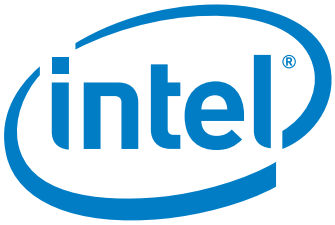


Intel® Atom™ Processor C2000 Product Family for Communications Infrastructure Software for Linux*

Getting Started Guide

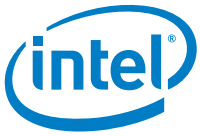
March 2016

Document Number: 333035-002

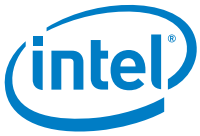
Revision History

Date	Revision	Description
March 2016	002	Updated for software package version 2.6.0; changes include: • Removed Section 8.1, “Setting a Grub Passsword”
October 2015	001	Document declassified for general public. Updated for software package version 2.5.0; changes include: • Updated Section 1.2, “Where to Find Current Software and Documentation” on page 9, and Section 6.1, “Building the Yocto SDK Image” on page 43
March 2015	1.3	Updated for software package version 2.3.0; changes include: • Added Section 1.5.3, “Package Release Structure” on page 11 • Updated Section 9.2, “Installation Script” on page 59 • Updated Section 9.2.2, “Command Line Arguments” on page 60, and Section 9.3.1, “Installing the Acceleration Software” on page 61 • Added Section 9.7, “Minimizing Acceleration Software Compilation Time” on page 64 • Updated Section 10.2.1, “Compiling the Acceleration Functional Sample Code” on page 72
January 2015	1.2	Updated for software package version 2.2.0; changes include: • Removed information regarding re-building the Linux kernel; the re-building process is no longer needed. • Updated Section 2.0, “Deciding your Development Path” on page 13, Section 6.0, “Installing, Building, and Running Yocto*” on page 43, and Section 10.0, “Running Sample Applications” on page 67.
February 2014	1.1	Updated for software package version 1.1; changes include: • Multiple updates to steps in Section 6.0, “Installing, Building, and Running Yocto*” on page 43. • Added Section 9.4, “Cross-Compilation Capabilities for Intel® QuickAssist Technology” on page 62. • Added a note to step 1 in Section 10.1.1, “Compiling the Acceleration Sample Code” on page 67.
November 2013	1.01	Updates include • Updated Chapter 6.0, “Installing, Building, and Running Yocto*”, plus added Section 6.4, “Running the SDK Image” on page 48.
October 2013	1.0	Updates include: • Updated Section 6.0, “Installing, Building, and Running Yocto*” on page 43. • Removed reference to the standalone GbE driver (document number 525876); the GbE driver is now available in the networking software drivers package (document number 537330). • Updated product branding.

Contents



Figures



Tables

1.0 Introduction

1.1 About this Manual

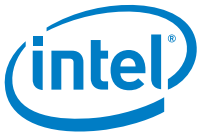
This Getting Started Guide documents the instructions to obtain, build, install and exercise the Intel® Atom™ Processor C2000 Product Family for Communications Infrastructure Software for Linux* Software package 002. Additionally, this document includes brief instructions on configuring the supported development board(s).

Note: In this document, for convenience:

- *Development board*, *development kit* and *CRB* are used as generic terms for the Mohon Peak Customer Reference Board (CRB). The full name is used where appropriate.
- *Software package* is used as a generic term for the Intel® Atom™ Processor C2000 Product Family for Communications Infrastructure Software for Linux* Software package.
- *Acceleration drivers* is used as a generic term for the software that allows the QuickAssist Software Library APIs to access the Intel® QuickAssist accelerator integrated in the Intel® Atom™ Processor C2000 Product Family for Communications Infrastructure System-on-a-Chip (SoC).

This document contains the following sections:

- Introduction (this section)
- Deciding your Development Path
- Development Board Overview
  - — Chapter 3.0, “System Setup and Installation”
  - — Chapter 4.0, “Customer Reference Board (CRB) Features”
  - — Chapter 5.0, “Board Jumper Selection”
- Yocto*
  - — Chapter 6.0, “Installing, Building, and Running Yocto*”
- Running on Fedora*
  - — Chapter 7.0, “Installing and Setting Up Fedora*”
- Chapter 8.0, “System Security Considerations”
- Chapter 9.0, “Building and Installing the Software”
- Running Sample Applications
  - — Chapter 10.0, “Running Sample Applications”
- Appendix A, “Intel® Atom™ Processor C2000 Product Family BIOS Post/Error Codes”
- Appendix B, “BIOS Update Using an SPI Programmer”
- Appendix C, “BIOS Update Using Bootable Software”



1.2 Where to Find Current Software and Documentation

The software release and associated collateral are made available via two separate websites:

- Intel's Open Source Technology Centre (https://01.org); includes public software and documentation; see Section 1.2.1
- Intel Business Portal (www.ibl.intel.com); includes classified software and documentation; see Section 1.2.2

1.2.1 Accessing Content from the Open Source Technology Centre

Intel® QuickAssist Technology Drivers and Patches

1. In a web browser, go to https://01.org/packet-processing/intel%C2%AE-quickassist-technology-drivers-and-patches.
2. Scroll down to the table labeled "attachments".

Intel® Ethernet Drivers and Utilities

1. In a web browser, go to http://sourceforge.net/projects/e1000/files/igb%20stable/
2. Click on the latest version in "igb stable" table and follow further steps to download igb driver.

Table 1 shows the items made available via 01.org.

Table 1. Software and Collateral Available via 01.org

Title	Number
Intel® QuickAssist Technology Software Release Notes	330683
Intel® Atom™ Processor C2000 Product Family for Communications Infrastructure Software for Linux* Getting Started Guide (this document)	333035
Intel® QuickAssist Technology API Programmer's Guide	330684
Intel® Atom™ Processor C2000 Product Family for Communications Infrastructure Software Programmer's Guide	333036
Intel® QuickAssist Technology Cryptographic API Reference Manual	330685
libcrypto* (OpenSSL*) Sample Patch for Intel® QuickAssist Technology	477629

1.2.2 Accessing Additional Content from the Intel Business Portal

1. In a web browser, go to www.ibl.intel.com.
2. Enter your login ID in the **Login ID** box. Check **Remember my login ID** only if you are not using a shared computer. Click **Submit**.

   Note: To acquire a new Intel Business Portal account, please contact your Intel Field Sales Representative.
3. Enter your password in the **Password** box. Click **Submit**.
4. Within the Design Kit Categories, under the **Platform & Solutions** heading, click **Embedded**.

   Under the **Performance Platforms** heading, click **Embedded Platform Code Named Rangeley for Communications and Embedded Applications**.

   a. For the Intel® QuickAssist Technology software, under the Associated Collateral Lists heading, click **Embedded Platforms: Rangeley Platform for Embedded Applications - Software**. This collateral list contains product documentation and software listed in Table 2.

b. For Intel® Atom™ Processor platform information, under the **Associated Collateral Lists** heading, click **Embedded Platforms: Rangeley/Avoton Platform - Platform**. This collateral list contains the product documentation listed in Table 3.

c. For the Edisonville collateral, under the **Associated Collateral Lists** heading, click **Microserver Platform Code Named Edisonville - BIOS, Software and Tools**. This collateral list contains the product documentation listed in Table 4.

Table 2. Software and Collateral Available via the Intel Business Portal

Title	Number
Yocto BSP (meta-intel tarball)	526174
Intel® QuickAssist Technology Acceleration Software OS Porting Guide	446467
Linux* Kernel Cryptographic Framework Sample Driver for Intel® QuickAssist Technology	499692

Table 3. Customer Reference Board Documentation

Title	Number
Mohon Peak Customer Reference Board User's Guide	514980

Table 4. Microserver Platform Code Named Edisonville - BIOS, Software and Tools Documentation

Title	Number
Mohon Peak CRB (Customer Reference Board) - BIOS Images	518736
Avoton/Rangeley SoC Linux Kernel Patch for ES0 and ES1 Silicon - Technical Advisory	518578
Intel® Atom C2000 GbE eeproms	516867
Intel® Network Connections Tools, PV – LAN Software Tools	348742

1.2.3 Pre-boot Firmware

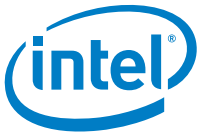
The latest release of the development board pre-boot firmware (BIOS) is also located on the Intel Business Portal.

Please see the Intel® QuickAssist Technology Software Release Notes (document 330683) for the correct firmware version.

1.3 Related Software and Documentation

Refer to the Mohon Peak Customer Reference Board User's Guide (514980) for additional information on the development board including board layout, components, connectors, jumpers, headers, power and environmental requirements, and pre-boot firmware.

Please follow the directions in Section 1.2 to locate this collateral.



1.4 Conventions

The following conventions are used in this manual:

- `Courier font` - code examples, command line entries, API names, parameters, filenames, directory paths, and executables
- **Bold** text - graphical user interface entries and buttons

1.5 Software Overview

1.5.1 Features Implemented

The software provides the following features:

- Access to the cryptographic services in the Intel® QuickAssist Technology hardware
- Sample cryptographic applications
- Intel® QuickAssist Technology Data Plane APIs (`cpa_cy_sym_dp.h`)

Note: For feature details and limitations, if any, refer to the Release Notes.

1.5.2 List of Files in Release

The Bill of Materials (BOM) is included as a text file in the released software package.

1.5.3 Package Release Structure

After unpacking the tar file, the directory should contain:

Files / Directory	Comments
./QATmux.L.a.b.c-n.tar.gz	Top-level QAT package
./filelist	List of files in this package
./installer.sh	Installer script
./LICENSE.GPL	Licence file
./Versionfile	Version file
./QAT1.5	Directory containing QAT1.5.L.a.b.c-n.tar.gz
./QAT1.6	Directory containing QAT1.6.L.a.b.c-n.tar.gz

Depending on the devices detected on the platform, the installer script will build the appropriate driver(s) and, if more than one driver is then built, it will also build the qat_mux. The Intel® Atom™ Processor C2000 Product Family for Communications Infrastructure is supported by the QAT1.5 driver.

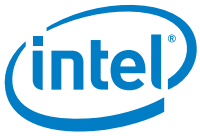
The following extra files/directories may be present.

In the top-level folder:

Files / Directory	Comments
./cpa_mux	Directory containing qat_mux software. This will only be present if both drivers are built.
./InstallerLog.txt	Log of installer script output

In one or both of the QAT1.x directories:

Files / Directory	Comments
./QAT1.x/QAT1.x.L.a.b.c-n.tar.gz	QAT1.x driver package
./QAT1.x/filelist	List of files in the QAT1.x package
./QAT1.x/build	Build output directory
./QAT1.x/versionfile	Version of the QAT1.x driver
./QAT1.x/quickassist	Top-level acceleration software directory for QAT1.x driver
./QAT1.x/quickassist/adf	Acceleration Device Framework source files
./QAT1.x/quickassist/build_system	Build system files
./QAT1.x/quickassist/config	Configuration files
./QAT1.x/quickassist/include	Common acceleration software header files
./QAT1.x/quickassist/lookaside	Service Access Layer source files
./QAT1.x/quickassist/utilities	OSAL and Firmware loading source files



2.0 Deciding your Development Path

This document details several ways to get up and running with Linux and the Intel® Atom™ Processor C2000 Product Family for Communications Infrastructure Software for Linux* Software. Select one of the following options:

- Run using Wind River Linux
  See Section 2.1 for an overview of this development path.
- Install and run on Fedora 17
  Use this option to evaluate the software using Fedora 17.
- Build a customized SDK image; use this option to build your own Linux image using Yocto. This requires knowledge of the Yocto build process.

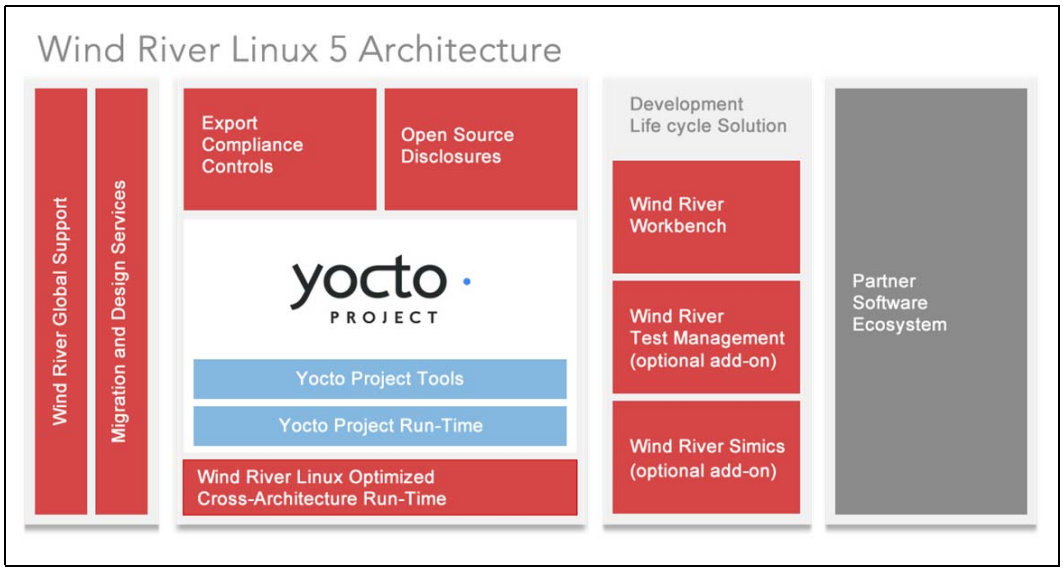
2.1 Wind River Solutions for Mohon Peak CRB

2.1.1 Wind River Linux 5

Platform agility, reduced development costs, and increased capability are important differentiators for Wind River, and we know they are critical factors for your project as well. High-performance, flexible Wind River Linux 5 delivers the power of the industry-standard Yocto Project* infrastructure with better interoperability across many popular platforms. With Wind River Linux 5 as your embedded platform foundation, our extensive quality assurance process and unrivaled global support and services ensure your project's rapid delivery and market success.

With the adoption of the Yocto Project* as its core foundation, Wind River® Linux 5 is the evolutionary next step for open source platforms, elevating the value of our popular embedded Linux platform. We now deliver a hardened embedded Linux distribution with world-class Linux support and differentiation at even faster rates for a lower cost and with significantly higher value. Our certified support and maintenance practices, advanced tools, and rich partner ecosystem will jump-start your development, raise software quality, and reduce the total cost of ownership for your embedded Linux-based next-generation devices.

Figure 1. Wind River Linux 5 Architecture



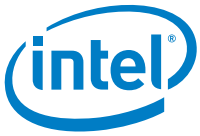
2.1.2 Yocto Project Compatibility

The Yocto Project compliance program is meant to strengthen the project's strategic initiatives, and as a leading Yocto Project participant, Wind River has built Wind River Linux 5 to be Yocto Project Compatible-enabling developers to increase cross-architecture portability, improve software and middleware interoperability, and reduce the cost of change for their embedded Linux platforms.

Table 5. Yocto Project Compatibility Matrix

FEATURES	YOCTO PROJECT	WIND RIVER LINUX
Kernel, toolchain, core user space packages, build system	√	√ Hardened
Commercial quality assurance (QA) across entire feature/ architecture matrix over a large number of development hosts	None	√
Long-term customer support and maintenance, including SLAs	None	Monthly
Regular cumulative service packs that include response fixes to security alerts	Every six months	Monthly
Commercial IP review and license disclosure documentation	None	√
Advanced simulation, compilation, debugging, visualization, and analysis tools	None	√

Prebuilt Wind River Linux 5 images, and tutorials for the Mohon Peak CRB can be downloaded from Intel Business Link.



2.1.3 Wind River Intelligent Network Platform

Wind River® Intelligent Network Platform enables equipment providers to build high performance, high value products that accelerate, analyze, and secure network traffic and applications. The platform is an integrated and optimized software system consisting of the critical run-time components and life cycle development tools needed to build intelligent network elements. Experience the performance gains:

- Up to 1,100% faster IP forwarding than native Linux
- Up to 500% faster UDP or TCP termination than native Linux
- Up to 28x reduced latency for deep packet inspection pattern matching

Wind River Intelligent Network Platform is architected to give our customers maximum scalability and flexibility to build faster, smarter, and more secure products. The platform integrates the key software components to design high-performance, intelligent network applications in a consolidated management and data plane system. Customers can utilize the entire platform or, in some cases, interchange and extend individual components with existing in-house technologies. The platform offers the following:

- Wind River Application Acceleration Engine for ultra-fast layer 3 and layer 4 network protocols
  http://www.windriver.com/products/platforms/intelligent-network/application-acceleration-engine/
- Wind River Content Inspection Engine for high-speed software pattern matching
  http://www.windriver.com/products/platforms/intelligent-network/content-inspection-engine/
- Integrated Intel® Data Plane Development Kit library
  http://www.windriver.com/announces/intel-data-plane-development-kit/

As advanced multi-core technologies proliferate throughout the network, true performance capabilities can only be unlocked by optimizing intelligent software for the latest hardware platforms. Wind River Intelligent Network Platform takes full advantage of Intel multi-core communication platforms. Performance gains include the following:

- Significant improvements for IP forwarding, UDP, and TCP
- Reduced pattern-matching latency
- Substantially reduced data plane network application latency

Software is at the forefront of defining the next-generation network. Not only can it significantly improve network performance but it can also provide the built-in intelligence necessary to deliver new services, greater security, and even more value than ever before. Wind River Intelligent Network Platform provides the following:

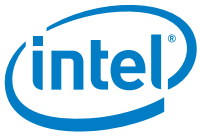
- Deep packet inspection for pattern-matching use cases such as intrusion prevention
- Smart acceleration of security protocols for use in secure clouds and VPNs
- Cost savings through system consolidation

Wind River has done the heavy lifting for you. Wind River Intelligent Network Platform includes all the components necessary to build an ultra-fast, highly intelligent network product-and all the software necessary to consolidate management plane and data

plane applications into one system, saving capital expenditures. The platform can save development teams several years of engineering effort to build, test, and engineer a similar solution with the following features:

- Comprehensive management plane and data plane software system, including Linux, Intel® DPDK, layer 3 and 4 accelerated network protocols, and pattern matching libraries, with extensibility for other DPI engines
- Performance-optimized network acceleration functionality pre-integrated with Intel communications platform
- Consolidation of multiple network applications into one system

Prebuilt Wind River Intelligent Network Platform binaries, use cases, and tutorials for the Mohon Peak CRB can be downloaded from Intel Business Link.



