused range within the boot ROM device.

6.7.6.6 MBI_FILE

Prompt: Intel MBI path and filename

The path and filename of the file to use as VGA BIOS.

6.7.7 Display Menu

6.7.7.1 FRAMEBUFFER_SET_VESA_MODE

Prompt: Set framebuffer graphics resolution

Set VESA/native framebuffer mode (needed for bootsplash and graphical framebuffer console)

6.7.7.2 FRAMEBUFFER_KEEP_VESA_MODE

Prompt: Keep VESA framebuffer

This option keeps the framebuffer mode set after coreboot finishes execution. If this option is enabled, coreboot will pass a framebuffer entry in its coreboot table and the payload will need a framebuffer driver. If this option is disabled, coreboot will switch back to text mode before handing control to a payload.

6.7.7.3 BOOTSPLASH

Prompt: Show graphical bootsplash

This option shows a graphical bootsplash screen. The grapics are loaded from the CBFS file bootsplash.jpg.

6.7.7.4 BOOTSPLASH_FILE

Prompt: Bootsplash path and filename

The path and filename of the file to use as graphical bootsplash screen. The file format has to be jpg.

6.7.7.5 BOOTSPLASH_ADDRESS

Prompt: Bootsplash image address in CBFS

Place the bootsplash image at a specific address. If left blank, the bootsplash image will be placed at an unused address range in CBFS. If an address is specified, it must point to an unused range within the boot ROM device.

6.7.8 PXE ROM Menu

6.7.8.1 PXE_ROM

Prompt: Add a PXE ROM image

Select this option if you have a PXE ROM image that you would like to add to your ROM.

6.7.8.2 PXE_ROM_FILE

Prompt: PXE ROM filename

The path and filename of the file to use as PXE ROM.

6.7.8.3 PXE_ROM_PUT_IN_GENROMS

Prompt: Put the PXE ROM in CBFS genroms directory

All option ROMs in the CBFS genroms directory are loaded. The PXE ROM vendor and device IDs are ignored for the purpose of determining which option ROMs to load when this option is selected.

6.7.8.4 PXE_ROM_ID

Prompt: network card PCI IDs

The comma-separated PCI vendor and device ID that would associate your PXE ROM to your network card.

Example: 10ec,8168

In the above example 10ec is the PCI vendor ID (in hex, but without the "0x" prefix) and 8168 specifies the PCI device ID of the network card (also in hex, without "0x" prefix).

Under GNU/Linux you can run **lspci -nn** to list the IDs of your PCI devices.

6.7.8.5 PXE_ROM_ADDRESS

Prompt: PXE ROM address in CBFS

Place the PXE ROM at a specific address. If left blank, the PXE ROM will be placed at an unused address range in CBFS. If an address is specified, it must point to an unused range within the boot ROM device.

6.7.8.6 SUBSYSTEM_VENDOR_ID

Prompt: Override PCI Subsystem Vendor ID

This configuration option will override the device tree settings for PCI Subsystem Vendor ID.

6.7.8.7 SUBSYSTEM_DEVICE_ID

Prompt: Override PCI Subsystem Device ID

This configuration option will override the device tree settings for PCI Subsystem Device ID.

6.7.9 Generic Drivers Menu

6.7.9.1 ELOG

Prompt: Support for flash based event log

Enable support for flash based event logging.

6.7.9.2 ELOG_FLASH_BASE

Prompt: Event log offset into flash

Offset into the flash chip for the ELOG block. This should be allocated in the FMAP.

6.7.9.3 ELOG_AREA_SIZE

Prompt: Size of Event Log area in flash

This should be a multiple of flash block size.

Default is 4K.

6.7.9.4 ELOG_FULL_THRESHOLD

Prompt: Threshold at which flash is considered full

When the Event Log size is larger than this it will be shrunk to ELOG_SHRINK_SIZE. Must be greater than ELOG_AREA_SIZE, and ELOG_AREA_SIZE - ELOG_FULL_THRESHOLD must be greater than the maximum event size of 128.

Default is 75% of the log, or 3K.

6.7.9.5 ELOG_SHRINK_SIZE

Prompt: Resulting size when the event log is shrunk

When the Event Log is shrunk it will go to this size. ELOG_AREA_SIZE - ELOG_SHRINK_SIZE must be less than CONFIG_ELOG_FULL_THRESHOLD.

Default is 1K.

6.7.9.6 ELOG_CBMEM

Prompt: Store a copy of ELOG in CBMEM

This option will have ELOG store a copy of the flash event log in a CBMEM region and export that address in SMBIOS to the OS. This is useful if the ELOG location is not in memory-mapped flash, but it means that events added at runtime via the SMI handler will not be reflected in the CBMEM copy of the log.

