

Intel® EP80579 Software for Security Applications on Intel® QuickAssist Technology for Linux*

Package Version 1.0.2

Getting Started Guide

November 2009



INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. Intel products are not intended for use in medical, life saving, life sustaining, critical control or safety systems, or in nuclear facility applications.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or by visiting [Intel's Web Site](http://www.intel.com).

Any software source code reprinted in this document is furnished under a software license and may only be used or copied in accordance with the terms of that license.

Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families. See http://www.intel.com/products/processor_number for details.

BunnyPeople, Celeron, Celeron Inside, Centrino, Centrino logo, Core Inside, FlashFile, i960, InstantIP, Intel, Intel logo, Intel386, Intel486, Intel740, IntelDX2, IntelDX4, IntelSX2, Intel Core, Intel Inside, Intel Inside logo, Intel. Leap ahead., Intel. Leap ahead. logo, Intel NetBurst, Intel NetMerge, Intel NetStructure, Intel SingleDriver, Intel SpeedStep, Intel StrataFlash, Intel Viiv, Intel vPro, Intel XScale, Itanium, Itanium Inside, MCS, MMX, Oplus, OverDrive, PDCharm, Pentium, Pentium Inside, skool, Sound Mark, The Journey Inside, VTune, Xeon, and Xeon Inside are trademarks of Intel Corporation in the U.S. and other countries.

*Other names and brands may be claimed as the property of others.

Copyright © 2009, Intel Corporation. All rights reserved.



Contents

1.0	Introduction	7
1.1	About this Manual	7
1.2	Additional Information on Software	7
1.2.1	Where to Find Current Software and Documentation	7
1.2.2	Product Documentation	7
1.2.3	Pre-Boot Firmware	8
1.3	Related Software and Documentation	8
1.3.1	Open Source Software and Patches Required	8
1.3.2	Intel® EP80579 Integrated Processor Software Shims	8
1.4	Conventions	9
1.5	Software Overview	9
1.5.1	Features Implemented	9
1.5.2	List of Files in Release	10
1.5.3	Package Release Structure	10
2.0	Configuration Requirements	12
2.1	Development Board Configuration	12
2.1.1	Package Components	12
2.1.2	Development Kit Setup	12
2.1.3	Safety	13
2.1.4	Connecting the Serial ATA Hard Drive and Cable	13
2.1.5	Connecting the Keyboard and Mouse	13
2.1.6	Connecting the PCI Express* Video Card	13
2.1.7	Connecting the Serial ATA DVD-ROM Drive (Optional)	14
2.1.8	Connecting the Power Cables	14
2.1.9	Powering Up the System	14
2.2	Development Board Setup Requirements	19
3.0	Installing the OS on a Development Board	20
3.1	System Requirements	20
3.2	Acquiring CentOS 5.2 Linux	20
3.3	Installing CentOS 5.2 Linux	20
3.3.1	Recommended Installation Customizations	21
3.4	Installing CentOS 5.2 Kernel Source	22
3.5	Unpacking the EP80579 Security Software Linux Package	24
3.6	Updating PCI Device Names	25
3.7	Modifications for Use with OCF (Optional)	25
3.7.1	Applying the OCF Patch	25
3.7.2	Configuring OCF Parameters	26
3.7.3	Backporting OCF Features and Bug Fixes	27
3.8	Rebuilding the Kernel	27
3.8.1	Restore the Old Configuration	27
3.8.2	Modifying the Makefile	27
3.8.3	Building and Installing the Patched Kernel Source	27
3.8.4	Making the New Kernel Default	28
3.8.5	Rebooting and Verifying	28
4.0	Building EP80579 Security Software on a Target Development Board	29
4.1	Environment Setup	29
4.2	Build Options	29
4.3	Build Using Top Level Make	30
4.4	Installation of Build Output	31
4.4.1	Statistics Collection	31