Part 1: Development Board Overview

This section contains the following chapters from the Mohon Peak CRB User's Guide:

- System Setup and Installation
- Customer Reference Board (CRB) Features
- Board Jumper Selection

The following appendixes from the Mohon Peak CRB User's Guide are also included in this document:

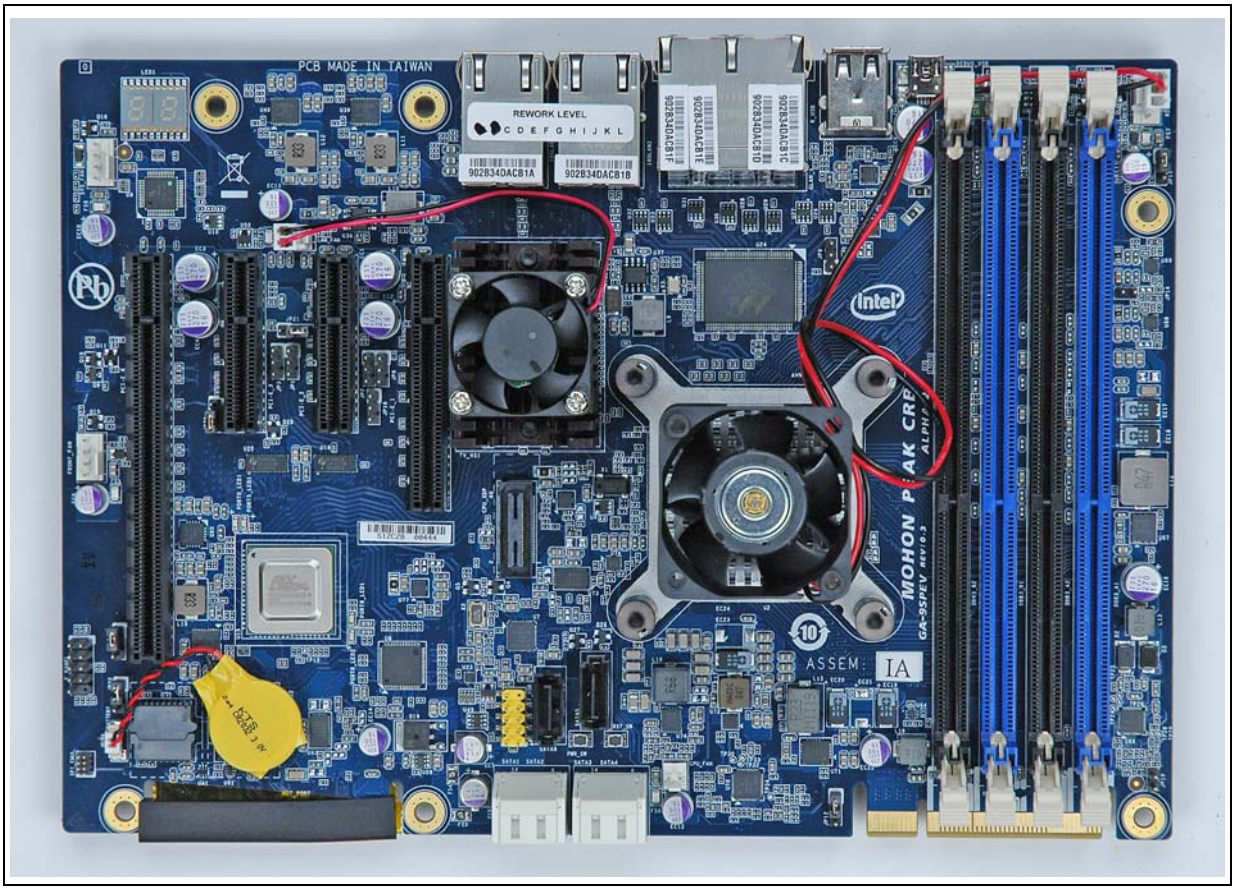
- Appendix A, "Intel® Atom™ Processor C2000 Product Family BIOS Post/Error Codes"
- Appendix B, "BIOS Update Using an SPI Programmer"
- Appendix C, "BIOS Update Using Bootable Software"

3.0 System Setup and Installation

3.1 Power-On Kit Part Inventory

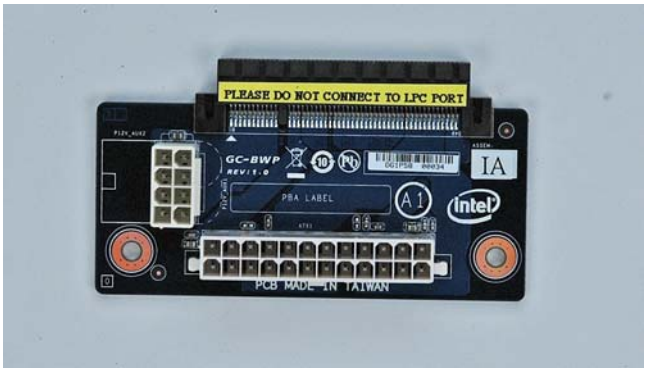
3.1.1 Customer Reference Board (CRB)

Figure 2. Customer Reference Board



3.1.1.1 ATX Power Connector

Figure 3. ATX Power Connector



3.1.2 Chassis

The provided Mohon Peak chassis.

Figure 4. Chassis

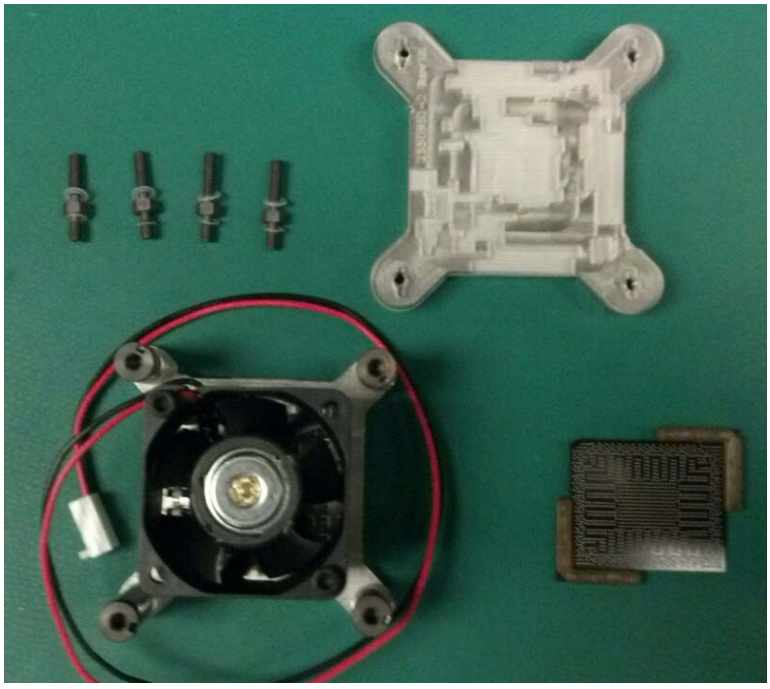
3.1.3 Memory

Two 2 GB UDIMMs will be included with every Mohon Peak Power-On Kit.

Figure 5. 2 x 2 GB UDIMMs

3.1.4 MPI Socket

Figure 6. MPI Socket



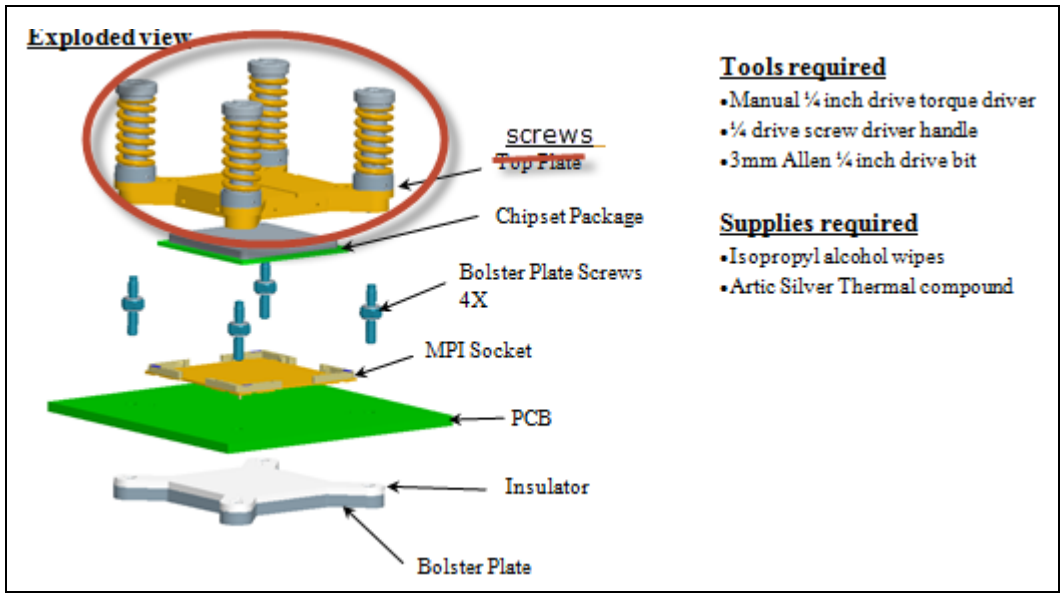
3.2 Initial Board Setup

3.2.1 Building of MPI Socket

Proper installation of the MPI socket and SoC processor are very important for the successful setup or upgrade of the Mohon Peak CRB. See step-by-step instructions below.

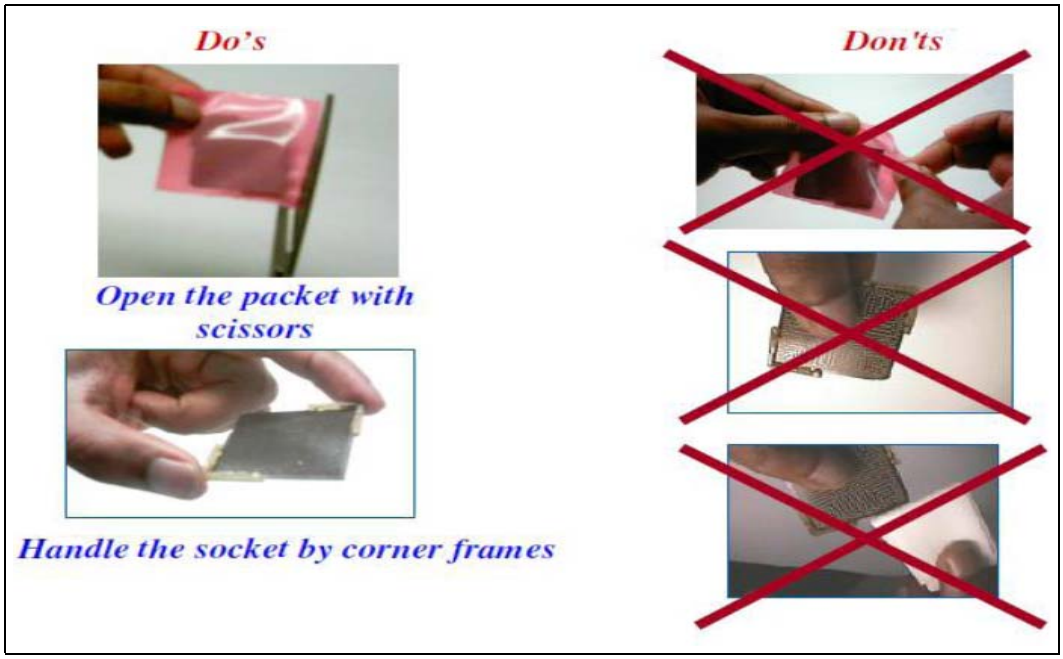
3.2.1.1 Tools and Supplies Required

Figure 7. Tools and Supplies Required



3.2.1.2 MPI Socket handling Instructions

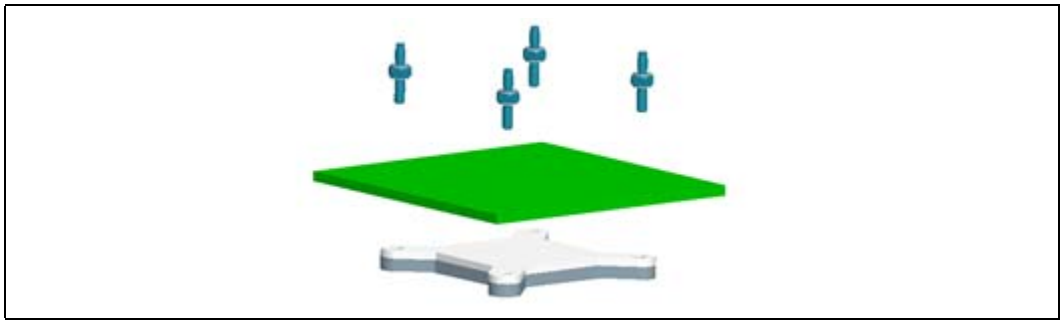
Figure 8. MPI Socket handling instructions



3.2.1.3 MPI Socket Installation

1. Assemble the C2xx0 or C2xx8 Bolster Plate on the secondary side of the PCB using four bolster plate screws. Align the pin on indicator on the bolster plate with pin one on the PCB. Torque screws to 10 in-lbs or tighten 1/8 to ¼ turn past screw seating on PCB. Ensure that each screw has plastic washer on each side of the hex nut.

Figure 9. Bolster Plate Assembly



2. Clean LGA pads on the PCB with Isopropyl Alcohol (IPA) wipes. Let dry 30 seconds to evaporate all of the IPA.
3. Clean LGA pads on the CPU package with Isopropyl alcohol (IPA) wipes. Let dry 30 seconds to evaporate all of the IPA.

4. Place the C2xx0 or C2xx8 MPI socket on the PCB with pin one in the proper location and with the alignment pins in the alignment holes.

Figure 10. PCB Diagram

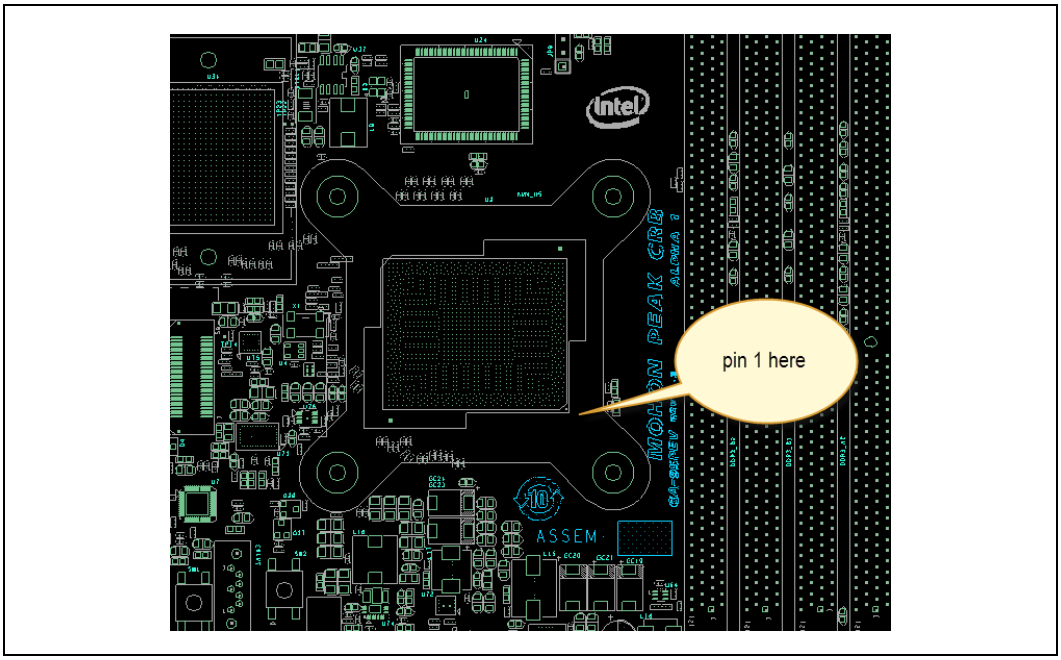
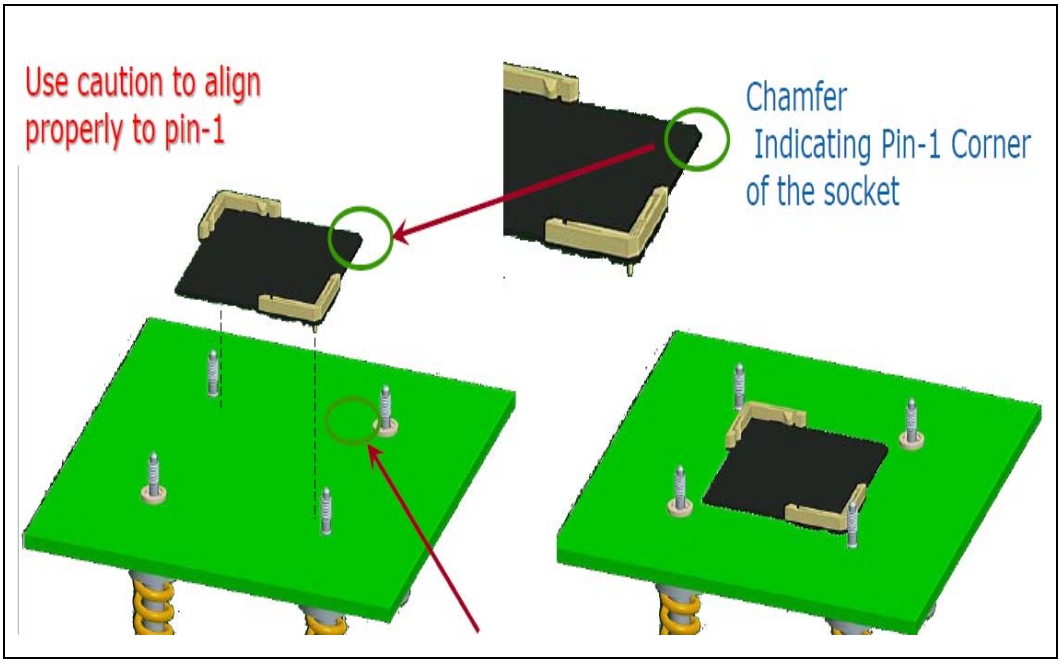
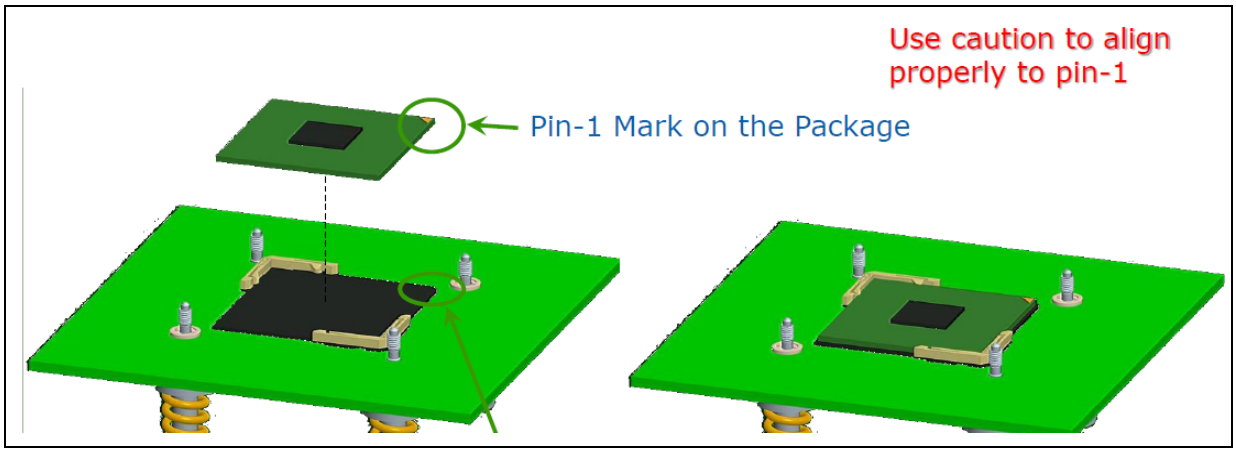


Figure 11. Pin-1 Alignment Diagrams



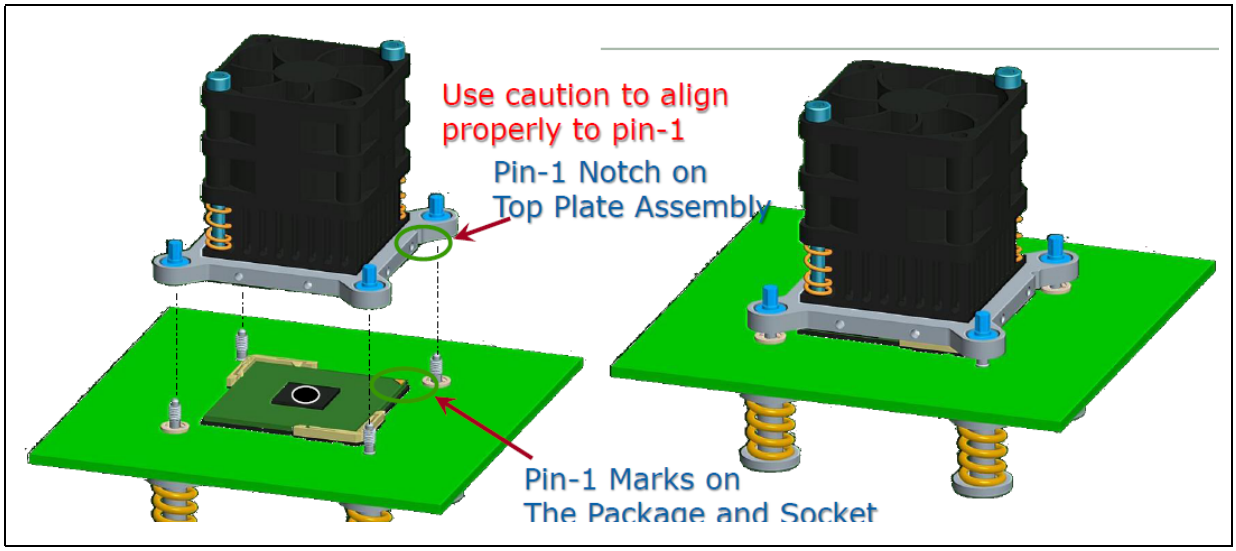
5. Processor Package Placement
   a. Place the C2xx0 or C2xx8 package into the socket.
   b. Gently press on the top of the package until the component snaps into place.
   c. Apply about one drop of the Arctic Silver thermal compound to the top center of the die.