6.7.9.7 ELOG_GSMI

Prompt: SMI interface to write and clear event log

This interface is compatible with the Linux kernel driver available with CONFIG_GOOGLE_GSMI and can be used to write kernel reset/shutdown messages to the event log.

6.7.9.8 ELOG_BOOT_COUNT

Prompt: Maintain a monotonic boot number in CMOS

Store a monotonic boot number in CMOS and provide an interface to read the current value and increment the counter. This boot counter will be logged as part of the System Boot event.

6.7.9.9 ELOG_BOOT_COUNT_CMOS_OFFSET

Prompt: Offset in CMOS to store the boot count

This value must be greater than 16 bytes so as not to interfere with the standard RTC region. Requires 8 bytes.

6.7.9.10 DRIVERS_OXFORD_OXPCIE

Prompt: Oxford OXPCIe952

Support for Oxford OXPCIe952 serial port PCIe cards. Currently only devices with the vendor ID 0x1415 and device ID 0xc158 will work.

Note

You must set the base address of your OXPCIe952 card to exactly the value that the device allocator would set them later on, or serial console functionality will stop as soon as the resource allocator assigns a new base address to the device.

6.7.9.11 OXFORD_OXPCIE_BRIDGE_BUS

Prompt: OXPCIe's PCIe bridge bus number

While coreboot is executing code from ROM, the coreboot resource allocator has not been running yet. Hence, PCI devices living behind a bridge are not yet visible to the system. In order to use an OXPCIe952 based PCIe card, coreboot has to set up the PCIe bridge that controls the OXPCIe952 controller first.

6.7.9.12 OXFORD_OXPCIE_BRIDGE_DEVICE

Prompt: OXPCIe's PCIe bridge device number

While coreboot is executing code from ROM, the coreboot resource allocator has not been running yet. Hence, PCI devices living behind a bridge are not yet visible to the system. In order to use an OXPCIe952 based PCIe card, coreboot has to set up the PCIe bridge that controls the OXPCIe952 controller first.

6.7.9.13 OXFORD_OXPCIE_BRIDGE_FUNCTION

Prompt: OXPCIe's PCIe bridge function number

While coreboot is executing code from ROM, the coreboot resource allocator has not been running yet. Hence, PCI devices living behind a bridge are not yet visible to the system. In order to use an OXPCIe952 based PCIe card, coreboot has to set up the PCIe bridge that controls the OXPCIe952 controller first.

6.7.9.14 OXFORD_OXPCIE_BRIDGE_SUBORDINATE

Prompt: OXPCIe's PCIe bridge subordinate bus

While coreboot is executing code from ROM, the coreboot resource allocator has not been running yet. Hence, PCI devices living behind a bridge are not yet visible to the system. In order to use an OXPCIe952 based PCIe card, coreboot has to set up the PCIe bridge that controls the OX-PCIe952 controller first.

6.7.9.15 OXFORD_OXPCIE_BASE_ADDRESS

Prompt: Base address for rom stage console

While coreboot is executing code from ROM, the coreboot resource allocator has not been running yet. Hence, PCI devices living behind a bridge are not yet visible to the system. In order to use an OXPCIe952 based PCIe card, coreboot has to set up a temporary address for the OXPCIe952 controller.

6.7.9.16 DRIVERS_PS2_KEYBOARD

Prompt: PS/2 keyboard init

Enable this option to initialize PS/2 keyboards found connected to the PS/2 port.

Some payloads (e.g., filo) require this option. Other payloads (e.g., SeaBIOS, Linux) do not require it. Initializing a PS/2 keyboard can take several hundred milliseconds.

If you know you will only use a payload which does not require this option, then you can type **N** here to speed up boot time. Otherwise type **Y**.

6.7.9.17 RTL8168_ROM_DISABLE

Prompt: Disable RTL8168 ROM

Just enough of a driver to make coreboot not look for an Option ROM. No configuration is necessary for the OS to pick up the device.

6.7.9.18 DRIVERS_SIL_3114

Prompt: Silicon Image SIL3114

It sets PCI class to IDE compatible native mode, allowing SeaBIOS, FILO etc... to boot from it.

6.7.9.19 SPI_FLASH_SMM

Prompt: SPI flash driver support in SMM

Select this option if you want SPI flash support in SMM.

6.7.9.20 SPI_FLASH_NO_FAST_READ

Prompt: Disable Fast Read command

Select this option if your setup requires to avoid "fast reads" from the SPI flash parts.