4.5	Uninstalling Embedded and Security Kernel Modules	33
5.0	Runtime Configuration	34
5.1	Loading OCF and OCF Shim	34
5.2	Using the Intel® QuickAssist Technology Cryptographic API	34
6.0	Build and Install Openswan	35
6.1	Environment Setup	35
6.2	Building and Installing the GMP Library	35
6.3	Building and Installing OpenSSL	36
6.4	Building and Installing Openswan	38
6.4.1	Prepare to Build Openswan	38
6.4.2	Building and Installing Openswan	38
7.0	Configuring Sample VPN Application	40
7.1	Configure IPsec on Both Gateways	40
7.1.1	Testing if IPsec is Configured Correctly	41
7.2	Set Up a Tunnel Between Gateways	42
7.3	Configuring Crypto Parameters in ipsec.conf	42
7.3.1	RSA SIG	43
7.3.2	PSK	44
7.4	Installing the OS and Openswan on Gateway GW2	45
8.0	Building Components Individually	47
8.1	Building Components Individually Using Top-Level Make	47
9.0	Building, Installing and Loading Individual EP80579 Embedded Software Drivers ...	48
9.1	Controller Area Network (CAN) Driver	48
9.1.1	Linux Compilation Instructions	48
9.1.2	Linux Module Load/Unload Instructions	48
9.1.3	Linux Sample Codelet	49
9.2	Enhanced Direct Memory Access (EDMA) Driver	49
9.2.1	Linux Compilation Instructions	49
9.2.2	Linux Module Load/Unload Instructions	49
9.2.3	Runtime Configuration of EDMA	50
9.2.4	Linux Sample Codelet	50
9.3	Watchdog Timer (WDT) Driver	50
9.3.1	Linux Compilation Instructions	50
9.3.2	Linux Module Load/Unload Instructions	51
9.3.3	Runtime Configuration of WDT	51
9.3.4	Linux Sample Codelet	51
9.4	General Purpose I/O (GPIO) Driver	52
9.4.1	Linux Compilation Instructions	52
9.4.2	Linux Module Load/Unload Instructions	52
9.5	IEEE 1588 Hardware Assist Driver	52
9.5.1	Linux Compilation Instructions	52
9.5.2	Linux Module Load/Unload Instructions	53
9.5.3	Runtime Configuration of IEEE 1588	53
9.5.4	Linux Sample Codelet	53
9.6	Global Configuration Unit and Gigabit Ethernet Drivers	54
9.6.1	Linux Compilation Instructions	54
9.6.2	Linux Module Load/Unload Instructions	54
9.6.3	Runtime Configuration of GCU and Gigabit Ethernet	54
9.7	System Management Bus (SMBus) Driver	55
9.7.1	Linux Compilation Instructions	55
9.7.2	Linux Module Load/Unload Instructions	55
9.7.3	Runtime Configuration of SMBus	56



9.8	CompactFlash* Driver	56
9.8.1	Linux Compilation Instructions	56
9.8.2	Linux Module Load/Unload Instructions	56
9.8.3	Linux Runtime Configuration of CompactFlash Driver.....	57
10.0	Building and Using Crypto Tools	58
10.1	Building Crypto Tools	58
10.2	Using Crypto Tools	59
11.0	Booting from CompactFlash*	60
11.1	System Requirements	60
11.2	Procedure	60
12.0	Sample Applications	64
13.0	Pre-Boot Firmware (BIOS)	65
13.1	Pre-Boot Firmware Setup Menu	65
13.1.1	Serial Console Redirection	66
13.1.2	Changing the Boot Device	66
13.1.3	Maximum Memory Speed Setup	66
13.1.4	Coherent and Non-Coherent Memory Allocation	67
13.1.5	Legacy and AHCI SATA Mode	67
13.2	Pre-Boot Firmware Image Reflashing Instructions	68
13.2.1	Aptio Flash Update Utility (AFUEFI)	68
14.0	Uninstalling the Software	69
14.1	Linux Module/Driver Dependencies	69
15.0	Troubleshooting	70
15.1	Using a Graphics Card.....	70
15.2	Using the Serial Port.....	70
16.0	Glossary	72

Figures

1	Development Board - Top View of Component and Connector Locations.....	15
2	Development Board - Bottom View of Component and Connector Locations.....	16
3	Development Board - Side View of the Board Connectors.....	17
4	Development Board System Setup	19
5	VPN Example.....	40

Tables

1	Development Board - Key Components and Connectors Legend.....	17
2	Output Files Created in the \$ICP_BUILD_OUTPUT Directory	31
3	Pre-Boot Firmware Setup Main Menu	65
4	Pre-Boot Firmware Setup Program Action Keys	65
5	Serial Console Redirection Default Settings.....	66
6	Linux Module/Driver Dependencies	69



Revision History

Date	Revision	Description
November 2009	001	<p>This document is derived from document number 320182-004 prepared for the 1.0.2 release. Differences between the 1.0.2 version and this version are:</p> <ul style="list-style-type: none">• Added CompactFlash driver information in Section 9.8 and Section 11.0.• Other updates are noted with changebars.

§ §



1.0 Introduction

1.1 About this Manual

This Getting Started Guide documents the instructions to obtain, build (if necessary), install, and execute the Intel® EP80579 Software for Security Applications on Intel® QuickAssist Technology release package. Additionally, this document describes some brief instructions on configuring the Intel® EP80579 Integrated Processor with Intel® QuickAssist Technology Development Board.

Note: In this document, for convenience:

- “Intel® EP80579 Integrated Processor with Intel® QuickAssist Technology Development Board” is referred to as the “development board”
- “Intel® EP80579 Software for Security Applications on Intel® QuickAssist Technology” is referred to as the “EP80579 security software”

Note: The release package includes a directory with sample code.