Figure 12. Processor Package Placement Diagrams



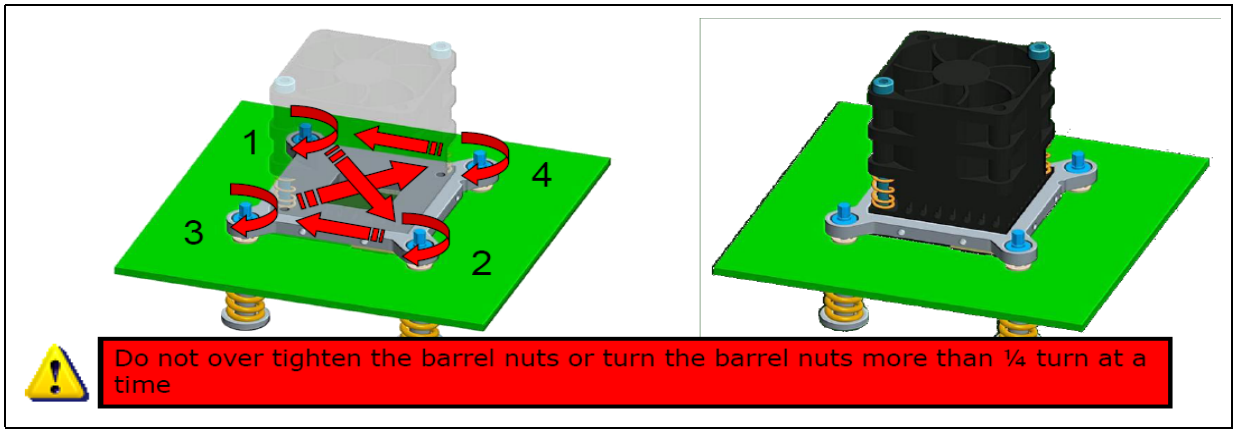
6. Top Plate and CPU Fan Placement:
   a. Place the top Plate Assembly on top of the Package with the notch aligned with pin1.
   b. Lightly tighten each of the Barrel nuts until each Barrel nut has engaged with the plungers.

Figure 13. Top Plate and CPU Fan Placement Diagrams



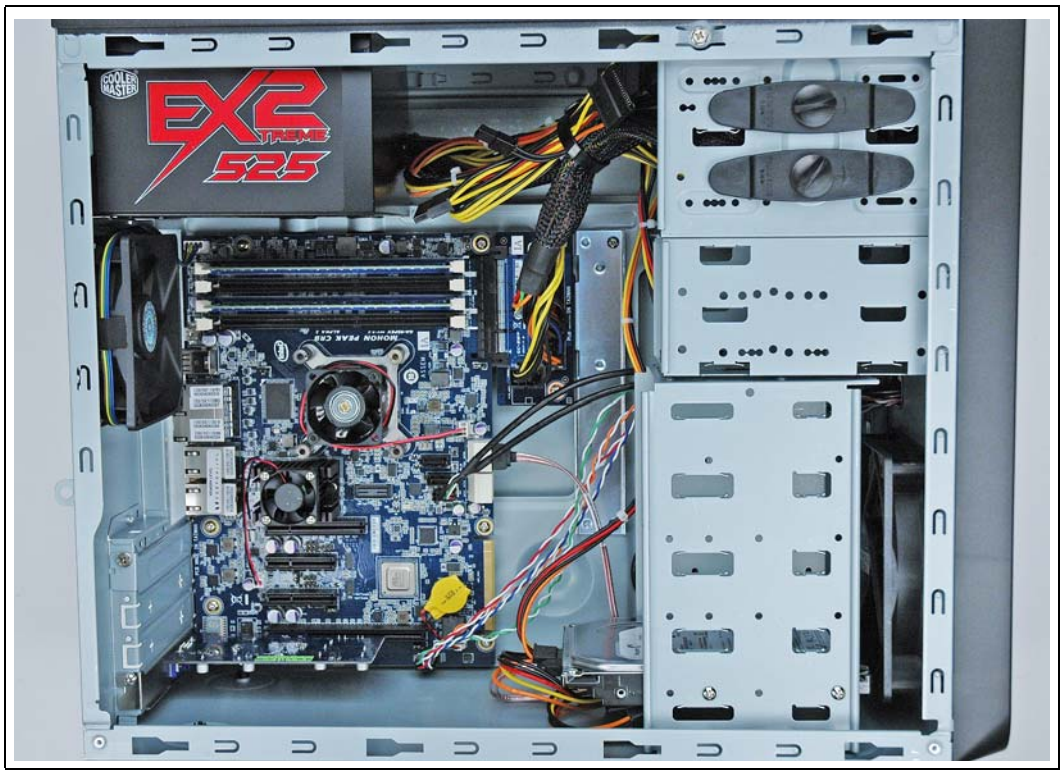
7. Seating the Spring Assemblies
   a. Tighten the Barrel nuts in an X pattern as shown below ¼ turn each.
   b. Repeat the ¼ turn tightening sequence until the Barrel nuts seat on the plunger top.
   c. To verify that the Barrel nuts are seated properly, torque the assemblies to 3.5 to 4 in-lbs of torque.

Figure 14. Seating Spring Assemblies



3.2.2 Placement of the Board in the Chassis

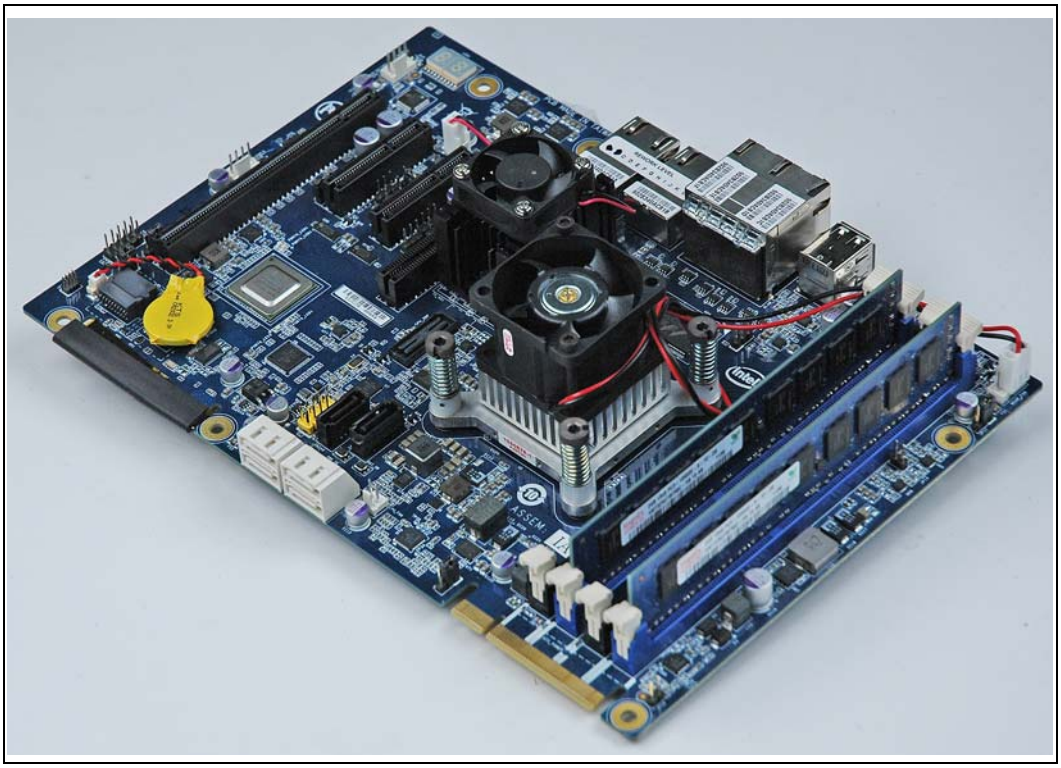
Figure 15. Placement of Board in Chassis



3.2.3 Proper UDIMM Placement

The UDIMM installation on the board is important for the BIOS and OS recognition of populated UDIMMs and proper functioning of the memory on the system. The first two UDIMMs to be installed on the CRB must properly terminate each memory channel on the CRB. Both blue DIMM slots are located at the end of each memory channel and must be populated first before the other two slots can be used.

Figure 16. Proper Placement for UDIMM Socket



3.2.4 Correct Connection of ATX Power Connector

The correct installation of the ATX Power connector is on the **bottom right-side** of the CRB. Do NOT connect the power connector to any other port on the board. Any other application of the power connector will OVER-LOAD the signal circuits permanently disabling the CRB. The board will be unusable in the future.

Figure 17. Connection of Power Pins to ATX Power Connector

3.3 BIOS Flash Update Process

Updating the BIOS firmware on the Mohon Peak CRB requires reprogramming the Flash memory. There are several methods of accomplishing this including: use bootable software to erase and reprogram, use an in-circuit SPI bus Flash programmer, or remove the socketed Flash component and reprogram it in an external (stand alone) programmer.

A specific example of programming with an SPI bus Flash programmer is presented in Appendix B, “BIOS Update Using an SPI Programmer.”

A specific example of programming with bootable software is presented in Appendix C, “BIOS Update Using Bootable Software.”

3.4 Network EEPROM Update

If it is found necessary to modify the factory programmed configuration of the network ports on the Mohon Peak board, the EEPROM contents will need to be updated. Intel will provide the upgraded code along with the appropriate software tool and instructions at that time. Be aware that not following the provided instructions very carefully may render one or more of the network ports on the board inoperable, and may require return of the CRB to Intel for repair.

Currently, the software tool used is called "eeupdate64E.EFI" and the process is as follows (this may change in the future, please see the instructions that come with the update before performing this operation):

1. Copy the "eeupdate64E.EFI" program to a USB Flash drive
2. Copy the Intel provided EEPROM data file to the USB Flash drive
3. Power the Mohon Peak system on and stop in the EFI Shell
4. Plug the USB Flash drive into an available USB port
5. At the "Shell" prompt, type the following series of commands:
   > map -r
   > fs0:
   > eeupdate64E.EFI

This should produce a eeupdate sample output that looks like:

NIC	Bus	Dev	Fun	Vendor-Device	Branding string
1	7	00	00	8086-1528	Intel(R) Ethernet Controller X540-AT2
2	7	00	01	8086-1528	Intel(R) Ethernet Controller X540-AT2
3	0	20	00	8086-1F41	Intel(R) Avoton SGMII
4	0	20	01	8086-1F41	Intel(R) Avoton SGMII
5	0	20	02	8086-1F41	Intel(R) Avoton SGMII
6	0	20	03	8086-1F41	Intel(R) Avoton SGMII

Only update the EEPROM for the SGMII ports. Performing the EEUPDATE function on the X540-AT2 ports will cause them to no longer function. In the example above, the NICs to update are 3 through 6.

For the purposes of this example the EEPROM data is in a file called "eeprom.hex". Enter the following commands to update all four "SGMII" ports:

> eeupdate64E.EFI /NIC=3 /DATA eeprom.hex
> eeupdate64E.EFI /NIC=4 /DATA eeprom.hex
> eeupdate64E.EFI /NIC=5 /DATA eeprom.hex
> eeupdate64E.EFI /NIC=6 /DATA eeprom.hex

Once the steps above are complete, power the Mohon Peak system off, and remove the USB Flash drive. The system is now up to date and ready to use.

3.5 Clearing CMOS

Clearing the CMOS resets only the variables saved in the RTC registers. To clear the CMOS on a Mohon Peak CRB, the board must be powered off completely. Once the machine is off, move the JP12 jumper *(see Figure 27, "Mohon Peak Jumper Selection Diagram (Lower Right Corner))" on page 41)* from the 1-2 position (default) to the 2-3 position (clearing CMOS). Leave the jumper in the clearing CMOS position for two seconds, then return it to the default position. The CMOS should now be clear and the CRB can now be booted with default firmware settings.

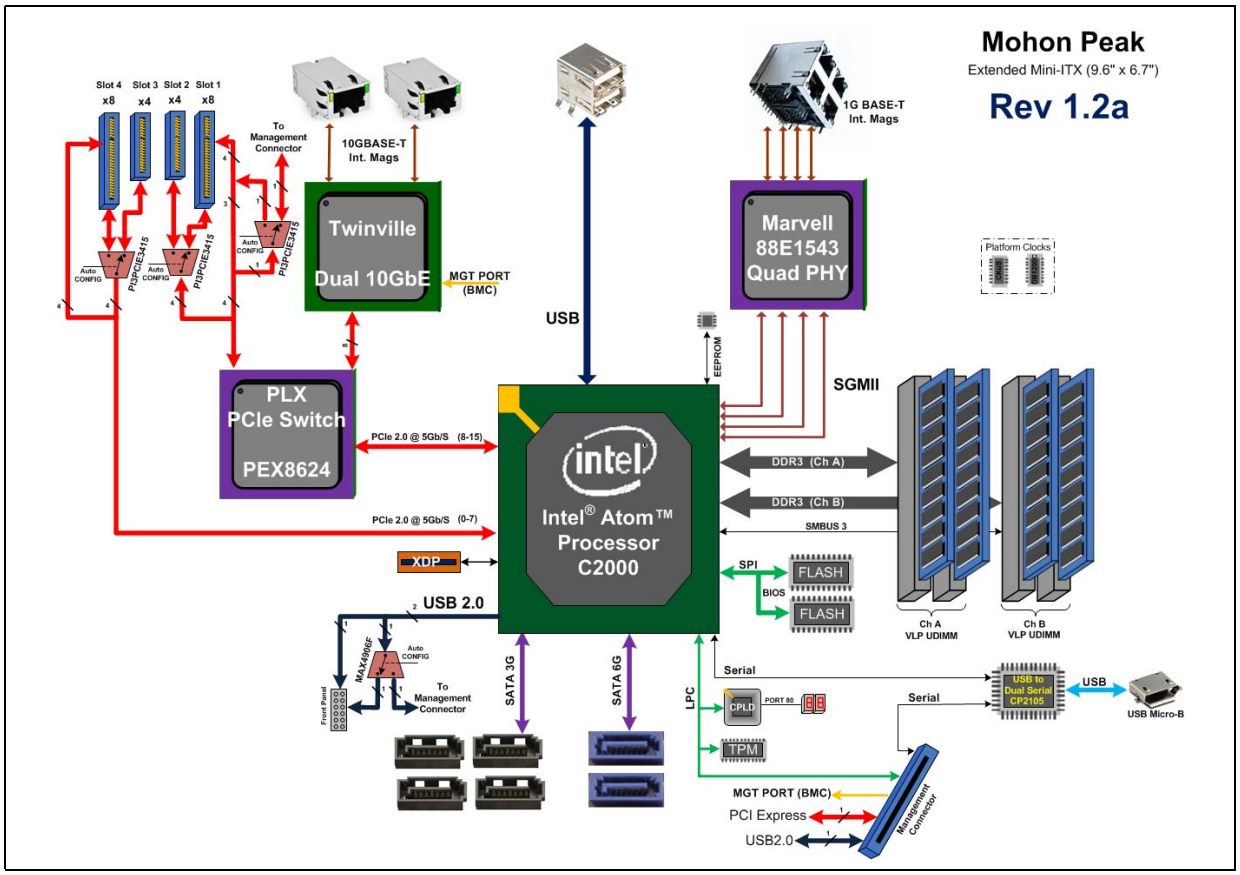
§ §

4.0 Customer Reference Board (CRB) Features

4.1 System Architecture Block Diagrams

This document is intended as a guide for users of the Mohon Peak customer reference board (CRB). The Mohon Peak CRB is intended for customer reference and validation of the Intel® Atom™ Processor C2000 Product Family for Microserver or the Intel® Atom™ Processor C2000 Product Family for Communications Infrastructure System-on-a-Chip (SoC). A high-level block diagram for the Mohon Peak System Board (MPK) is shown in Figure 18.

Figure 18. Mohon Peak Block Diagram



4.2 Hardware Components

This section briefly describes main hardware components and important design details.