6.7.10 Console Menu

6.7.10.1 BOOTBLOCK_CONSOLE

Prompt: Enable early (bootblock) console output.

Use console during the bootblock if supported

6.7.10.2 EARLY_CONSOLE

Prompt: Enable early (pre-RAM) console output.

Use console during early (pre-RAM) boot stages

6.7.10.3 SQUELCH_EARLY_SMP

Prompt: Squelch AP CPUs from early console.

When selected only the BSP CPU will output to early console.

Console drivers have unpredictable behaviour if multiple threads attempt to share the same resources without a spinlock.

If unsure, type **Y**.

6.7.10.4 CONSOLE_SERIAL

Prompt: Serial port console output

Send coreboot debug output to a serial port (should be one or more of CONSOLE_SERIAL8250, CONSOLE_SERIAL8250MEM, CONSOLE_SERIAL_UART)

6.7.10.5 CONSOLE_SERIAL8250

Prompt: Serial port console output (I/O mapped, 8250-compatible)

Send coreboot debug output to an I/O mapped serial port console.

6.7.10.6 CONSOLE_SERIAL8250MEM

Prompt: Serial port console output (memory-mapped, 8250-compatible)

Send coreboot debug output to a memory-mapped serial port console.

6.7.10.7 CONSOLE_SERIAL_UART

Prompt: Serial port console output (device-specific UART)

Send coreboot debug output to a device-specific serial port console.

6.7.10.8 CONSOLE_SERIAL_COM1

Prompt: COM1/ttyS0, I/O port 0x3f8

Serial console on COM1/ttyS0 at I/O port 0x3f8.

6.7.10.9 CONSOLE_SERIAL_COM2

Prompt: COM2/ttyS1, I/O port 0x2f8

Serial console on COM2/ttyS1 at I/O port 0x2f8.

6.7.10.10 CONSOLE_SERIAL_COM3

Prompt: COM3/ttyS2, I/O port 0x3e8

Serial console on COM3/ttyS2 at I/O port 0x3e8.

6.7.10.11 CONSOLE_SERIAL_COM4

Prompt: COM4/ttyS3, I/O port 0x2e8

Serial console on COM4/ttyS3 at I/O port 0x2e8.

6.7.10.12 CONSOLE_SERIAL_IO_328

Prompt: I/O port 0x328

Serial console at I/O port 0x328.

6.7.10.13 CONSOLE_SERIAL_115200

Prompt: 115200

Set serial port Baud rate to 115200.

6.7.10.14 CONSOLE_SERIAL_57600

Prompt: 57600

Set serial port Baud rate to 57600.

6.7.10.15 CONSOLE_SERIAL_38400

Prompt: 38400

Set serial port Baud rate to 38400.

6.7.10.16 CONSOLE_SERIAL_19200

Prompt: 19200

Set serial port Baud rate to 19200.

6.7.10.17 CONSOLE_SERIAL_9600

Prompt: 9600

Set serial port Baud rate to 9600.

6.7.10.18 SPKMODEM

Prompt: spkmodem (console on speaker) console output

Send coreboot debug output through speaker

6.7.10.19 CONSOLE_NE2K

Prompt: Network console over NE2000 compatible Ethernet adapter

Send coreboot debug output to a Ethernet console, it works same way as Linux netconsole, packets are received to UDP port 6666 on IP/MAC specified with options bellow. Use following netcat command: **nc -u -l -p 6666**

6.7.10.20 CONSOLE_NE2K_DST_MAC

Prompt: Destination MAC address of remote system

Type in either MAC address of logging system or MAC address of the router.

6.7.10.21 CONSOLE_NE2K_DST_IP

Prompt: Destination IP of logging system

This is IP adress of the system running, for example netcat command to dump the packets.

6.7.10.22 CONSOLE_NE2K_SRC_IP

Prompt: IP address of coreboot system

This is the IP of the coreboot system

6.7.10.23 CONSOLE_NE2K_IO_PORT

Prompt: NE2000 adapter fixed IO port address

This is the I/O port address for the I/O port on the card, please select some non-conflicting region, 32 bytes of I/O spaces will be used (and align on 32 bytes boundary, qemu needs broader align)

6.7.10.24 CONSOLE_CBMEM

Prompt: Send console output to a CBMEM buffer

Enable this to save the console output in a CBMEM buffer. This would allow to see coreboot console output from Linux space.

6.7.10.25 CONSOLE_CBMEM_BUFFER_SIZE

Prompt: Room allocated for console output in CBMEM

Space allocated for console output storage in CBMEM. The default value (64K or 0x10000 bytes) is large enough to accommodate even the BIOS_SPEW level.

6.7.10.26 CONSOLE_CAR_BUFFER_SIZE

Prompt: Room allocated for console output in Cache as RAM

Console is used before RAM is initialized. This is the room reserved in the DCACHE based RAM to keep console output before it can be saved in a CBMEM buffer. 3K bytes should be enough even for the BIOS_SPEW level.

6.7.10.27 CONSOLE_QEMU_DEBUGCON

Prompt: QEMU debug console output

Send coreboot debug output to QEMU's isa-debugcon device: **qemu-system-x86_64 \ -chardev file,id=debugcon,path=/dir/file.log \ -device isa-debugcon,iobase=0x402,chardev=debugcon**

6.7.10.28 DEFAULT_CONSOLE_LOGLEVEL_8

Prompt: 8: SPEW

Way too many details.

6.7.10.29 DEFAULT_CONSOLE_LOGLEVEL_7

Prompt: 7: DEBUG

Debug-level messages.

6.7.10.30 DEFAULT_CONSOLE_LOGLEVEL_6

Prompt: 6: INFO

Informational messages.

6.7.10.31 DEFAULT_CONSOLE_LOGLEVEL_5

Prompt: 5: NOTICE

Normal but significant conditions.

6.7.10.32 DEFAULT_CONSOLE_LOGLEVEL_4

Prompt: 4: WARNING

Warning conditions.

6.7.10.33 DEFAULT_CONSOLE_LOGLEVEL_3

Prompt: 3: ERR

Error conditions.

6.7.10.34 DEFAULT_CONSOLE_LOGLEVEL_2

Prompt: 2: CRIT

Critical conditions.

6.7.10.35 DEFAULT_CONSOLE_LOGLEVEL_1

Prompt: 1: ALERT

Action must be taken immediately.

6.7.10.36 DEFAULT_CONSOLE_LOGLEVEL_0

Prompt: 0: EMERG

System is unusable.

6.7.10.37 DIE_CONSOLE_LOGLEVEL

Prompt: Messages at or below this level halt the system

Set the log level to force a "die" after. If a console message is output that is at this level or lower the system will die after the message. Enter -1 to disable this feature. The DEFAULT_CONSOLE_LOGLEVEL must be set to a value >= this level.

6.7.10.38 CONSOLE_POST

Prompt: Show POST codes on the debug console

If enabled, coreboot will additionally print POST codes (which are usually displayed using a so-called "POST card" ISA/PCI/PCI-E device) on the debug console.

6.7.10.39 CMOS_POST

Prompt: Store post codes in CMOS for debugging

If enabled, coreboot will store post codes in CMOS and switch between two offsets on each boot so the last post code in the previous boot can be retrieved. This uses 3 bytes of CMOS.

6.7.10.40 CMOS_POST_OFFSET

Prompt: Offset into CMOS to store POST codes

If CMOS_POST is enabled then an offset into CMOS must be provided. If CONFIG_HAVE_OPTION_TABLE is enabled then it will use the value defined in the mainboard option table.

6.7.10.41 CMOS_POST_EXTRA

Prompt: Store extra logging information into CMOS

This will enable extra logging of work that happens between post codes into CMOS for debug. This uses an additional 8 bytes of CMOS.

6.7.10.42 IO_POST

Prompt: Send POST codes to an I/O port

If enabled, POST codes will be written to an I/O port.

6.7.10.43 IO_POST_PORT

Prompt: I/O port for POST codes

POST codes on x86 are typically written to the LPC bus on port 0x80. However, it may be desirable to change the port number depending on the presence of coprocessors/microcontrollers or if the platform does not support I/O in the conventional x86 manner.

6.7.11 RELOCATABLE_MODULES

Prompt: Relocatable Modules

If RELOCATABLE_MODULES is selected then support is enabled for building relocatable modules in the ram stage. Those modules can be loaded anywhere and all the relocations are handled automatically.

6.7.12 RELOCATABLE_RAMSTAGE

Prompt: Build the ramstage to be relocatable in 32-bit address space.

The relocatable ramstage support allows for the ramstage to be built as a relocatable module. The stage loader can identify a place out of the OS way so that copying memory is unnecessary during an S3 wake. When selecting this option the romstage is responsible for determining a stack location to use for loading the ramstage.

6.7.13 CACHE_RELOCATED_RAMSTAGE_OUTSIDE_CBMEM

Prompt: Cache the relocated ramstage outside of cbmem.