Figure 19. Mohon Peak CRB Component Diagram

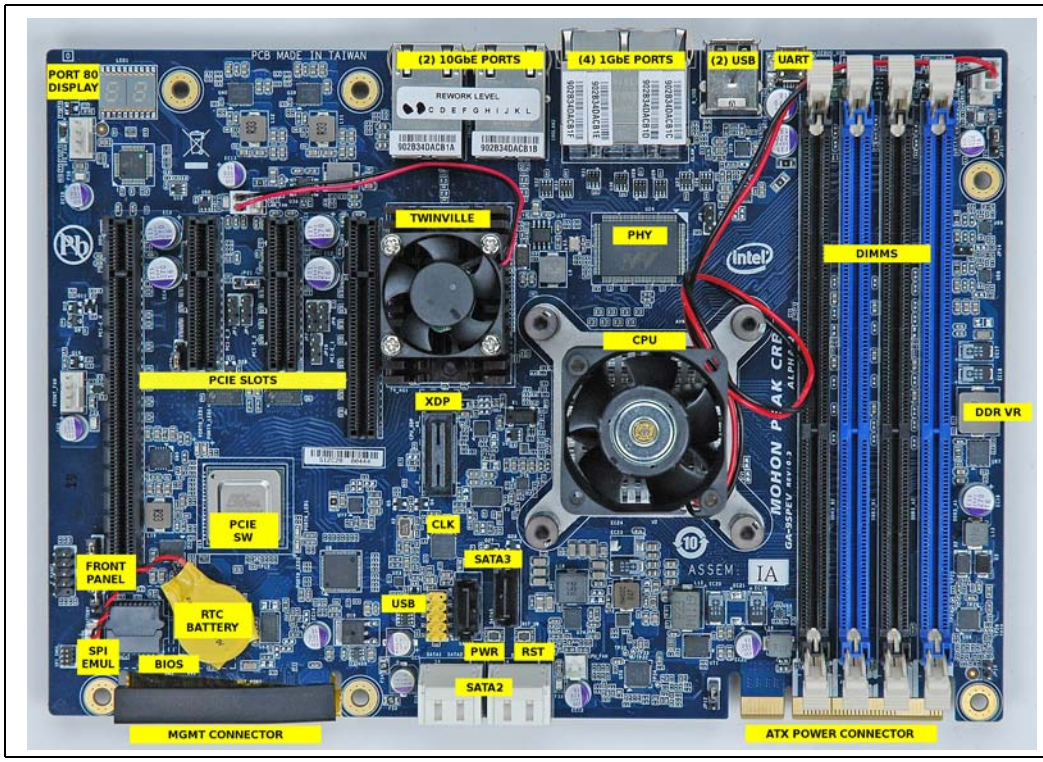
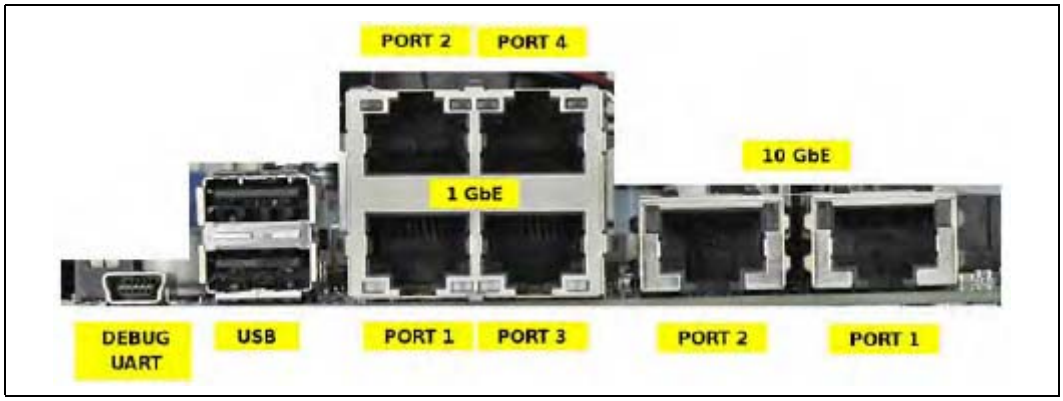


Figure 20. Mohon Peak Rear Panel Connectors



4.2.1 CPU

The SoC is based on the Silvermont, two-way superscalar, out-of-order IA core. The Silvermont is a 22 nm core used for ultra-low-power applications. The SoC can contain up to 8 Silvermont cores; supporting SKUs with 2, 4, or 8 cores. The processor supports the following CPU features:

- VT-x2 (virtualization)
- Turbo Boost
- Various C-States (C0 through C6)

The processor does not support Hyper-Threading (HT).

4.2.2 SoC SATA Controllers

The SoC has two independent integrated SATA host controllers. The SATA2 controller supports Direct Memory Access (DMA) operation up to four ports and supports data transfer rates of 3.0 Gb/s (300 MB/s) and 1.5 Gb/s (150 MB/s). The SATA3 controller in addition to legacy data rates supports data rates of up to 6 Gb/s (600 MB/s). The SATA3 controller is the legacy IDE controller and has two ports. Both SATA controllers contain two modes of operation; a native mode and an Advanced Host Controller Interface (AHCI) mode using memory space. Software that uses legacy mode does not have AHCI capabilities.

4.2.3 Memory

4.2.3.1 DRAM Memory

The Mohon Peak board provides four DDR3 DIMM sockets for a dual channel memory system (two UDIMMs per channel). The characteristics of the supported memory include:

- DDR3 (1.5V) or DDR3L (1.35V) UDIMMs
- ECC is supported
- 1066/1333/1600 MT/s data rates
- Single or double rank x8 data width (x16 not supported)
- 1/2/4/8/16 GB UDIMM sizes

The blue sockets (DDR3_A1 and DDR3_B1) should always be populated before the black sockets (DDR3_A2 and DDR3_B2).

The first memory channel is designated as DDR3_A* (one blue, one black socket), and the second channel is DDR3_B* (one blue, one black socket). When populating memory in both channels, channel A and channel B must be populated identically.

Table 6. Supported DDR3 UDIMM Timings Chart

Supported Data Rate	Supported Memory Timing			
1066 MT/s	6-6-6	7-7-7	8-8-8	
1333 MT/s	7-7-7	8-8-8	9-9-9	10-10-10
1600 MT/s	8-8-8	9-9-9	10-10-10	11-11-11

4.2.3.2 BIOS Flash

The system BIOS firmware is stored on board in Flash memory. Two STMicroelectronics M25P64-VMF6P Flash devices (8 MB each) are provided:

- BIOS1 (socketed)
- BIOS2 (soldered onto the board)

Two jumpers control the functioning of the BIOS Flash devices:

- JP4 selects the active device (BIOS1 or BIOS2)
- JP22 enables the SPI_EMUL header exclusive access to the selected Flash device for programming via the SPI bus.

Note: See Figure 19, “Mohon Peak CRB Component Diagram” on page 32.

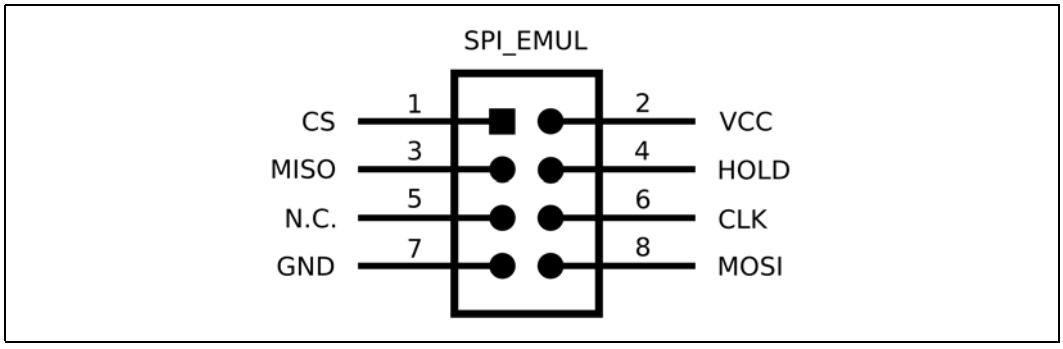
The SPI_EMUL header may also be used to connect an external ROM emulator if desired. In this case, BIOS1 must be selected for normal operation, and the BIOS1 Flash device must be removed from its socket. Return the Flash device to its socket once finished using the emulator. Make sure to align pin one of the Flash device with pin one of the socket (designated by a white triangle on the board next to the socket).

Table 7. BIOS Flash Configuration Chart

Function	Flash	JP4	JP22	SPI_EMUL Header
Normal Operation	BIOS 1	1-2	1-2	Empty
	BIOS 2	2-3	1-2	
BIOS Update Using Bootable Software	BIOS 1	1-2	1-2	Empty
	BIOS 2	2-3	1-2	
BIOS Update Using SPI Programmer	BIOS 1	1-2	2-3	SPI Programmer
	BIOS 2	2-3	2-3	
ROM Emulator	-	1-2	1-2	ROM Emulator
	(Remove BIOS1 device from the socket)			

Note: See Table 8, “Mohon Peak Jumpers” on page 39.

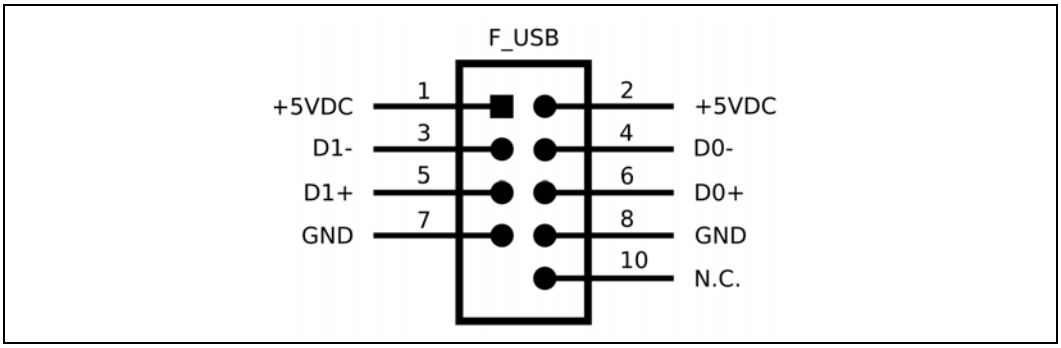
Figure 21. SPI_EMUL Header Pinout



4.2.4 USB

Four USB 2.0 ports are supported on the Mohon Peak CRB. Two route to the back facing panel and two route either to a front-panel connector or to a MUXed Baseboard Management Connection (BMC). The F_USB front panel header is located adjacent to the SATA connectors, denoted by a nine pin header with a yellow base (see Figure 19, "Mohon Peak CRB Component Diagram" on page 32). The SoC contains one Enhanced Host Controller Interface (EHCI). It complies to the EHCI 1.0 Specification and supports up to four USB 2.0 root ports. USB 2.0 allows data transfers up to 480 Mbps. The controller integrates a Rate-Matching Hub (RMH) to support USB 1.1 devices.

Figure 22. F_USB Front Panel USB Header Pinout



4.2.5 Storage

The Mohon Peak CRB has a total of six internal SATA ports for connection to storage SATA devices:

- Four SATA2 (3Gb/s) ports
- Two SATA3 (6Gb/s) ports

4.2.6 Ethernet

4.2.6.1 Four 1 GbE BASE-T Ports

The Mohon Peak CRB has four 1 GbE ports using the SoC Network controller. The Marvell* 88E1543 provides the quad-PHY functionality.

4.2.6.2 Two 10 GbE BASE-T Ports

The Mohon Peak CRB has two 10 GbE ports using a Twinville dual network controller.

4.2.7 PCIe Slots

The SoC provides 16 lanes of PCIe* Version 2.0. The PCIe bus slots are routed in two x8 channels:

- Channels 0-7 connect to a PCIe multiplexor (Pericom* P13PCIE3415) that provides routing to either the x16 Slot 4 or shares bandwidth with Slot 3, a x4 slot. The P13PCIE3415 switch can auto-configure to two x4 Channels if both slots are populated or one x8 channel if the only populated slot is the x16.
- Channels 8-15 are routed to a PLX PCIe Switch (PEX8624). From this switch, the channels are bifurcated into eight lanes routed to the Twinville Dual 10 GbE controller and to PCIe x4 Slot 1 and x4 Slot 2. The bandwidth from the eight lanes of PCIe is shared between the PCIe slots and network controller.

4.2.8 ATX Power Connector

The ATX power connector is located on the lower **right-hand** corner of the CRB; located below the UDIMM sockets (see Figure 19, “Mohon Peak CRB Component Diagram” on page 32). An ATX Power connector LPC daughter card connects to this male connection to provide power to the Mohon Peak CRB. **DO NOT** connect the daughter card to the Management connector on the **left side** of the board and power it on. This will permanently disable the board.

4.2.9 Management Connector

The Management connector is at the lower **left-hand** corner of the CRB; located below the dual BIOS Flash chip enclosures. (see Figure 19, “Mohon Peak CRB Component Diagram” on page 32) This connector can be used for creating management connections to the Mohon Peak CRB. This provides a way to monitor SMBUS variables and functions as well as allowing communication with different modules on the board. For more information on the use and function of this connector, please refer to page 53 in the Intel® Atom™ Processor C2000 Product Family - *Mohon Peak A2 Customer Reference Board Schematic,* Revision 1.0 (CDI document number 506415).

4.2.10 Trusted Platform Module

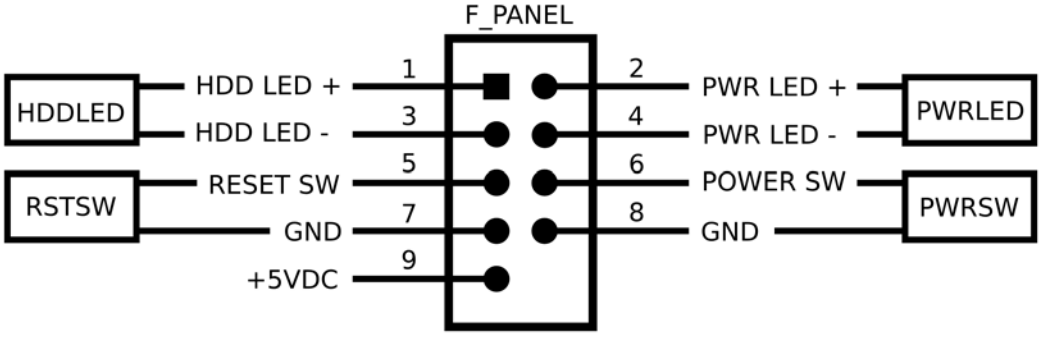
The Trusted Platform Module (TPM) component is connected to the SoC Low Pin Count (LPC) bus providing trusted platform functionality. The component manufacturer/model used on the Mohon Peak CRB is the ST19NP18ER28PVMK from ST Microelectronics*. The TPM uses embedded TPM 1.2 firmware and has a 33-MHz LPC bus. The TPM is located on the backside of the board near the Management Connector (Figure 23).

Figure 23. Mohon Peak CRB Diagram - Trusted Platform Module Supported on LPC Bus

4.2.11 Front Panel Connector

The Front Panel header allows for connection of chassis LEDs and buttons. This header is labeled "F_PANEL" and is in the lower left corner of the CRB.

Figure 24. F_Panel Header and Connections



4.3 Serial Port Drivers

From a Windows* client machine (for instance, a work laptop), install the serial port drivers, located at the following address:

http://www.silabs.com/Support%20Documents/Software/CP210x_VCP_Windows.zip

1. Save and run the driver install file.

   Follow the on-screen prompts to install the driver.
2. Obtain a mini-USB cable (USB Mini B). Connect the mini-USB end of the cable to the mini-USB connection on the front of the Mohon Peak CRB. Connect the other end of the cable to the Windows client machine.
3. On the Windows client machine, open the Device Manager to find the COM# of the Silicon Labs Dual CP210x USB to UART Bridge: Standard COM Port. This will be visible under **Ports (COM & LPT)**.
4. Download (if necessary) and install a terminal emulator (for example, PuTTY). Select the Serial connection type, update the COM#, and set the speed to 115200. Open the connection.
5. When booting the Mohon Peak CRB, debug messages should be displayed. Some console information will also be redirected over the serial port.

§ §

5.0 Board Jumper Selection

Table 8 shows the Mohon Peak Jumpers.

Table 8. Mohon Peak Jumpers

Jumper	MPK Usage	Options	Default
JP1	HOST SMBus	For Debug	No Jumper
JP2	Sensor SMBus	For Debug	No Jumper
JP3	PECI SMBus	For Debug	No Jumper
JP12	CMOS Clear	[1-2]: Normal Operation [2-3]: CMOS Clear	[1-2]: Normal Operation
SPI_EMUL	BIOS Programming - use for both emulator or SPI onboard programming	BIOS Programming	No Connection
JP4	BIOS Chip Selection	[1-2]: Select BIOS1 chip [2-3]: Select BIOS2 chip	[1-2]: Boot from BIOS1
JP22	BIOS Onboard Programming	[1-2]: Normal Operation [2-3]: BIOS Programming	[1-2]: Normal Operation
JP20	PCIe* configure between Slot 1 and 2	[1-2]: No PCIe card at Slot 2 [2-3]: With a PCIe card at Slot 2	[1-2]: No PCIe card at Slot 2
JP21	PCIe configure between Slot 3 and 4	[1-2]: No PCIe card at Slot 3 [2-3]: With a PCIe card at Slot 3	[1-2]: No PCIe card at Slot 3
JP9	GbE SMBus at PHY	For Debug	No Jumper
JP11	RTC CMOS Clear	[1-2]: Normal Operation [2-3]: RTC Clear CMOS	[1-2]: Normal Operation
JP10	GbE SMBus at CPU	For Debug	No Jumper
F_PANEL1	Front Panel		
FAN_CPU1	System Front Panel		
FAN_CPU2	System Front Panel		
SYS_FAN1	System Side Front Panel		
CPU_FAN1	CPU Fan		Connect CPU Fan
LAN_FAN1	LAN Fan		Connect Twinville Fan
JP14	For PVDDRA Voltage Margin	DO NOT SHORT - For EV only	OPEN: Normal Operation
JP19	For P1v0 Voltage Margin	DO NOT SHORT - For EV only	OPEN: Normal Operation

Figure 25. Mohon Peak Jumper Selection Diagram (Upper Left Corner)

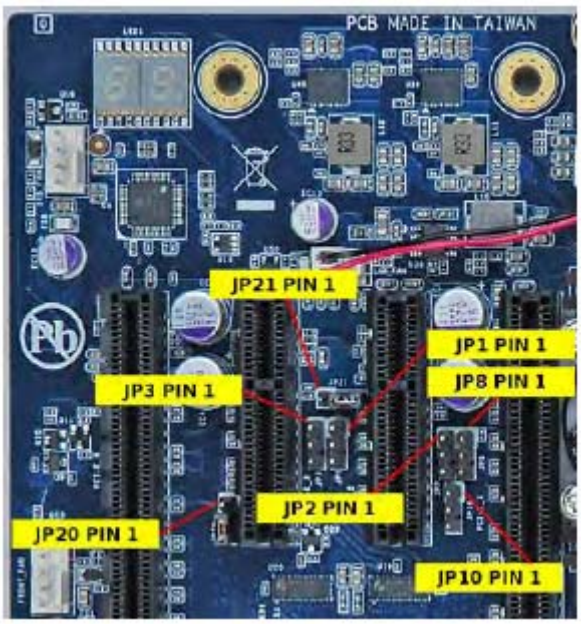


Figure 26. Mohon Peak Jumper Selection Diagram (Lower Left Corner)

Figure 27. Mohon Peak Jumper Selection Diagram (Lower Right Corner))

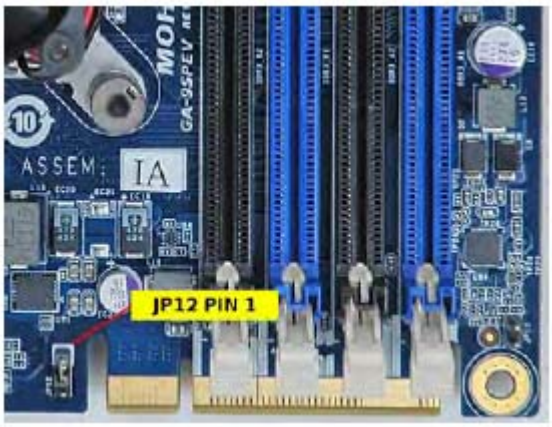
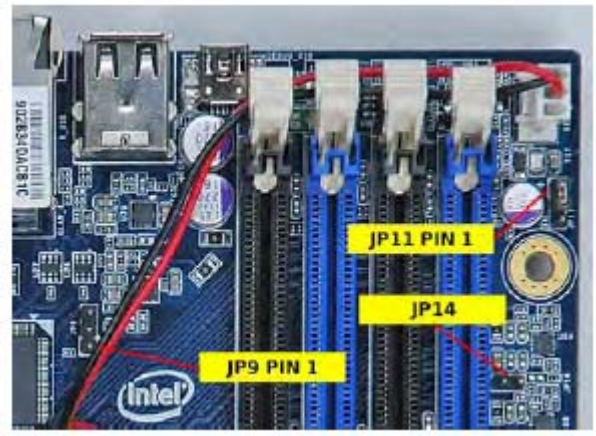


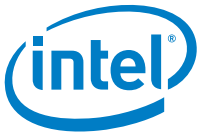
Figure 28. Mohon Peak Jumper Selection Diagram (Upper Right Corner)



§ §

Part 2: Yocto*

This section contains the following chapter:



6.0 Installing, Building, and Running Yocto*

The Yocto Project is an open-source collaboration project focused on embedded Linux. Yocto includes a set of tools to build a custom Linux distribution. The process to create your custom Linux distribution using Yocto involves creating your own image on a software development workstation. The steps in Section 6.1 should be done on a software development workstation, not the Mohon Peak CRB.