The relocated ramstage is saved in an area specified by the by the board and/or chipset.

6.7.14 HAVE_REFCODE_BLOB

Prompt: An external reference code blob should be put into cbfs.

The reference code blob will be placed into cbfs.

6.7.15 REFCODE_BLOB_FILE

Prompt: Path and filename to reference code blob.

The path and filename to the file to be added to cbfs.

6.7.16 System tables Menu

6.7.16.1 GENERATE_ACPI_TABLES

Prompt: Generate ACPI tables

Generate ACPI tables for this board.

If unsure, type **Y**.

6.7.16.2 GENERATE_SPCR_ACPI_TABLE

Prompt: Generate Serial Port Console Redirection Table (SPCR)

Generate an ACPI SPCR (Serial Port Console Redirection) table. "This table is used to indicate whether a serial port or a non-legacy UART interface is available for use with Microsoft Windows Emergency Management Services (EMS)." It is also used by BITS to determine the serial port. It no longer seems to be used by Linux, so the value is somewhat questionable.

For more information, look at the Reference Serial Port Console Redirection Table, Version 1.00, January 11, 2002 at http://msdn.microsoft.com/en-us/windows/hardware/gg487465

If unsure, type **Y**.

6.7.16.3 GENERATE_MP_TABLE

Prompt: Generate an MP table

Generate an MP table (conforming to the Intel MultiProcessor specification 1.4) for this board.

If unsure, type **Y**.

6.7.16.4 GENERATE_PIRQ_TABLE

Prompt: Generate a PIRQ table

Generate a PIRQ table for this board.

If unsure, type **Y**.

6.7.16.5 GENERATE_SMBIOS_TABLES

Prompt: Generate SMBIOS tables

Generate SMBIOS tables for this board.

If unsure, type **Y**.

6.7.17 Payload Menu

6.7.17.1 PAYLOAD_NONE

Prompt: None

Select this option if you want to create an "empty" coreboot ROM image for a certain mainboard, i.e., a coreboot ROM image which does not yet contain a payload.

For such an image to be useful, you have to use **cbfstool** to add a payload to the ROM image later.

6.7.17.2 PAYLOAD_ELF

Prompt: An ELF executable payload

Select this option if you have a payload image (an ELF file) which coreboot should run as soon as the basic hardware initialization is completed.

You will be able to specify the location and file name of the payload image later.

6.7.17.3 PAYLOAD_LINUX

Prompt: A Linux payload

Select this option if you have a Linux bzImage which coreboot should run as soon as the basic hardware initialization is completed.

You will be able to specify the location and file name of the payload image later.

6.7.17.4 PAYLOAD_FILE

Prompt: Payload path and filename

The path and filename of the ELF executable file to use as payload.

6.7.17.5 COMPRESSED_PAYLOAD_LZMA

Prompt: Use LZMA compression for payload

In order to reduce the size payloads take up in the ROM chip coreboot can compress them using the LZMA algorithm.

6.7.17.6 PAYLOAD_BASE

Prompt: Payload base address

Select a base address in memory for the payload. A zero tells cbfstool to choose the location for you. This is calculated as `0xFFFFFFFF - rom size + payload location *in the rom* +1`.

Note

The base address is where the payload binary starts. The header starts 0x40 bytes lower.

6.7.17.7 EXTRA_1ST_COMPRESSED_PAYLOAD_LZMA

Prompt: Use LZMA compression for extra payload

In order to reduce the size payloads take up in the ROM chip coreboot can compress them using the LZMA algorithm.

6.7.17.8 EXTRA_1ST_PAYLOAD_BASE

Prompt: Extra payload base address

Select a base address in memory for the payload. A zero tells cbfstool to choose the location for you. This is calculated as `0xFFFFFFFF - rom size + payload location *in the rom* +1`.

Note

The base address is where the payload binary starts. The header starts 0x40 bytes lower.

6.7.17.9 EXTRA_2ND_COMPRESSED_PAYLOAD_LZMA

Prompt: Use LZMA compression for extra payload

In order to reduce the size payloads take up in the ROM chip, coreboot can compress them using the LZMA algorithm.

6.7.17.10 EXTRA_2ND_PAYLOAD_BASE

Prompt: Extra payload base address

Select a base address in memory for the payload. A zero tells cbfstool to choose the location for you. This is calculated as `0xFFFFFFFF - rom size + payload location *in the rom* +1`.

Note

The base address is where the payload binary starts. The header starts 0x40 bytes lower.