The steps to build and copy the image on Ubuntu 12.04 are included here. If you are using a different Linux distribution, consult the Yocto Project website (http://www.yoctoproject.org) for more information and documentation, including:

- Yocto Quick Start Guide:
  http://www.yoctoproject.org/docs/latest/yocto-project-qs/yocto-project-qs.html
- Git repository:
  http://git.yoctoproject.org/cgit/cgit.cgi/meta-intel/tree/meta-crystalforest

Note: In this release, the pre-built Yocto images were removed from the BSP. If you would like the pre-built image, please contact your Intel representative.

Note: The pre-built image has a Time-Limited-Kernel (TLK), which means that the image is restricted to a 10-day uptime and the image will be auto-rebooted after that time. TLK is added to encourage end-users to build their own image for production.

6.1 Building the Yocto SDK Image

Follow the instructions below to create the image and to run user-space and kernel-space examples.

Note: The build process using sato consumes about 50 GB of disk space. Therefore, at least 100 GB of free disk space is recommended.

1. Consult the Intel® Atom™ Processor C2000 Product Family for Communications Infrastructure Software for Linux* Release Notes (document 330683, “Board Support Package” section) to find the applicable package versions for Intel® QuickAssist Technology software, Intel® DPDK, Gigabit Ethernet driver, libcrypto, netkey, and Yocto BSP packages.
2. Install Ubuntu 12.04 (64-bit).

Note: Always execute the following instructions as a non-root user.

3. If required, update the proxy settings for your network environment by adding the following to `/etc/environment`:

```
https_proxy='https://<proxy_server>:<proxy_port>/'
http_proxy='http://<proxy_server>:<proxy_port>/'
ftp_proxy='http://<proxy_server>:<proxy_port>/'
GIT_PROXY_COMMAND=/usr/bin/git_proxy_command
```

4. Configure ssh configure file for proxy settings. Add following lines to ~/.ssh/config:

```
host *
```

```
ProxyCommand connect-proxy -s %h %p
```

5. Create /usr/bin/git_proxy_command file and add the following lines:

```
#!/bin/sh
connect-proxy -s $@
```

6. Change the git_proxy_command file to be executable:

```
sudo chmod +x /usr/bin/git_proxy_command
```

7. If there are script build errors that appear to be syntax errors, it is likely that the script is being passed to the wrong shell. Many of the Yocto scripts call /bin/sh, which may be symbolically linked to /bin/dash. This can be resolved by removing /bin/sh ("sudo rm /bin/sh") and linking to bash instead ("**sudo ln -s /bin/bash /bin/sh**").

8. Log out of the current user account and login, to allow the environment changes to take effect.

9. Update apt-get:

```
# sudo apt-get -y update
```

10. Install the required software components:

```
# sudo apt-get -y install gawk wget git-core diffstat unzip texinfo \
     build-essential chrpath libsdl1.2-dev xterm socat connect-proxy
```

Note: If the apt-get command does not succeed completely, try it again, since the specific mirror selected may not have transferred the files correctly.

11. Download the Yocto BSP and change to the poky directory:

```
# cd ~
# git clone -b dylan http://git.yoctoproject.org/git/poky
# cd poky
```

12. Check out using the following poky/dylan commit ID:

```
# git checkout d734ab491a30078d43dee5440c03acce2d251425
```

13. Extract the meta-intel tarball to the poky directory using the commands:

```
# cd ~/poky
# tar xjvf <path-to>/meta-mohonpeak-<version>.tar.bz2
# cd meta-mohonpeak-<version>
# mv meta-intel ../
```

Note: Obtain the meta-intel tarball from document 526174 listed in Table 2.

14. Verify that the following directory structure is present, at a minimum:

```
pwd: ~/poky
:
|-- bitbake
|
|-- meta-intel
|    |
|    |--meta-mohonpeak
```

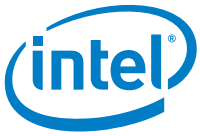
15. Run the script to build the environment essentials:

```
# cd ~/poky
# source oe-init-build-env
```

This will also change the current working directory to ~/poky/build.

Note: Be sure to source the file while in ~/poky since the build directory will be created based on the current working directory.

16. Edit the file ~/poky/build/conf/local.conf. Uncomment the lines with BB_NUMBER_THREADS and PARALLEL_MAKE and change the number to the max logical cores available, if desired for the greatest compilation speed. For instance, on a system with 8 logical cores, the two lines will be as follows:



```
BB_NUMBER_THREADS = "8"
PARALLEL_MAKE = "-j 8"
```

17. Edit the file ~/poky/build/conf/local.conf. Change the uncommented MACHINE line to the correct target. By default, the line is as follows:

    ```
    MACHINE ??= "qemux86"
    ```

    If building for 32-bit, change the line to:

    ```
    MACHINE ?= "mohonpeak32"
    ```

    If building for 64-bit, change the line to:

    ```
    MACHINE ?= "mohonpeak64"
    ```

Note: To add extra packages to the generated image, edit EXTRA_IMAGE_FEATURES variable and add the following package names separated by a space:

```
EXTRA_IMAGE_FEATURES = "debug-tweaks dev-pkgs"
```

Note: To execute a package management tool "smart" on the generated image, edit the PACKAGE_CLASSES variable and add the following:

```
PACKAGE_CLASSES = "package_rpm"
```

18. Optional: Edit the file ~/poky/build/conf/local.conf to enable the DL_DIR variable to a directory path of your choosing. This can shorten the build time when rerunning the bitbake command below.

19. Edit the file ~/poky/build/conf/bblayers.conf to add meta-intel and meta-mohonpeak to the targets. Before the edit, the applicable lines will be the same as or similar to the following:

    ```
    BBLAYERS ?= " \
      /home/user/poky/meta \
      /home/user/poky/meta-yocto \
      /home/user/poky/meta-yocto-bsp \
    ```

    After the edit, the applicable lines should be as follows:

    ```
    BBLAYERS ?= " \
      /home/user/poky/meta \
      /home/user/poky/meta-yocto \
      /home/user/poky/meta-yocto-bsp \
      /home/user/poky/meta-intel \
      /home/user/poky/meta-intel/meta-mohonpeak \
      /home/user/poky/meta-intel/meta-tlk \
    ```

Note: meta-intel tar ball may have pre-configured to build and install Intel® DPDK and Intel® QuickAssist Technology. The SDK image will be used, and other software will need to be installed after booting. Here are the steps to change the configuration:

a. Depending upon the MACHINE selected in step 17 above, find the **mohonpeak32** or **mohonpeak64** configuration file; for example, mohonpeak64 is at meta-intel/meta-mohonpeak/conf/machine/mohonpeak64.conf

b. Open the mohonpeak32 or mohonpeak64 file

c. Locate MACHINE_EXTRA_RECOMMENDS

   For example:

   MACHINE_EXTRA_RRECOMMENDS += "intel-quickassist intel-qatsamples \

   intel-qat-perf intel-qat-netkey \
   intel-qat-netkey-test \
   openssl-qat openssl-qat-module \
   zlib-qat zlib-qat-module \
   calgary-corpus canterbury-corpus \
   silesia-corpus \

intel-dpdk"

d. Remove all of the above packages added to MACHINE_EXTRA_RECOMMENDS

   MACHINE_EXTRA_RRECOMMENDS += ""

   In this way, it is ensured that the Yocto build process will not try to locate and build Intel® DPDK or Intel® QuickAssist Technology related packages.

20. Optional: To build the disk image with a newer version of the GbE driver (when available), modify the `~/poky/meta-intel/meta-mohonpeak/recipes-igb/igb/igb.bb` file to match the newer `PV` string, `SRC_URI[md5sum]`, and `SRC_URI[sha256sum]`. Use the commands `md5sum` and `sha256sum` to find the updated checksums.

    Download the newer driver package, following the instructions in Section 1.2. Copy the newer driver package, in the format `igb-<version>.tar.gz`, to the `~/poky/meta-intel/meta-mohonpeak/recipes-igb/igb/igb/` directory. It is also recommended to remove the older version of the igb package from the same directory.

21. Optional: To add support for the new graphics card, it is recommended to use a graphics card that is compatible with X.org driver. Please refer to following configuration file and links used for Matrox Graphics card.

    — `/home/user/poky/meta-intel/common/recipes-graphics/xorg-driver/xf86-video-mga_1.6.2.bb`

    — http://git.yoctoproject.org/cgit/cgit.cgi/meta-intel/tree/common/recipes-graphics/xorg-driver/

    — http://git.yoctoproject.org/cgit/cgit.cgi/meta-intel/tree/common/recipes-graphics/xorg-driver/xf86-video-mga

22. Build the SDK image by running the commands:

```
# cd ~/poky/build
# bitbake core-image-sato-sdk
```

    Note: `bitbake` cannot run as a root user.

    If any errors are encountered, consult the Readme file at `~/poky/meta-intel/meta-mohonpeak/README`.

    Note: Please note the following:

    — If there are build errors that appears to be script errors such as the one shown in the following example, verify that `/bin/sh` is linked to `/bin/bash` as described in step 7.

```
[: 128: cmdline: unexpected operator |
```

    — If the build fails and `bitbake` command needs to be executed again, repeat step 15 to source oe-init-build-env.

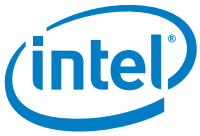
    — Log files (including errors and warnings) for the build are included in the `~/poky/build/tmp/log` directory.

    — If an error is returned stating `bitbake` is not installed, verify that you sourced the `oe-init-build-env file` in `~/poky` as described in Step 15.

    — The command may take several hours to complete, depending on the particular software development machine and network speed.

23. Verify that the `hddimg` is created in `~/poky/build/tmp/deploy/images/core-image-sato-sdk-mohonpeak-<timestamp>.hddimg`

    Note: If the build failed, this file may exist with no content. Verify that the file size is on the order of hundreds of MB.



6.2 Creating the Linux Boot Disk

6.2.1 Locating the hddimg

If you have created your own `hddimg` via the process in Section 6.1, your `hddimg` is located in `~/poky/build/tmp/deploy/images/core-image-sato-sdk-mohonpeak-<timestamp>.hddimg`.

6.2.2 Creating the Boot Disk

Note: Special care must be taken when creating the boot disk, since any misidentification of the target disk can overwrite critical data. You are encouraged to backup your data if there is any doubt about which disk you will be writing to in the following steps.

Select which of the following two sections to complete depending on the type of boot disk you are creating.

6.2.2.1 Creating a USB Boot Disk

Note: To create a USB Boot Disk using SDK image, follow the instructions in Section 6.2.2.2. The following instructions may not create a bootable disk for SDK image in all cases.

1. Identify the device name for your USB boot disk. In a typical case, this will be `/dev/sdf`. Study the output of one or more of the following commands to give confidence as to which disk is which:

```
# sudo parted -l
# df -h
# cat /proc/partitions
```

Caution: The following commands assume that your flash drive is `/dev/sdf`. Modify the commands if necessary. Failure to use the correct path for the target drive may result in loss of data.

2. Clone the image into the flash drive as follows:

```
# sudo dd if=/dev/zero of=/dev/sdb bs=1M count=512
# sudo sync
# sudo dd if=<path_to>/<image_name>.hddimg of=/dev/sdf
# sudo sync
# sudo eject /dev/sdf
```

6.2.2.2 Creating a SATA Boot Disk

1. Identify the device name for your SATA disk. In a typical case, this will be `/dev/sdb`. Study the output of one or more of the following commands to give confidence as to which disk is which:

```
# sudo parted -l
# df -h
# cat /proc/partitions
```

   Note: The following commands assume that the target SATA drive is `/dev/sdb`. Modify the commands if necessary. Failure to use the correct path for the target drive may result in loss of data.

2. Determine the physical characteristics of the target disk:

```
# fdisk /dev/sdb
```

   At the fdisk prompt, enter `p` to print out the physical characteristics of the disk. The output will include the information on the number of heads, sectors per tracks, and number of cylinders. Save this information for the next step.

   Enter `q` to exit the fdisk prompt.

3. Create the correct partition table and partition, substituting the correct value from the previous step for the number of cylinders, heads, and sectors per track.

   ```
   # mkdiskimage -4 /dev/sdb <cylinders> <heads> <sectors_per_track>
   ```

   The command above creates a FAT32, bootable partition 4.

   This command may take an hour or more for large disks. Choose a smaller value for the number of cylinders (for instance, 1024) to complete the command faster.

   Note: If `mkdiskimage` is not available on your system, you can use other software (for instance, `parted`) to create the partition table and the partition. The partition should be FAT32 with only the boot flag set. If your partition number is not 4, you should replace the remaining sdb4 references in this section with the `sdb<#>` that corresponds to your partition number.

4. Copy the contents of the `hddimg` to the device:

   ```
   # mkdir /tmp/Yocto-image
   # mkdir /tmp/Yocto-hdd
   # mount -o loop <path_to>/<img_file.hddimg> /tmp/Yocto-image
   # mount /dev/sdb4 /tmp/Yocto-hdd
   # cp -rf /tmp/Yocto-image/* /tmp/Yocto-hdd
   ```

5. Install the syslinux boot loader:

   ```
   # syslinux /dev/sdb4
   ```

6. Unmount the devices:

   ```
   # umount /tmp/Yocto-image
   # umount /tmp/Yocto-hdd
   ```

6.3 Updating EEPROM Image

The EEPROM image may need to be updated to the current version, which is available in the EDK (see document 516867).

Section 7.5, “Updating EEPROM Image” on page 56 includes instructions to update the EEPROM. Please note that updating the EEPROM image as detailed in Section 7.5 may require additional steps from earlier sections in Section 7.0, “Installing and Setting Up Fedora*” on page 51. Also note that `lanconf` is available for other environments, including the EFI shell.

6.4 Running the SDK Image

1. Transfer the boot disk to the Mohon Peak CRB.
2. Boot the Mohon Peak CRB. Select the correct boot device in the BIOS menu if necessary.

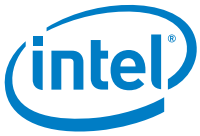
Note: Select the UEFI boot device; if that is not working, try the legacy boot device. If the method described in Section 6.2.2.1, “Creating a USB Boot Disk” on page 47 fails then use the method described in Section 6.2.2.2, “Creating a SATA Boot Disk” on page 47 to create bootable USB.

3. Depending on your particular video card, you may see a graphical desktop or a text prompt to login. If you see the text prompt, log in as root.
4. Runtime Package Management application `smart`:

   The application for performing runtime package management of RPM packages on the target is called `smart`. You can now use the `smart` query and `smart` install commands to find and install packages from the repositories.

Note: The `PACKAGE_CLASSES` variable should have `"package_rpm"` selected in ~/poky/build/conf/local.conf: `PACKAGE_CLASSES = "package_rpm"`.

To add x86_64 repository mirror:



```
# smart mirror --add http://downloads.yoctoproject.org/releases/yocto/yocto-1.4.2/rpm/x86_64/
```

To add 32-bit mirror:

```
# smart mirror --add http://downloads.yoctoproject.org/releases/yocto/yocto-1.4.2/rpm//i686_nativesdk/
```

To search the package

```
# smart search <package_name>
```

To install the package

```
# smart install <package_name>
```

Alternately, the RPM can be installed from its location as:

```
# smart install http://downloads.yoctoproject.org/releases/yocto/yocto-1.4.2/rpm/core2/<package_name>.core2.rpm
```

5. Prior to installing Intel® QuickAssist Technology software, the following steps are required. This step also ensures that SDK toolchain is properly installed.

```
# cd /usr/src/kernel
# make scripts
# ln -s /usr/src/kernel  /lib/modules/3.4.18-yocto-standard/build
```

Note: See Section 1.2, “Where to Find Current Software and Documentation” on page 9 to find the applicable package versions for Intel® QuickAssist Technology software, Intel® DPDK, Gigabit Ethernet driver, libcrypto, netkey, and Yocto BSP packages.

6.5 Intel® QuickAssist Technology

See Section 1.2, “Where to Find Current Software and Documentation” on page 9 for details.

6.6 Intel® Data Plane Development Kit

See Section 1.2, “Where to Find Current Software and Documentation” on page 9 for details.

6.7 libcrypto* (OpenSSL*) Sample Patch for Intel® QuickAssist Technology

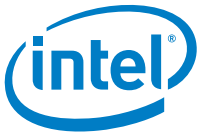
See Section 1.2, “Where to Find Current Software and Documentation” on page 9 for details.

6.8 Linux* Kernel Cryptographic Framework Sample Driver for Intel® QuickAssist Technology

See Section 1.2, “Where to Find Current Software and Documentation” on page 9 for details.

Part 3: Running on Fedora*

This section contains the following chapters:



7.0 Installing and Setting Up Fedora*

7.1 Acquiring Fedora

Fedora 17 is a Linux distribution built on free and open source software. The Intel® Atom™ Processor C2000 Product Family for Communications Infrastructure Software for Linux* package does not include a distribution of Fedora or any other Linux variant. The package includes Linux device driver source developed by Intel.

The software package has passed basic testing with the Linux distributions listed below:

- Fedora 17 x86_64 can be obtained from:
  http://archive.fedoraproject.org/pub/fedora/linux/releases/17/Fedora/x86_64/iso/
- Fedora 17 32-bit can be obtained from:
  http://archive.fedoraproject.org/pub/fedora/linux/releases/17/Fedora/i386/iso/

Note: This section is written with the Fedora 17 Install Media in mind. Using the Live Media version is not recommended.

7.2 Configuring the BIOS on the Development Kit

Note: Ensure that you have an updated BIOS image (see document 518736) and that you have read and are familiar with Chapter 3.0, "System Setup and Installation" and Chapter 4.0, "Customer Reference Board (CRB) Features."

Power on the board, and insert the Fedora DVD into the DVD-ROM drive.

The BIOS may be configured to boot directly from DVD. If the machine boots to an EFI prompt, enter `exit` to exit EFI. Under the BIOS settings screen, the override for a boot can be found under `Boot Manager Menu`. Use this to boot from DVD.

To change the boot order of every boot, select `Boot Maintenance Manager -> Boot Options -> Change Boot Order`. Press `<Enter>` to edit the boot order, and press `+` or `-` to move the boot options up or down, respectively.

7.3 Installing Fedora

For complete Fedora installation instructions, please refer to the online Installation Guide at:

http://docs.fedoraproject.org/en-US/Fedora/17/html/Installation_Guide/

This section contains basic installation instructions. For the purposes of this Getting Started Guide, it is assumed that the installation is from a DVD image.

Note: Due to the graphics cards included with many of the development systems, the installation may be text only, also known as a *basic graphics mode* installation. This may result in some different steps below and some packages not being installed, including the graphical desktop. If the full graphical desktop is required or desired, there are three options:

a. On the Mohon Peak CRB system, use a different video card than the one provided by Intel,

b. Install the operating system using another Intel Architecture-based development system that supports the graphical installation, or

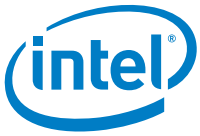
c. Install the graphical desktop after the operating system has been installed. For details, see Section 7.3.1, “Working Around Linux* Boot Errors” on page 53.

Note: If the hard drive already has an operating system, some of the following steps may be slightly different.

1. Select the language to use during the installation process and click **Next**.
   Select the appropriate keyboard for the system and click **Next**.
2. If specialized storage devices are not required, select **Basic Storage Devices** and click **Next**.

   Note: The following warning may be displayed when installing on a blank disk: "REINITIALIZING WILL CAUSE ALL DATA TO BE LOST!"
   If you definitely know the hard drive in use is blank, proceed. Otherwise, power down and replace the hard drive.

   If the hard drive is blank, select **Re-initialize all**, then **Next**.
   If the disk has existing Linux installation, select **Fresh Installation**, then **Next**.
3. Enter a host name for the computer and click **Next**. The default host name will be acceptable in most cases.
4. Select the nearest city to your time zone and click **Next**.
5. Enter the desired root password for the system in the box labeled **Root Password**: and reaffirm the root password by entering the same password in the **Confirm:** box. Click **Next** to continue.
6. Select the hard drive where Fedora will be installed and the drive options. Options include **Use All Space** and **Replace Existing Linux System(s)**. Click **Next** to continue.
7. If multiple drives are visible to the system, you may be asked to select a target device. Select the appropriate target device, select the right arrow, and click **Next**.
8. Select **Write changes to disk**.
9. Select the software to install. Ensure that **Software Development** is selected. Click **Next** to continue.
10. When the installation completes, the install DVD should be ejected. (If the install DVD was not ejected, eject it.) Remove the DVD and select **Reboot** when prompted.



Note: If the system does not begin rebooting within 60 seconds, power cycle the board or press the reset button on the board, and continue.

Due to compatibility issues between the video card supplied by Intel and some recent Linux distributions, the operating system may not load correctly. If this is the case, see Section 7.3.1 for more options.

11. After the reboot, some normal system setup is required, such as creating user account(s) and setting the time and date. Note that a user account is required to boot into the Fedora GUI. When asked, follow the prompt by clicking the **Forward** button.
12. When the installation completes, continue with the next section.

7.3.1 Working Around Linux* Boot Errors

Due to compatibility issues between the video card supplied by Intel and some recent Linux distributions, the operating system may not load correctly.

To work around this:

1. Interrupt the Linux boot by pressing `e` at the `grub` menu.
2. Use the down arrow key to move to the line starting with `linux`.
3. Add the option `3` to the end of the line, separated by a space from any other options.
4. Press **F10** to resume booting.

Depending on your use case and install details, select one of the following subsections to complete:

- Booting to text mode permanently
- Adding support for the graphical desktop
- Repairing the Monitor Configuration File

7.3.1.1 Booting to text mode permanently

1. Update the `/etc/default/grub` file to add the `GRUB_CMDLINE_LINUX` option `3`. This can added between `rhgb` and `quiet`, offset by spaces.
2. Save the changes to the file and execute the following command to generate the `grub` configuration file:

   ```
   # grub2-mkconfig -o /boot/grub2/grub.cfg
   ```

3. Reboot the system.

   ```
   # shutdown -r now
   ```

7.3.1.2 Adding support for the graphical desktop

If you installed Linux via the text-based install, you did not install support for the graphical desktop. This section gives the steps necessary to add this support.

Note: Many other packages will not have installed correctly if you did a text-based install. In most cases, the missing command support can be added with the following:

```
# yum -y install <command_name>
```

1. Update the yum configuration files as described in Section 7.4.1.1, if necessary.

   Note: The onboard GbE ports may not be functional until the steps in Section 7.6 have been completed.

2. Install X-windows and the GNOME Desktop Environment:

   ```
   # yum -y groupinstall "X Window System"
   ```

```
# yum -y groupinstall "GNOME Desktop Environment"
# mv /etc/systemd/system/default.target /etc/systemd/system/default.target.org
# ln -s /lib/systemd/system/graphical.target /etc/systemd/system/
default.target
```

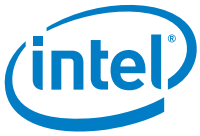
3. Install unzip:
   ```
   # yum -y install unzip
   ```
4. Install tar:
   ```
   # yum -y install tar
   ```
5. Download the ASPEED* Graphics Family BIOS/Driver Package from the site:
   http://www.aspeedtech.com/support.php
   At the webpage above, go to the **BIOS/Driver Package Downloads** section, browse to the latest version of the ASPEED Graphics Family BIOS/Driver Package, and download it.
6. Extract this file to the `/tmp` directory:
   ```
   # cd /tmp
   # unzip <path_to>/v<version>_whql.zip
   ```
7. Extract the driver:
   ```
   # cd v<version>_whql/Linux
   # tar xzof lxdrv.tar.gz
   ```
8. Install the driver:
   ```
   # cd /tmp/v<version>_whql/Linux
   # ./auto-update.sh
   ```
   Note: If the `auto-update.sh` script returns the following error:
   `Can't find proper X Window Version, Please update driver manually!!`
   perform the steps below to install the driver manually.
   Refer to FAQ – Package description section of `README.TXT` to select the proper driver package. For Fedora 17, it is `xorg77_2`.
   64-bit: `# cd /tmp/v<version>_whql/Linux/x86_64/xorg77_2`
   32-bit: `# cd /tmp/v<version>_whql/Linux/x86/xorg77_2`
   ```
   # chmod u+x *
   # ./update.sh
   ```
9. Shutdown the system:
   ```
   # shutdown -h now
   ```
10. Power on the system. It should now boot into the graphical desktop mode.

7.3.1.3 Repairing the Monitor Configuration File

Some versions of Linux will crash or fail to boot when loading the default graphics driver with the graphical desktop. One workaround to this issue is to use the `vesa` driver instead of the `ast` driver.

1. To use the `vesa` driver, create (or replace) the file `/etc/X11/xorg.conf.d/10-monitor.conf` with the following text:
   ```
   Section "Monitor"
       Identifier             "Monitor0"
   EndSection

   Section "Device"
       Identifier             "Device0"
       Driver                 "vesa"
   EndSection
   ```



```
Section "Screen"
    Identifier             "Screen0"
    Device                 "Device0"
    Monitor                "Monitor0"
    DefaultDepth            16
    SubSection             "Display"
        Depth               16
        Modes              "1024x768_75.00"
    EndSubSection
EndSection
```

Note: The specific mode `1024x768_75.00` may not work on all monitors.

2. Reboot the system:

```
# shutdown -r now
```

7.4 Configuring Linux

Once the operating system is installed, there are a few configuration items that may need to be completed, such as updating the yum configuration files and enabling the development platform to operate within a corporate network. This section describes these items.

7.4.1 Updating yum Configuration Files

yum is an application that can be used to perform operating system updates. In order to use yum in a corporate network, the following change may be required.

7.4.1.1 /etc/yum.conf

If the system needs to connect to internet through a corporate firewall, yum needs to be updated to use the proxy server. Add a line similar to the following in the `/etc/yum.conf` file. The line can be added to the end of the file. Contact your network administrator for details on the proxy server.

```
proxy=http://<proxy_server:portnum>
```

where `<proxy_server:portnum>` is replaced with your server information.

7.4.2 Network Connectivity on Corporate Network

To get the development platform onto a corporate network, it may be necessary to modify the Network Proxy settings as described in this section.

7.4.2.1 Network Proxy Settings in Linux*

To get the development platform onto a corporate network, it may be necessary to modify the Network Proxy settings under the **Activities > Applications > System Settings > Network > Network Proxy** tab in Fedora. Change the Network Proxy dropdown box from None to Manual. The network proxy is used by Linux to set up the correct connectivity out to the internet.

7.4.2.2 Network Proxy Settings in Browser

Network proxy also needs to be set in the internet browser. Open the browser, for example, Mozilla Firefox*, and perform the following:

1. Click **Edit > Preferences**.
2. Click **Advanced**, and then the **Network** tab.

3. Click **Settings**.
4. Click the **Automatic proxy configuration URL** button to enter proxy information.

7.4.2.3 Proxy Settings for Shell Prompt

The Linux environment variable `http_proxy` allows you to connect text-based applications via the proxy server. To set up the proxy environment variable, add a line similar to the following in the `/etc/environment` file. The line can be added to the end of the file. Contact your network administrator for details on the proxy server.

```
http_proxy=http://<proxy_server:portnum>
```

where `<proxy_server:portnum>` is replaced with your server information.

Note: It will be necessary to log out and then back in again for these settings to take effect.

7.5 Updating EEPROM Image

The EEPROM image may need to be updated to the current version, which is available in the EDK (see document 516867).

Perform the following steps to update the EEPROM image:

1. Download the EEPROM image following the instructions in Section 1.2.
2. Extract the package. The EEPROM images are located in the `\EEPROMS` directory.
3. Download the Intel® Network Connections Tools (formerly codenamed Quartzville) from the EDK, see document 348742.
4. Extract the Intel® Network Connections Tool package and change to the correct directory:

   ```
   # cd /tmp
   # unzip <path_to>/348742_Quartzville_Tools_<version>.zip
   ```

   If running 32-bit Linux:

   ```
   # cd TOOLS/Linux32/SVTools
   ```

   If running 64-bit Linux:

   ```
   # cd TOOLS/Linux_x64/SVTools
   ```

5. Copy the `.hex` EEPROM image to the current directory:

   ```
   # cp <path_to>/Avoton_<version>.hex .
   ```

6. Install the network tools:

   ```
   # chmod +x install
   # ./install
   ```

   Note: The error below may be returned; it can be safely ignored.

   ```
   Error: Module iqvlinux is not currently loaded
   Installed!!
   ```

7. Run `lanconf`:

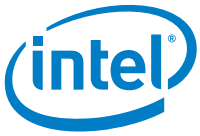
   For 64-bit:

   ```
   # chmod +x lanconf64e
   # ./lanconf64e
   ```

   For 32-bit:

   ```
   # chmod +x lanconf32
   # ./lanconf32
   ```

8. Select the first NIC (device ID is 8086-1F41), and press `<Enter>`.
   If no external NICs are installed in the system, this will be preselected.
9. Arrow down to `<EEPROM/FLASH>` and press `<Enter>`.



10. Select <Raw EEPROM> and press <Enter>.
11. Press <F4> and select the appropriate EEPROM file. You are asked if you would like to keep the existing MAC address. Select YES and press <Enter>.

    Note: If you encounter errors at this step, ensure that you have selected the correct .hex file.
12. Press <ESC>, and then confirm the EEPROM write by selecting YES and pressing <Enter>.
13. After the EEPROM has been updated, press ESC 3 times to get back to the list of NICs. Press <x> to exit lanconf.
14. Restart the system using the following command:

    ```
    # shutdown -r now
    ```

7.6 Installing the 4x 1GbE Network Driver

If required, install the 1 GbE Network driver; see "Intel® Ethernet Drivers and Utilities" in Section 1.2.1, “Accessing Content from the Open Source Technology Centre” on page 9.

1. Following the instructions in Section 1.2, download the Networking Software Drivers for Edisonville and Rangeley PV package, which contains the GbE Network Driver.
2. Extract the package. The network driver is located in the \PRO1000\LINUX directory.
3. Extract the network drivers package and change to the correct directory:

    ```
    # cd /tmp
    # tar xzof igb-<version>.tar.gz
    ```
4. Change to the src directory:

    ```
    # cd igb-<version>
    # cd src
    ```
5. Build and install the driver:

    ```
    # make
    # make install
    ```
6. Load the driver:

    ```
    # modprobe igb
    ```
7. Restart network stack:

    ```
    # service NetworkManager restart
    ```
8. Verify that the 4x 1GbE network ports are visible using ifconfig. Verify that these network ports can obtain an IP address and pass traffic.

If the 4x 1GbE network ports are not all visible with ifconfig or the network ports cannot obtain an IP address, verify the following items:

- Using latest EEPROM version, refer to the Section 7.5

8.0 System Security Considerations

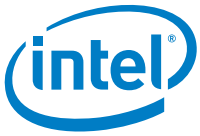
This section contains a high-level list of system security topics. Specific OS/filesystem topics are outside of the scope of this document. For more information, see the Rangeley Software Programmer's Guide, specifically the Secure Architecture Considerations section.

Securing your operating system is critical. You should consider the following items:

Note: This is not an exhaustive list.

- Employing effective security policies and tools; for instance, SELinux is configured correctly and is active
- Running and configuring the firewall(s)
- Preventing privilege escalation at boot (including recovery mode)
- Removing unnecessary software packages
- Patching software in a timely manner
- Monitoring the system and the network
- Configuring and disabling (as appropriate) remote access
- Disabling network boot
- Requiring secure passwords
- Encrypting files, up to full-disk encryption
- Ensuring physical security of the system and the network
- Using mlock to prevent swapping sensitive variables from RAM to disk
- Zeroing out sensitive variables in RAM

§ §



9.0 Building and Installing the Software

This chapter provides details on building and installing the software on the development kit.

9.1 Unpacking the Software

The software package comes in the form of a tarball. See If required, install the 1 GbE Network driver, see "Intel® QuickAssist Technology Drivers and Patches" in Section 1.2.1, “Accessing Content from the Open Source Technology Centre” on page 9.

The instructions in this document assume that you have super user privileges.

```
# su
<enter password for root>
```

1. Create a working directory for the software. This directory can be user defined, but for the purposes of this document, a recommendation is provided.

   ```
   # mkdir /QAT
   # cd /QAT
   ```

   Note: In this document, the working directory is assumed to be `/QAT`.

2. Transfer the tarball to the development board using any preferred method, for example USB memory stick, CDROM, or network transfer in the `/QAT` directory. Unpack the tarball using the following command:

   ```
   # tar -zxof <tarball name>
   ```

3. Restricting access to the files is recommended:

   ```
   # chmod -R 770 /QAT
   ```

The installation script and driver files are created under the `/QAT` directory.

9.2 Installation Script

An installation script is provided that walks you through building/installing the software. Refer to Section 9.3 for instructions on installing the software on a new platform.

The installation script can also be launched with command line arguments giving the option to bypass the interactive setup. Refer to Section 9.2.2 for additional information.

For details on minimizing Acceleration Software compilation time, refer to Section 9.7.

The *Programmer's Guide* for your platform describes required and optional build flags in the “Build Flag Summary” section. If you are using the installation script, the required build flags are handled by default. The optional build flags control the driver functionality and can be used with the installation script or the command line arguments.

The installation script file must be run as root. If the script is executed as a user other than root, the following error is returned: `ERROR This script must be run as root.`

Launch the script using the following command:

```
# ./installer.sh
```

A welcome message is displayed, followed by Installation Options. The table below lists the available installation options.

Note: If you have not unpacked the release package tarball as described in Section 9.1, errors will be returned.

Table 9. Installation Options

Option	Name	Description
1	Build	Builds the acceleration software/sample code mentioned in the Configuration. The software is not installed.
2	Clean Build	Cleans the previously built software.
3	Install	Installs the acceleration software/sample code mentioned in the Configuration
4	Uninstall	Uninstalls the software.
5	Show Accel Info	Displays the number of chipset devices available on the system and the B:D:F for each device.
6	Change Configuration	Option to change the configuration to another combination. There is a sub-menu that allows you to select which software components are built/installed.
0	Exit	Exit the installation script.

9.2.1 InstallerLog

The `InstallerLog.txt` file is appended after each installation with the time/date and the output of the build/install. If any issues were seen during the installation, check the log file for details.

9.2.2 Command Line Arguments

The *Programmer's Guide* for your platform describes required and optional build flags in the “Build Flag Summary” section. If you are using the installation script, the required build flags are handled by default. The optional build flags control the driver functionality and can be used with the installation script or the command line arguments.

If the kernel source is not at location `"/usr/src/kernels/`uname -r`"`, then export the kernel source path to an appropriate location:

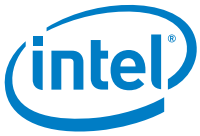
```
export KERNEL_SOURCE_ROOT=<path_to_kernel_source>
```

Use the installer to help to select available options: `#./installer.sh` help

Usage: `./installer.sh <options>`

You may choose options from the following groups:

- actions: `build / install / clean / uninstall / version / help`
- driver_options: `QAT1.5 / QAT1.6 / mux / QAT1.6_mux`
- sriov_options: `host / guest`
- service_options: `build_sample / build_dc_only / gige`



- path: Location where you would like to build the Intel® QuickAssist Technology package

Note: Do not combine action and service options, and be sure to start your path with /

Example usage:

- To install acceleration for C2000 devices only `./installer.sh` install QAT1.5
- To install acceleration for C2000 and DH895xcc devices `./installer.sh` install mux

Refer to "`./installer help`" output for details.

Note: Not all command line options are supported by every software package.

9.3 Installation on a New Development Platform

This section walks you through the installation of the software on a new development platform.

9.3.1 Installing the Acceleration Software

1. Power on the system and proceed with the instructions below.
2. Open a Terminal Window and switch to superuser.
   ```
   # su
   <enter root password>
   ```
3. In the `/QAT` directory, start the installation script.
   ```
   # cd /QAT
   # ./installer.sh
   ```
   Select the `Install Acceleration` installation option. This installs the Acceleration software. You will be prompted for a directory location to build the package and the Build Output Directory. Use the default values provided by the installation script.

   Note: ICP_ROOT is automatically set by installer.sh and can vary during the installer run, depending on which driver is being built, i.e., it is set to /QAT/QAT1.x/ during the build of each QAT1.x driver.

   Note: Some older Linux distributions may require the grub boot parameter `intel_iommu=off`. If a failed to start device error is shown during the Acceleration installation, this parameter may be required.

   Note: If Software Development was not selected during the OS install, you may see an error message similar to the following during the acceleration install:
   ```
   "make: *** /lib/modules/3.3.4-5.fc17.x86_64/build/: No such file or
   directory. Stop."
   ```
   Reinstall the OS and select the `Software Development` option as described in Section 7.3, or run the following commands:
   ```
   # yum -y install kernel-devel-$(uname -r)
   # yum -y install gcc
   ```
4. During the installation, the following message is displayed:
   ```
   *** No error detected in InstallerLog.txt file ***
   ```
   At the end of the installation, the following messages are displayed:
   ```
   *** Acceleration Installation Complete ***
   ```
   Refer to the `InstallerLog.txt` file for additional detail on the installation. It is also a good idea to check `/var/log/messages` or `dmesg` to make sure that the acceleration service started.

Note: After building/installing the Acceleration Software, it is highly recommended to secure the build output files (the files located in `$ICP_ROOT\build`) by either deleting them or setting permissions according to your needs.

5. Use the `6` option to exit the installation.
6. After installing the Acceleration software, it is recommended to verify that the acceleration software kernel object is loaded and ready to use. This can be done by performing the following operation:

   ```
   # lsmod | grep qa
   ```

   If the appropriate kernel object is not returned (either `icp_qa_al` or `qat_1.5_mux` — see the Rangeley Software Programmer's Guide for build object details), then the acceleration software is not installed and is not ready for use. Refer to the `Installer.log` file in the `/QAT` directory for additional information. If necessary, run the installation script again and select `Install Acceleration`.

The command to verify the SoC is enumerated properly is:

```
# lspci  -d :1f18
00:0b.0 Co-processor: Intel Corporation Device 1f18 (rev 01)
```

Once the installation/building is complete, proceed to Section 10.0 to execute applications that exercise the software.

9.4 Cross-Compilation Capabilities for Intel® QuickAssist Technology

These capabilities enable users to compile the Intel® QuickAssist Technology driver on one system for the targeted architecture of another system.

Note: The following instructions are valid for Fedora. Steps for Yocto will be included in a future revision.