6.7.17.11 EXTRA_3RD_COMPRESSED_PAYLOAD_LZMA

Prompt: Use LZMA compression for extra payload

In order to reduce the size payloads take up in the ROM chip coreboot can compress them using the LZMA algorithm.

6.7.17.12 EXTRA_3RD_PAYLOAD_BASE

Prompt: Extra payload base address

Select a base address in memory for the payload. A zero tells cbfstool to choose the location for you. This is calculated as `0xFFFFFFFF - rom size + payload location *in the rom* +1`.

Note

The base address is where the payload binary starts. The header starts 0x40 bytes lower.

6.7.17.13 EXTRA_4TH_COMPRESSED_PAYLOAD_LZMA

Prompt: Use LZMA compression for extra payload

In order to reduce the size payloads take up in the ROM chip coreboot can compress them using the LZMA algorithm.

6.7.17.14 EXTRA_4TH_PAYLOAD_BASE

Prompt: Extra payload base address

Select a base address in memory for the payload. A zero tells cbfstool to choose the location for you. This is calculated as `0xFFFFFFFF - rom size + payload location *in the rom* +1`.

Note

The base address is where the payload binary starts. The header starts 0x40 bytes lower.

6.7.17.15 PAYLOAD_USE_CONFIG_FILE

Prompt: Payload saved configuration path and filename

Specify the saved configuration file that is passed to the payload makefile for payload projects that use Kconfig. Example: **payloads/seabios/configs/config.default**

Leave blank if the project does not use Kconfig. Example: explorer does not use Kconfig

6.7.17.16 LINUX_COMMAND_LINE

Prompt: Linux command line

A command line to add to the Linux kernel.

6.7.17.17 LINUX_INITRD

Prompt: Linux initrd

An initrd image to add to the Linux kernel.

6.7.17.18 SEABIOS_ENABLE_OPTIONS

Prompt: Enable SeaBIOS Payload Options

Enables various options to configure SeaBIOS

6.7.17.19 SeaBIOS Menu

6.7.17.19.1 SEABIOS_ADD_BOOTORDER

Prompt: Add a Bootorder file for SeaBIOS to CBFS

Add a bootorder file to control the boot order in SeaBIOS. For more information, go to http://www.coreboot.org/SeaBIOS#Configuring_boot_order

6.7.17.19.2 SEABIOS_BOOTORDER_FILE

Prompt: bootorder source path and filename

Add a bootorder file to control the boot order in SeaBIOS. For more information, go to http://www.coreboot.org/SeaBIOS#Configuring_boot_order

6.7.17.19.3 SEABIOS_ADD_BOOT_MENU_WAIT

Prompt: Add a BOOT-MENU-WAIT file for SeaBIOS to CBFS

This modifies SeaBIOS's default delay when pausing at the "Press F12 for Boot Menu" prompt.

6.7.17.19.4 SEABIOS_BOOT_MENU_WAIT_TIME

Prompt: BOOT-MENU-WAIT time in msecs

This modifies SeaBIOS's default delay when pausing at the "Press F12 for Boot Menu" prompt

6.7.18 Additional Option ROM Menu

6.7.18.1 OPTION_ROM

Prompt: Add an additional option ROM

Select this option if you have an additional option ROM that you would like to add to the CBFS image.

You will be able to specify the location and file name of the image later.

6.7.18.2 OPTION_ROM_FILE_IN

Prompt: Option ROM source path and filename

The path and filename of the option ROM you wish to add.

6.7.18.3 OPTION_ROM_FILE_OUT

Prompt: Option ROM destination in CBFS

The destination path and name in the CBFS.

6.7.18.4 OPTION_ROM_ADDRESS

Prompt: Option ROM address in CBFS

Place the option ROM at a specific address. If left blank, the option ROM will be placed at an unused address range in CBFS. If an address is specified, it must point to an unused range within the boot ROM device.

6.7.19 Additional File in CBFS Menu

6.7.19.1 CBFS_FILE

Prompt: Add an additional file

Select this option if you have an generic file that you would like to add to the CBFS image.

You will be able to specify the location and file name of the image later.

6.7.19.2 CBFS_FILE_FILE_IN

Prompt: File source path and filename

The path and filename of the file you wish to add.

6.7.19.3 CBFS_FILE_FILE_OUT

Prompt: File destination in CBFS

The destination path and name in the CBFS.

6.7.19.4 CBFS_FILE_ADDRESS

Prompt: File address in CBFS

Place the file at a specific address. If left blank, the file will be placed at an unused address range in CBFS. If an address is specified, it must point to an unused range within the boot ROM device.