If you are cross-compiling for the first time, you need to install the following packages:

- `yum -y install glibc-devel.i686`
- `yum -y install openssl-devel.i686`
- `yum -y install libgcc.i686 --setopt=protected_multilib=false`
- `cp /lib/libz.so.1.2.5 /usr/lib/libz.so`

If cross-compiling for both `QAT1.x` drivers and the QATmux, then:

```
# export WITH_CPA_MUX=1
```

Use the following commands to cross-compile each `QAT1.x` driver:

```
# export ICP_ROOT=<QATdir>
```

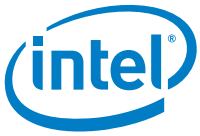
Note: Where `<QATdir>` is `/QAT/QAT1.6` or `/QAT/QAT1.5`, depending on what hardware support is required. If building with mux support, always use the QAT1.6 sample code.

```
# cd quickassist
# export ICP_ENV_DIR=$ICP_ROOT/quickassist/build_system/build_files/env_files
# export ICP_BUILDSYSTEM_PATH=$ICP_ROOT/quickassist/build_system
# export ICP_BUILD_OUTPUT=$ICP_ROOT/build
# export ICP_TOOLS_TARGET=accelcomp
# export LD_LIBRARY_PATH=$ICP_ROOT/build
```

For i386 machines:

```
# make ICP_ARCH_USER=i386
```

For i686 machines:



```
# make ICP_ARCH_USER=i686
```

Next, go to the build folder and enter:

```
# cd ../build
# file *
```

Here's a sample output for a non-mux build (see Section 1.5.3 for file names in mux build cases):

```
adf_ctl:                              ELF 32-bit LSB executable, Intel 80386,
version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.32, not
stripped
c2xxx_qa_dev0.conf:                   ASCII English text
dh89xxcc_qa_dev0.conf:                ASCII English text
dh89xxcc_qa_dev0_single_accel.conf:   ASCII English text
dh89xxcc_qa_dev1.conf:                ASCII English text
gige_watchdog_service:                Bourne-Again shell script, ASCII text
executable
icp_gige_watchdog:                    ELF 32-bit LSB executable, Intel 80386,
version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.32, not
stripped
icp_qa_al.ko:                         ELF 64-bit LSB relocatable, x86-64, version 1
(SYSV), not stripped
libadf_proxy.a:                       current ar archive
libicp_qa_al.a:                       current ar archive
libicp_qa_al_s.so:                    ELF 32-bit LSB shared object, Intel 80386,
version 1 (SYSV), dynamically linked, not stripped
libosal.a:                            current ar archive
mmp_firmware.bin:                     data
mmp_firmware_c2xxx.bin:               data
mof_firmware.bin:                     data
mof_firmware_c2xxx.bin:               data
qat_service:                          Bourne-Again shell script, ASCII text
executable
```

As you can see, the user files, `adf_ctl` and `libicp_qa_al_s.so`, are 32-bit files while `icp_qa_al.ko` is a 64-bit file.

Cross-Compiling the Sample Code for 32-Bit User Space

Change directories to the top-level sample code directory:

```
# cd $ICP_ROOT/quickassist/lookaside/access_layer/src/sample_code
```

Compile the user space sample application:

For i386 machines:

```
# make perf_user ARCH=i386
```

For i686 machines:

```
# make perf_user ARCH=i686
```

9.5 Starting / Stopping the Acceleration Software

When the Acceleration software is installed, a script file titled `qat_service` is installed in the `/etc/init.d` directory.

The script file can be used to start and stop the Acceleration software. To start the software, issue the following commands:

```
# service qat_service start
```

If the service `qat_service start` command fails, verify the following:

- Software is installed.
- Acceleration software is already running.

- Verify the SoC is enumerated properly using the `lspci` command described in Section 9.3.1.

To stop the software, issue the following command:

```
# service qat_service stop
```

To restart the software, issue the following command:

```
# service qat_service restart
```

To stop the software and remove the kernel driver, issue the following command:

```
# service qat_service shutdown
```

When the Acceleration software is installed, it is set to load automatically when the operating system loads.

9.6 Configuration Files

When the Acceleration software loads, it is configured based on settings in the configuration file (`c2xxx_qa_dev0.conf`). The configuration file is placed in the `/etc` directory. For example, the configuration files for the Intel® Atom™ Processor C2000 Product Family for Communications Infrastructure devices are: `c2xxx_qa_dev0.conf` and `c2xxx_qa_dev0_single_ae.conf`. The `c2xxx_qa_dev0_single_ae.conf` file is intended for use with SKUs that contain a single acceleration engine.

The files are processed when the system boots. If changes are made to the configuration file, the Acceleration software must be restarted for the changes to take effect. Refer to Section 9.5, “Starting/Stopping the Acceleration Software” on page 63 for detailed instructions.

The software package includes multiple types of configuration files. Depending on your installation options and SKU, a valid configuration file will be copied to the `/etc` directory for you.

Note: The software has been validated with the default configuration files included in the software package. Changes to the configuration files could have adverse effects.

Refer to the Rangeley Software Programmer’s Guide for additional information on the configuration files.

9.7 Minimizing Acceleration Software Compilation Time

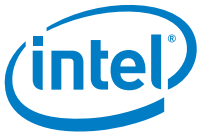
When compiling/installing the Acceleration Software, a "make clean" operation is performed. This results in rebuilding every source file included in the package, even if the source files are unaltered. The "make clean" is done to ensure a proper build is performed. Here are a few items that could cause issues that would require the "make clean":

- If any compile time build flags are added which may not be reflected in the already-built object files.
- If changes are made to header files, performing make clean prior to the make would be preferred.

If you are comfortable with these constraints, you can update the `$ICP_ROOT/quickassist/Makefile` line where ALL_TARGETS is defined and remove the 'clean' from the list of targets. This will remove the clean operation from the build process.

Before the update, the line looks like:

```
ALL_TARGETS = clean lac_lib_dir libosal libosal_user hal adf adf_user
```



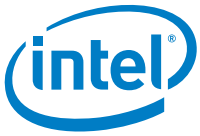
```
lac lac_user qat-fw install_scripts
```

After the update, the line looks like:

```
ALL_TARGETS = lac_lib_dir libosal libosal_user hal adf adf_user
lac lac_user qat-fw install_scripts
```

Part 4: Running Sample Applications

This section contains the following chapter:



10.0 Running Sample Applications

This section describes the sample code that can be executed on the target platform along with instructions on their usage for both Yocto and Fedora.

10.1 Intel® QuickAssist Technology Acceleration Sample Application

The software package contains a set of sample tests that exercises the SoC acceleration functionality. This section describes the steps required to build and execute the sample tests.

The sample application is provided for both Kernel Space and User Space and the following sections contain instructions for both.

Note: The memory driver included with the sample application is a sample memory driver and is not intended for actual deployment.

10.1.1 Compiling the Acceleration Sample Code

The acceleration sample code can be built from the installation script, or it can be compiled manually.

If the installer was used and all code built successfully, it may be possible to proceed directly to Section 10.1.2.

Note: These instructions assume the software package was untarred in the `/QAT` directory and the kernel source files were placed in the directory specified in this guide.

To build from the installation script, do the following:

1. Open a Terminal Window and switch to superuser:

```
# su
<enter root password>
```

Note: For details on running user space applications as non-root user, please refer to the "Running Applications as Non-Root User" section in the Rangeley Software Programmer's Guide.

2. In the `/QAT` directory, start the installation script.

```
# cd /QAT
# ./installer.sh
```

If the configuration section has built sample code for 1.x then select "build" to build the sample code. If the configuration section does not have sample code, select "`Change Configuration`" to option a1, "`Set Build Target as "sample_code only`", or use command line option "`./installer.sh build_sample [QAT1.5|QAT1.6|mux]`". This option compiles the Acceleration Sample code for both user space and kernel space. It also compiles the memory mapping driver used with the user space application.

You may be prompted for a directory location to build the package and the Build Output Directory. Use the default value for the location to build the package. The Build Output Directory parameter is ignored.

Note: In the case where both QAT1.5 and QAT1.6 drivers are loaded, the sample code from the QAT1.6 driver should be used.

3. Proceed to Section 10.1.2.1, "signOfLife Tests" on page 69 for instructions on executing the tests.

To manually compile the acceleration sample code, do the following:

1. The following environment variables must be set to build the modules:

```
# export ICP_ROOT=<QATdir>
# export ICP_BUILDSYSTEM_PATH=$ICP_ROOT/quickassist/build_system
# export ICP_ENV_DIR=$ICP_ROOT/quickassist/build_system/build_files/env_files
```

   Note: Where `<QATdir>` is `/QAT/QAT1.6` or `/QAT/QAT1.5`, depending on what hardware support is required. If building with mux support, always use the QAT1.6 sample code.

2. If intermediate modules are required, the following variables must also be set:

```
# export ICP_BUILD_OUTPUT=$ICP_ROOT/build
# export ICP_TOOLS_TARGET=accelcomp
```

3. The sample code is compiled with the default assumption that the kernel source header files are located in one of the following directories:

   64-bit: `/usr/src/kernels/3.1.0-7.fc16.x86_64`

   32-bit: `/usr/src/kernels/3.1.0-7.fc16.i686`

4. If the kernel source header files are located in a different directory, create the environment variable with the directory of desired target kernel sources. For example:

```
# export KERNEL_SOURCE_ROOT=/usr/src/kernels/linux
```

   For Yocto:

```
# export KERNEL_SOURCE_ROOT=/usr/src/kernel
```

5. You can compile for both Kernel space and User space at the same time using the following commands:

```
# cd $ICP_ROOT/quickassist/lookaside/access_layer/src/sample_code
# make perf_all
```

   The generated Linux kernel object and sample application are located at:

```
$ICP_ROOT/quickassist/lookaside/access_layer/src/sample_code/build
```

6. Proceed to Section 10.1.2.1, "signOfLife Tests" on page 69 for instructions on executing the tests.

10.1.2 Loading the Sample Code

1. The acceleration kernel module must be installed and the software must be started before attempting to execute the sample code. This can be verified by running the following commands:

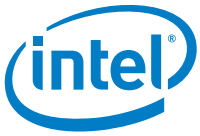
```
# lsmod | grep icp_qa_al
# service qat_service status
```

   The `lsmod` command verifies the kernel driver is installed.

   The `qat_service` command reports the status of the device.

   Typical output is:

```
There is 1 acceleration device(s) in the system:
icp_dev0 - type=c2XXX, inst_id=0, bsf=00:0b:0, #accel=1, #engines=2, state=up
```



Note: If the module is not returned from the first command, refer to Section 9.5, "Starting/Stopping the Acceleration Software" on page 63 for additional information on starting the Acceleration software.

2. The sample code is executed by installing the `cpa_sample_code` kernel object for kernel space, or by launching the application for user space.

   The application allows the kernel parameters listed below.

Table 10. Sample Code Parameters

Parameter	Description
signOfLife=v	Indicates shorter test run that verifies the acceleration software is working. This parameter executes a subset of sample tests. Details are included in Section 10.1.2.1. (default=0)
cyNumBuffers=w	Number of buffers submitted for each iteration. (default=20)
cySymLoops=x	Number of iterations of all symmetric code tests. (default=5000)
cyAsymLoops=y	Number of iterations of all asymmetric code tests. (default=5000)
runTests=1	Run symmetric code tests.
runTests=2	Run RSA test code.
runTests=4	Run DSA test code.
runTests=8	Run ECDSA test code.
runTests=32	Run Stateless Compression test. (Not supported)
runStateful=1	Run Stateful Compression test. (Not supported)
runTests=63	Run all tests. (default)
configFileVer	Version of configuration file. Can be 1 or 2 (default). If you are using the original version 1 configuration file, use 1. For configuration file details, see the Rangeley Software Programmer's Guide.
wirelessFirmware	Wireless Firmware enabled. Can be 0 (default) or 1. This parameter can only be used with the version 2 configuration file when wireless instances are configured. For configuration file details, see the Rangeley Software Programmer's Guide.

10.1.2.1 signOfLife Tests

The `signOfLife` parameter is used to specify that a subset of the sample tests are executed with smaller iteration counts. This provides a quick test to verify the acceleration software and hardware are set up correctly.

Note: If the signOfLife parameter is not specified, the full run of tests can take several hours to complete. In addition, for RSA 4096 and DH 4096 tests, there can be up to an hour with no perceived result activity.

Kernel Space

After building the sample code, the kernel space kernel driver, the user space application, and the memory mapping driver are located at:

```
$ICP_ROOT/quickassist/lookaside/access_layer/sample_code/build
```

To execute the sign of life test in Kernel space, use the following commands:

```
# cd $ICP_ROOT/quickassist/lookaside/access_layer/src/sample_code/build
# insmod ./cpa_sample_code.ko signOfLife=1
```

Note: This test takes a few minutes to complete. When the `insmod` command is executed, there is no indication on the terminal window of the activities.

Instructions on viewing the results are included in Section 10.1.3, "Test Results" on page 71.

Note: If loading of the module fails and some messages in `/var/log/messages` show `Device 0 not found or not stated` or `There are no crypto instances`, ensure that the kernel option `intel_iommu=off` has been configured as a kernel boot parameter, as may be required for older Linux distributions.

User Space

After building the sample code with the installation script, the kernel space kernel driver, the user space application, and the memory mapping driver are located at:

```
$ICP_ROOT/quickassist/lookaside/access_layer/sample_code/build
```

To execute the sign of life test in User space, use the following commands:

Install the kernel memory driver `qaeMemDrv.ko`, if the module has not already been installed.

```
# insmod $ICP_ROOT/quickassist/lookaside/access_layer/src/sample_code/build/
qaeMemDrv.ko
# cd $ICP_ROOT/quickassist/lookaside/access_layer/src/sample_code/build
# ./cpa_sample_code signOfLife=1
```

Note: You will observe that execution time of the user space code takes longer than the kernel space code. This is due to the sample code kernel space memory management driver (`qaeMemDrv.ko`), which is slow to allocate and map memory to user space. Before beginning performance measurements, the sample code allocates memory up-front which slows execution time. This does not affect the performance of the acceleration driver itself. The acceleration driver user space and kernel space performance are equivalent, other things being equal (for instance, no throttling takes place in either case).

10.1.2.2 Wireless Tests

Note: The Wireless tests are supported in user space only. They are not supported in kernel space.

The software package includes a version of the firmware optimized for small cryptography packets. In order to run the sample code with this firmware, the following steps must be performed.

1. An example configuration file is included in the package.

   ```
   # cp $ICP_ROOT/quickassist/config/c2xxx_qa_dev0.conf.v1.wireless /etc/
   c2xxx_qa_dev0.conf
   ```

2. Restart the acceleration service using the command:

   ```
   # /etc/init.d/qat_service restart
   ```

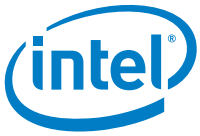
3. Run the sample code using the command:

   ```
   # cd $ICP_ROOT/quickassist/lookaside/access_layer/src/sample_code/build
   # ./cpa_sample_code wirelessFirmware=1
   ```

   Note: The `wirelessFirmware=1` parameter must be specified when the wireless firmware is used. The following error messages are displayed if the parameter is not specified.

   ```
   There are no crypto instances available
   setupSymmetricTest():1135 Failed to start Crypto services
   main():636 Error calling setupCipherTest
   ```

   Note: The wireless firmware must be used if the `wirelessFirmware=1` parameter is specified. The following error messages are displayed if you specify the parameter and the firmware is not being used.



```
[error] SalCtrl_CryptoInit() - : Failed to get Cy0AcceleratorNumber
from configuration file
[error] SalCtrl_ServiceEventInit() - : Failed to initialise all crypto
instances
ADF_PROXY err: adf_user_subsystemInit: Failed to initialise Subservice
SAL
[error] SalCtrl_ServiceEventStart() - : Private data is NULL
ADF_PROXY err: adf_user_subsystemStart: Failed to start Subservice SAL
```

10.1.3 Test Results

When running the application in kernel space, open a second terminal window, log in as root, and issue the following command:

```
# tail -f /var/log/messages
```

When running the application in user space, the results are printed to the terminal window in which the application is launched.

Here is an example of the log messages created during the test:

```
--------------------------------------
Algorithm Chaining - AES256-CBC HMAC-SHA512
Number of threads 2
Total Submissions 20
Total Responses 20
Packet Size 512
--------------------------------------
```

A similar pattern is repeated for each of the tests.

Note: The sample code will log the error messages shown below. They can be safely ignored, as the SoC does not support data compression.

```
main():1041 Unable to Get Number of DC instances

Sample Code Complete
```

10.1.4 Unloading the Sample Code

Once the kernel space sample code test has completed, the message `Sample Code Complete` is displayed. The module can then be unloaded using the following command:

```
# rmmod cpa_sample_code.ko
```

Once the user space sample code test has completed, the kernel memory driver `qaeMemDrv.ko` can be unloaded using the following command:

```
# rmmod qaeMemDrv.ko
```

10.2 Acceleration Functional Sample Code

The software package contains a set of sample tests that exercises acceleration functionality. This section describes the steps required to build and execute the sample tests.

The sample application is provided for both Kernel Space and User Space and the following sections contain instructions for both.

Note: The memory driver included with the sample application is a sample memory driver and is not intended for actual deployment.

10.2.1 Compiling the Acceleration Functional Sample Code

The acceleration functional sample code can be compiled manually.

Note: These instructions assume the software package has been untarred to the /QAT directory and that the kernel source files were placed in the directory specified in this guide.

1. The following environment variable must be set to build the modules:
   ```
   export ICP_ROOT=<QATdir>
   ```
   Note: Where <QATdir> is /QAT/QAT1.6 or /QAT/QAT1.5, depending on what hardware support is required. If building with mux support, always use the QAT1.6 sample code.
2. The sample code is compiled with the default assumption that the kernel source header files are located in one of the following directories:

- For 64-bit: /usr/src/kernels/3.1.0-7.fc16.x86_64
- For 32-bit: /usr/src/kernels/3.1.0-7.fc16.i686

3. If the kernel source header files are located in a different directory, create the environment variable with the directory of desired target kernel sources. For example:
   ```
   # export KERNEL_SOURCE_ROOT=/usr/src/kernels/`uname -r`
   ```
   For Yocto:
   ```
   #export KERNEL_SOURCE_ROOT=/usr/src/kernel
   ```
4. You can compile for both Kernel space and User space at the same time using the following commands:
   ```
   # cd $ICP_ROOT/quickassist/lookaside/access_layer/src/sample_code/functional
   # make all
   ```

The generated Linux kernel objects and sample applications are located at:
$ICP_ROOT/quickassist/lookaside/access_layer/src/sample_code/functional/build

10.2.2 Executing the Acceleration Functional Sample Code in Kernel Space

To execute the acceleration functional sample code in Kernel space, enter the commands:

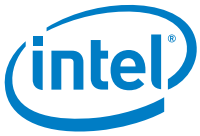
```
# export ICP_ROOT=/QAT/QAT1.5
# cd $ICP_ROOT/quickassist/lookaside/access_layer/src/sample_code/functional/
build
# insmod ./nrbg_sample.ko
```

Note: nrbg_sample.ko is one of the functional kernel modules. You can launch the other .ko modules in a similar fashion.

You may observe an error message similar to the following when submitting the insmod command:

```
insmod: error inserting './algchaining_sample.ko': -1 Resource temporarily
unavailable
```

This error can be safely ignored. When the test application is completed, the kernel object is removed which causes this error. Test results for the application are available in /var/log/messages.