6.7.19.5 CBFS_2ND_FILE

Prompt: Add a 2nd additional file

Select this option if you have an 2nd generic file that you would like to add to the CBFS image.

You will be able to specify the location and file name of the image later.

6.7.19.6 CBFS_2ND_FILE_FILE_IN

Prompt: File source path and filename

The path and filename of the file you wish to add.

6.7.19.7 CBFS_2ND_FILE_FILE_OUT

Prompt: File destination in CBFS

The destination path and name in the CBFS.

6.7.19.8 CBFS_2ND_FILE_ADDRESS

Prompt: File address in CBFS

Place the file at a specific address. If left blank, the file will be placed at an unused address range in CBFS. If an address is specified, it must point to an unused range within the boot ROM device.

6.7.19.9 CBFS_3RD_FILE

Prompt: Add a 3rd additional file

Select this option if you have a 3rd generic file that you would like to add to the CBFS image.

You will be able to specify the location and file name of the image later.

6.7.19.10 CBFS_3RD_FILE_FILE_IN

Prompt: File source path and filename

The path and filename of the file you wish to add.

6.7.19.11 CBFS_3RD_FILE_FILE_OUT

Prompt: File destination in CBFS

The destination path and name in the CBFS.

6.7.19.12 CBFS_3RD_FILE_ADDRESS

Prompt: File address in CBFS

Place the file at a specific address. If left blank, the file will be placed at an unused address range in CBFS. If an address is specified, it must point to an unused range within the boot ROM device.

6.7.19.13 CBFS_4TH_FILE

Prompt: Add a 4th additional file

Select this option if you have a 4th generic file that you would like to add to the CBFS image.

You will be able to specify the location and file name of the image later.

6.7.19.14 CBFS_4TH_FILE_FILE_IN

Prompt: File source path and filename

The path and filename of the file you wish to add.

6.7.19.15 CBFS_4TH_FILE_FILE_OUT

Prompt: File destination in CBFS

The destination path and name in the CBFS.

6.7.19.16 CBFS_4TH_FILE_ADDRESS

Prompt: File address in CBFS

Place the file at a specific address. If left blank, the file will be placed at an unused address range in CBFS. If an address is specified, it must point to an unused range within the boot ROM device.

6.7.20 Debugging Menu

6.7.20.1 GDB_STUB

Prompt: GDB debugging support

If enabled, you will be able to set breakpoints for gdb debugging. See `src/arch/x86/lib/c_start.S` for details.

6.7.20.2 GDB_WAIT

Prompt: Wait for a GDB connection

If enabled, coreboot will wait for a GDB connection.

6.7.20.3 DEBUG_CBFS

Prompt: Output verbose CBFS debug messages

This option enables additional CBFS related debug messages.

6.7.20.4 DEBUG_PIRQ

Prompt: Check PIRQ table consistency

If unsure, type **N**.

6.7.20.5 DEBUG_MALLOC

Prompt: Output verbose malloc debug messages

This option enables additional malloc related debug messages.

Note

This option will increase the size of the coreboot image.

If unsure, type **N**.

6.7.20.6 DEBUG_ACPI

Prompt: Output verbose ACPI debug messages

This option enables additional ACPI related debug messages.

Note

This option will slightly increase the size of the coreboot image.

If unsure, type **N**.

6.7.20.7 DEBUG_MP_TABLE

Prompt: Output verbose MP_Table debug messages

This option enables additional MP_Table related debug messages.

Note

This option will slightly increase the size of the coreboot image.

If unsure, type **N**.

6.7.20.8 REALMODE_DEBUG

Prompt: Enable debug messages for option ROM execution

This option enables additional x86emu related debug messages.

Note

This option will increase the time to emulate a ROM.

If unsure, type **N**.

6.7.20.9 X86EMU_DEBUG

Prompt: Output verbose x86emu debug messages

This option enables additional x86emu related debug messages.

Note

This option will increase the size of the coreboot image.

If unsure, type **N**.

6.7.20.10 X86EMU_DEBUG_JMP

Prompt: Trace JMP/RETF

Print information about JMP and RETF opcodes from x86emu.

Note

This option will increase the size of the coreboot image.

If unsure, type **N**.

6.7.20.11 X86EMU_DEBUG_TRACE

Prompt: Trace all opcodes

Print *all* opcodes that are executed by x86emu.

Caution

This will produce a LOT of output and take a long time.

Note

This option will increase the size of the coreboot image.

If unsure, type **N**.

6.7.20.12 X86EMU_DEBUG_PNP

Prompt: Log Plug&Play accesses

Print Plug And Play accesses made by option ROMs.