10.2.3 Executing the Acceleration Functional Sample Code in User Space

To execute the acceleration functional sample code in User Space, the kernel memory driver `qaeMemDrv.ko` must be installed. See Section 10.1, "Intel® QuickAssist Technology Acceleration Sample Application" on page 67 for information on compiling the performance sample code. The `qaeMemDrv.ko` kernel object is built as part of that sample code.

1. Install the kernel memory driver `qaeMemDrv.ko` as follows:

   ```
   # insmod $ICP_ROOT/quickassist/lookaside/access_layer/src/sample_code/build/
   qaeMemDrv.ko
   ```

2. To execute the acceleration functional sample code in user space, use the following commands:

   ```
   #cd $ICP_ROOT/quickassist/lookaside/access_layer/src/sample_code/functional/
   build
   #./nrbg_sample
   ```

   Note: `nrbg_sample` is one of the functional user space applications. You can launch the other user space applications in a similar fashion.

A Intel® Atom™ Processor C2000 Product Family BIOS Post/Error Codes

A.1 Post Codes

A.1.1 SEC Phase

Table A-1. SEC Phase

Post Code	Description
0x01	BSP Protected Mode Start
0x02	Status_Code
0x03	Platform Initialization – Microcode Loaded
0x04	WA to enable BIOS_REST_DONE
0x09	Initialize NEM
0x0A	Establish Stack
0x0B	Pei Core Entry Point
0x0C	Jump to PEI Core

A.2 DXE Phase

Table A-2. DXE Phase

Post Code	Description
0xC4	DXE Detect
0xC7	DXE Apply BIOS Knobs
0xC8	Update BIOS Repository
0xCD	Final post code, buy with efi shell

A.3 Memory Reference Codes (MRC) Post Codes

Table A-3. MRC Post Codes (Sheet 1 of 2)

Post Code	Description
0x10	Enable HPET
0x11	Update Variables
0x12	Enable Clock Gating
0x13	Clear Self Refresh
0x14	Oem Track Init Complete
0x42	Prog Ddr Timing Control
0x43	Prog Bunit
0x44	Prog Bl Mode
0x50	Handle Ddrio Phy Init
0xA0	MMRC SFR Vol Sel
0xA1	MMRC PLL Init
0xA2	MMRC DDR Static Init 2
0xA3	MMRC DDR Static Init Perf
0xA4	MMRC DDR Static Pwr Clk Gating
0xA7	MMRC DLL Init
0xA8	MMRC Comp Init 1
0xAA	MMRC Comp Init 2
0xAB	MMRC Periodic Comp Init
0xAC	MMRC HMC Init
0xAD	MMRC Wr Pointer Init
0xAE	MMRC IO BUF ACT Init
0xAF	MMRC Pre Jedec Init
0xB0	MMRC DDR3 Reset
0x61	Prog Dra Drb
0x62	Prog Memory Mapping Register
0x71	Perform D-unit Wake
0x81	Pre Jedec Init
0x82	Perform Jedec Init
0x83	Post Jedec Init
0x90	Disconnect BD
0x91	Disable Pmi
0x92	Disable B-unit Cache
0x92	Set DDR Initialization Complete
0xB0	Handle Rank Overrides... Rank to rank switching enabled
0xB1	Enable Diffamp And Odt Overrides
0xB2	Early Set Write Vref... Vref Set: 0x40
0xB3	Disconnect BD
0xB4	Enable Pmi

Table A-3. MRC Post Codes (Sheet 2 of 2)

Post Code	Description
0xB5	Handle Memory Training
0xB0	Receive Enable. Receive Enable Entry Hooks
0xB2	Early Mpr Read. Early Mpr Read Training Entry Hooks
0xB3	Fine Write Leveling. Fine Write Leveling Entry Hooks. Training Entry
0xB4	Coarse Write Leveling Entry Hooks
0xB6	Read Vref Entry Hooks
0xB7	Read Training Entry Hooks
0xB9	Write Vref Entry Hooks. Max Vref Center.
0xBA	Set Common Vref
0xBB	Write Training Exit Hooks
0xBC	Command Clock Training
0xBD	Command Clock Restore
0xBE	Performance Setting MMRC Static Init Perf Enable
0xBF	Phy View Table
0xBE	Enable B-unit Cache
0xC0	Reconfigure DR
0xD1	Prog Ddr Control
0xE1	Set Scrambler
0xE2	Set Periodic RComp
0xC0	MMRC Periodic Comp Init
0xE5	Change Refresh Period
0xE6	Disconnect BD
0xE8	Enable Pmi
0xF1	MemoryTest. Rank to rank switching disabled. HW engine testing 4096MB of memory in CH0
0xF2	ScrubMemory. ECC initilization. About to enable ECC on Channel 0.
0xF3	Connect BD
0xF4	Set Init Done.

A.4 MRC Error Codes

Table A-4. MRC Error Codes

Error Code	Description
0xE7	Frequency requested by Dimm or user not supported
0xE8	No memory detected
0xE9	Risers from two different platform mixed, 2SPC\3SPC risers are mixed
0xEA	DDR3 training did not complete successfully
0xEB	Memory test failure
0xEC	SMBUS read\write failure
0xED	UDIMMs and RDIMMs are both present DIMM vendor-specific errors
0xEF	Indicates a CLTT table structure error. A DIMM is populated in the 3rd slot when quad rank DIMM is present in the channel
0xEE	Number of active Home Agent (detected) exceeds the pre-defined number of Home Agent
0xEF	Indicates a CLTT table structure error. A DIMM is populated in the 3rd slot when quad rank DIMM is present in the channel

§ §

B BIOS Update Using an SPI Programmer

This section provides details for programming the Flash memory containing the BIOS on the Mohon Peak CRB. The following example was performed using a laptop running Windows* 7 Enterprise* installed, using an in-circuit SPI Flash programmer (DediProg* SF600).

Note: Any SPI Flash programmer, computer, and software may be used to program the Flash memory. The following specific example is presented in order to describe the programming process.

B.1 Tools Required

- Windows* PC with an available USB port
- DediProg* SF600 SPI Flash Programmer
  http://www.dediprog.com/SPI-Flash-Programmer/SF600
- SO Connection Adapter, model EM-CON-SO
  http://www.dediprog.com/SPI-Serial-Flash-Emulator/EM100-SO-Connection-Adaptor
- 1.27 mm, 2x4 Programming Cable, model CB-24-127
  http://www.dediprog.com/SPI-Serial-Flash-Emulator/Female-Header-2x4-Cable

Note: If you happen to be using an older SF100 programmer instead of the SF600, acquire the SF100 Universal Adapter, model ISP-UADP-127/254:
(http://dediprog.com/SPI-flash-in-circuit-programming/SF100-Universal-Adaptor)

Figure B-1. DediProg SF600 Programmer and Adapter

B.2 Software Required

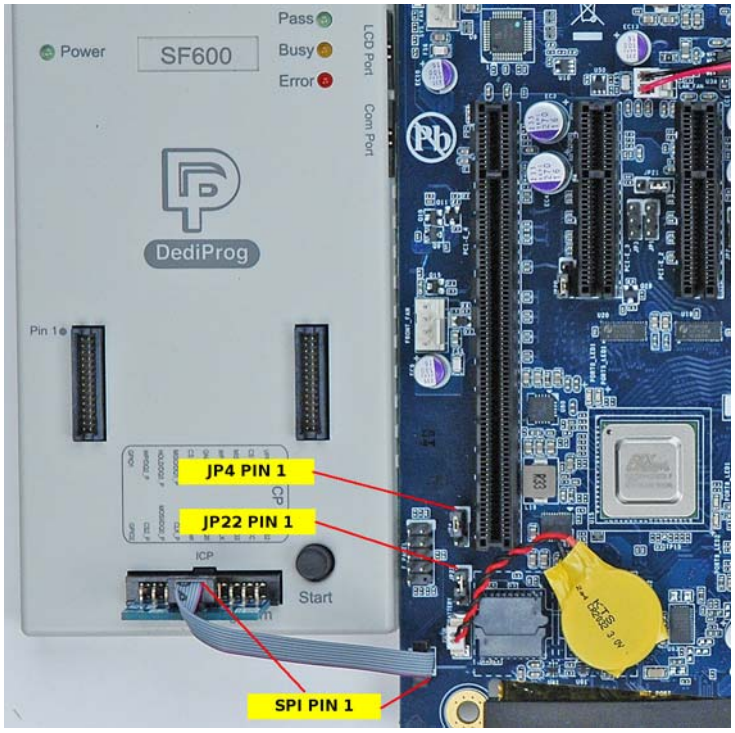
Download and install the SF600 installation software on the Windows PC (this example uses `SF6.0.4.04.zip`).

http://www.dediprog.com/programmer_tools.php

B.3 Setup

1. Plug the programming cable into the SO connection adapter, making sure to align the striped edge of the cable with pin 1 of the connector.
2. Plug the adapter with cable into the SF600 programmer, aligning the bump-out on the adapter connector with the slot in the SF600 programmer ICP connector.
3. With power disconnected from the Mohon Peak CRB, plug the free end of the SF600 programming cable into the SPI_EMUL header on the Mohon Peak CRB. Make sure to align the striped edge of the cable with pin 1 of the header.
4. To program the BIOS1 Flash component, place the jumper on JP22 to connect pins 2-3 and the jumper on JP4 to connect pins 1-2 (see Table 8, “Mohon Peak Jumpers” on page 39).
5. To program the BIOS2 Flash component, place the jumper on JP22 to connect pins 2-3 and the jumper on JP4 to connect pins 2-3 (see Table 8, “Mohon Peak Jumpers” on page 39).

Figure B-2. DediProg SF600 Connected to the Mohon Peak CRB



B.4 Programming

With the SF600 programmer connected as described in Section B.3, plug the system (ATX) power supply in and turn the power supply on.

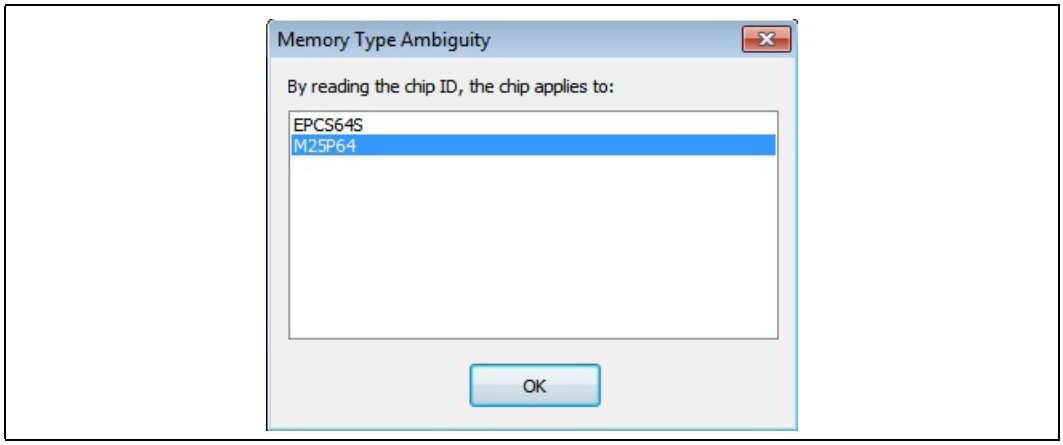
Do not push the power button on the CRB, or the front panel of the chassis (SPI Flash must be programmed with standby power from the system power supply, but without the CRB actually running.)

Start the DediProg Engineering software by double-clicking the desktop icon on the PC.

Figure B-3. DediProg SF600 Engineering Desktop Icon

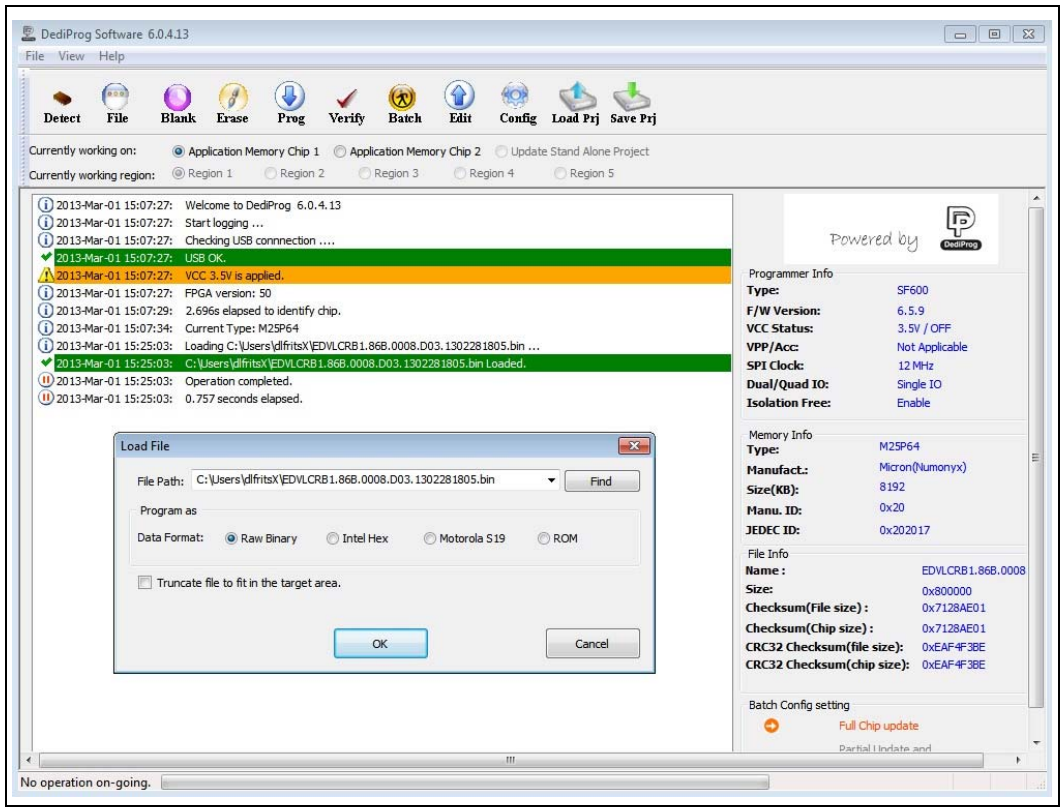
6. Select **M25P64** in the **Memory Type** pop-up window and then click **OK**.

Figure B-4. DediProg Memory Type Selection



7. In the main **DediProg Software** window that appears, click the **File** icon in the tool bar at the top, and select the BIOS firmware file in the selector pop-up window.

Figure B-5. DediProg BIOS File Loading



8. Click the **Erase** icon in the tool bar to reset the contents of Flash memory for programming.

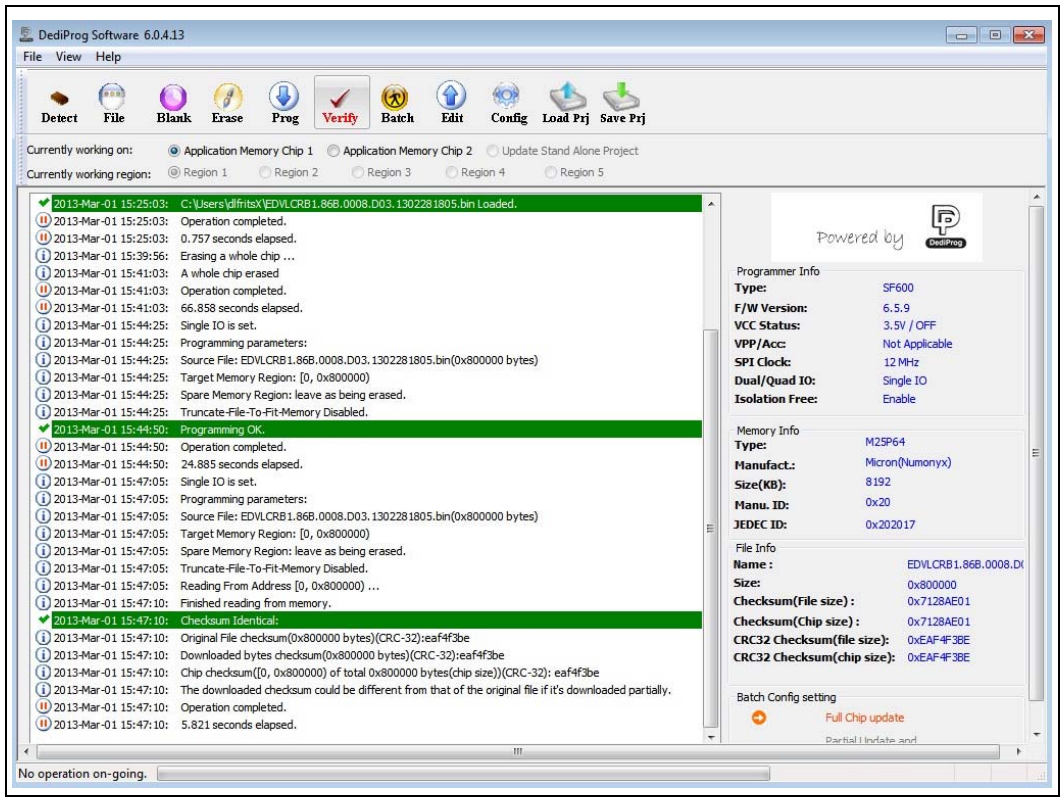
Figure B-6. DediProg Erase Flash

9. Click the **Prog** icon in the tool bar to actually put the BIOS into Flash.

Figure B-7. DediProg Program Flash

10. Click the **Verify** icon to make sure the contents of Flash memory were programmed correctly.

Figure B-8. DediProg Verify Flash



11. Turn the power to the Mohon Peak CRB off.
12. Exit the DediProg software on the PC, disconnect the SF600 programmer from the Mohon Peak CRB, and change jumper JP22 to connect pins 1-2.
13. Power the board on to configure and use the new BIOS.

§ §

C BIOS Update Using Bootable Software

If a BIOS update becomes available, you can update the BIOS image on the CRB with a utility from Intel called the Flash Programming Tool (`fpt`). This utility can be run from the BIOS UEFI Shell.

Note: Prior to using the FPT, confirm that the BIOS is configured so it can be updated via this tool. Ensure that the option **EDKII Menu -> Security -> Relax Security Configuration** is set to "Enabled."

If, however, the system does not have a working BIOS, a hardware method of reprogramming the Flash memory must be used. Either an in-circuit SPI bus programmer (described in Appendix B, "BIOS Update Using an SPI Programmer") or an external programmer and the "BIOS1" Flash part removed from the CRB socket must be used. (See Table 7, "BIOS Flash Configuration Chart" on page 34. Move JP4 to select the desired Flash part to program. Leave JP22 set to connect pins 1-2.)

The following steps describe how to perform a BIOS update using the currently installed BIOS. See Table 7, "BIOS Flash Configuration Chart" on page 34. Move JP4 to select the desired Flash part to program. Leave JP22 set to connect pins 1-2.

1. Download the Flash Programming Tool from the Intel Business Portal..
2. Unzip the package and copy the files below to the root directory of a USB Flash drive.
   ```
   Fpt.efi
   fparts.txt
   ```
3. Add the new BIOS file to the USB Flash drive.
4. Power off the CRB, then plug in the USB Flash drive.
5. Power the board on and stop the UEFI boot process in the UEFI Shell.
6. Enter `map -r` to show the available drives. In most cases, the USB Flash drive will be `fs0`. If so, enter `fs0:` to change to the USB Flash drive.
7. Use the following command to update the BIOS:
   ```
   Fpt.efi -f <BIOSfilename.bin> -bios -p fparts.txt
   ```
8. After programming is complete, power the board "off", remove the USB Flash drive, and then power the board on to configure and use the new BIOS.

§ §