Note

This option will increase the size of the coreboot image.

If unsure, type **N**.

6.7.20.13 X86EMU_DEBUG_DISK

Prompt: Log Disk I/O

Print Disk I/O related messages.

Note

This option will increase the size of the coreboot image.

If unsure, type **N**.

6.7.20.14 X86EMU_DEBUG_PMM

Prompt: Log PMM

Print messages related to POST Memory Manager (PMM).

Note

This option will increase the size of the coreboot image.

If unsure, type **N**.

6.7.20.15 X86EMU_DEBUG_VBE

Prompt: Debug VESA BIOS Extensions

Print messages related to VESA BIOS Extension (VBE) functions.

Note

This option will increase the size of the coreboot image.

If unsure, type **N**.

6.7.20.16 X86EMU_DEBUG_INT10

Prompt: Redirect INT10 output to console

Let INT10 (i.e., character output) calls print messages to debug output.

Note

This option will increase the size of the coreboot image.

If unsure, type **N**.

6.7.20.17 X86EMU_DEBUG_INTERRUPTS

Prompt: Log intXX calls

Print messages related to interrupt handling.

Note

This option will increase the size of the coreboot image.

If unsure, type **N**.

6.7.20.18 X86EMU_DEBUG_CHECK_VMEM_ACCESS

Prompt: Log special memory accesses

Print messages related to accesses to certain areas of the virtual memory (e.g., BIOS Data Area, BDA, or interrupt vectors)

Note

This option will increase the size of the coreboot image.

If unsure, type **N**.

6.7.20.19 X86EMU_DEBUG_MEM

Prompt: Log all memory accesses

Print memory accesses made by option ROM. This also includes accesses to fetch instructions.

Note

This option will increase the size of the coreboot image.

If unsure, type **N**.

6.7.20.20 X86EMU_DEBUG_IO

Prompt: Log IO accesses

Print I/O accesses made by option ROM.

Note

This option will increase the size of the coreboot image.

If unsure, type **N**.

6.7.20.21 DEBUG_SPI_FLASH

Prompt: Output verbose SPI flash debug messages

This option enables additional SPI flash related debug messages.

6.7.20.22 DEBUG_USBDEBUG

Prompt: Output verbose USB 2.0 EHCI debug dongle messages

This option enables additional USB 2.0 debug dongle related messages.

Select this to debug the connection of usbdebug dongle.

Note

You need some other working console to receive the messages.

6.7.20.23 TRACE

Prompt: Trace function calls

If enabled, every function will print information to console once the function is entered. The syntax is ~0xaaaabbbb(0xccccdddd) the 0xaaaabbbb is the actual function and 0xccccdddd is EIP of calling function. Please note some printk related functions are omitted from trace to have good looking console dumps.

6.7.20.24 DEBUG_COVERAGE

Prompt: Debug code coverage

If enabled, the code coverage hooks in coreboot will output some information about the coverage data that is dumped.

6.7.20.25 POISON_STACK

Prompt: 'Poison the stack' by writing data into it

"Poisoning the stack" helps to uncover places where the stack is used before it is initialized. This involves writing a pattern to the stack area when the stack is created instead of clearing it or leaving the values that were previously in those memory locations. This also allows a developer to look at the stack and see just how much was used.

6.7.21 Exclusions Menu

6.7.21.1 EXCLUDE_CLASSES_BOOTBLOCK

Prompt: Exclude the bootblock classes from the project build

If selected, the bootblock classes will not be built. Use this only on platforms where the bootblock is highly customized.

6.7.21.2 EXCLUDE_CLASSES_SMM

Prompt: Exclude the SMM classes from the project build

If selected, the SMM support code will not be built. Use this only on platforms where SMM support is not desired.

6.7.21.3 EXCLUDE_CLASSES_RMODULES

Prompt: Exclude the rmodules (memset, memcpy, etc.) classes from the project build

If selected, the run-time support modules will not be built. Use this only on platforms where you know these features will not be used.

6.7.21.4 EXCLUDE_RAMSTAGE

Prompt: Exclude the ramstage from the coreboot build

Normally, you will want to include the ramstage. If enabled, this option will keep ramstage from being built. This is useful only if you have a custom ramstage.

XPedite8150 User's Manual

Rev. C

X-ES Competition Sensitive - Do Not Share

Extreme Engineering Solutions, Inc.
3225 Deming Way, Suite 120 • Middleton, WI 53562
Phone: 608.833.1155 • Fax: 608.827.6171
sales@xes-inc.com • http://www.xes-inc.com