1.2 Additional Information on Software

The EP80579 security software release package for Linux has been validated with CentOS 5.2 Linux.

1.2.1 Where to Find Current Software and Documentation

The software release and associated collateral can be found on the Hardware Design resource center.

1. In a web browser, go to <http://www.intel.com/go/soc>.
2. For software and pre-boot firmware: Click on “Tools & Software” tab.
3. For documentation: Click on “Technical Documents” tab.

Note: The EP80579 security software release package contains encryption software and is subject to export requirements defined by the U.S Department of Commerce. To satisfy these requirements, the End User Certification Form must be filled out and submitted for review/approval. Instructions on this process are included during the download process. Please note that this process may take up to two business days to complete.

1.2.2 Product Documentation

The following documentation supporting this release can be accessed as described in [Section 1.2.1](#):

- Intel® EP80579 Software for Security Applications on Intel® QuickAssist Technology for Linux* Getting Started Guide (this document)
- Intel® EP80579 Software for Security Applications on Intel® QuickAssist Technology Release Notes



- Intel® EP80579 Software for Security Applications on Intel® QuickAssist Technology Programmer's Guide
- Intel® EP80579 Software for Security Applications on Intel® QuickAssist Technology Cryptographic API Reference Manual
- Intel® EP80579 Software on Intel® QuickAssist Technology Debug Services API Reference Manual
- Intel® EP80579 Software Drivers for Embedded Applications Programmer's Guide and API Reference Manual

1.2.3 Pre-Boot Firmware

The latest release of the development board pre-boot firmware (BIOS) is also located on the Hardware Design resource center.

Please refer to Release Notes listed in [Section 1.2.1](#) for the correct firmware version.

1.3 Related Software and Documentation

Refer to the Intel® EP80579 Integrated Processor with Intel® QuickAssist Technology Development Kit User's Guide for 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 1.2.1](#) to locate this collateral.

1.3.1 Open Source Software and Patches Required

Depending on your applications, the following table shows required open source software and patches.

Item	URL
OCF patch for Linux*	http://heanet.dl.sourceforge.net/sourceforge/ocf-linux/ocf-linux-26-20071215.patch.gz
OCF patch for OpenSSL*	http://heanet.dl.sourceforge.net/sourceforge/ocf-linux/ocf-linux-20080917.tar.gz
GNU Multiple Precision Arithmetic library	ftp://ftp.sunet.se/pub/gnu/gmp/gmp-4.2.1.tar.gz
Crypto Tools	http://heanet.dl.sourceforge.net/sourceforge/ocf-linux/crypto-tools-20071215.tar.gz
Openswan	http://www.openswan.org/download/old/openswan-2.4/openswan-2.4.9.tar.gz
Openswan Patch	http://heanet.dl.sourceforge.net/sourceforge/ocf-linux/ocf-openswan-2.4.9-20070727.patch.gz
CentOS* Linux Kernel	http://vault.centos.org/5.2/os/SRPMS/ (kernel-2.6.18-92.el5.src.rpm)
Open SSL	http://www.openssl.org/source/openssl-0.9.8g.tar.gz

1.3.2 Intel® EP80579 Integrated Processor Software Shims

The software listed in this section contains sample source code provided "As-Is" without warranty of any kind. This software can be found in the Acceleration/shims folder of the software package provided by Intel:

- OCF Shim



1.4 Conventions

The following conventions are used in this manual:

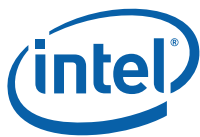
- `Courier font` - commands and code examples

1.5 Software Overview

1.5.1 Features Implemented

The software provides the following features:

- Gigabit Ethernet (GbE) Controller Driver for Network Connectivity
- Support for Symmetric Cryptography (Synchronous and Asynchronous modes)
 - Ciphers:
 - NULL Cipher
 - DES (ECB, CBC),
 - 3DES (ECB, CBC, CTR)
 - AES (ECB, CBC, CTR, CCM, GCM)
 - ARC4
 - Hash Algorithms:
 - MD5
 - SHA-1
 - SHA-2 (224, 256, 384, 512)
 - Nested Hashing
 - Authentication Schemes:
 - AES-XCBC-MAC-96
 - HMAC-MD-5
 - HMAC-SHA-1
 - HMAC-SHA-2 (224, 256, 384, 512)
 - Chaining Cipher and Authentication algorithms
- Support for Public Key Cryptography (Synchronous and Asynchronous modes)
 - Large Number Modular Exponentiation and Inversion
 - RSA (Key Generation, Encrypt, Decrypt)
 - DSA (Parameter Generation and Signatures)
 - DH (Phase1 and Phase2 secret key generation)
 - SSL/TLS Key and Mask generation
 - True Random Number Generation (RNG)
 - Prime Number Tests
- Support for OCF Shim (a kernel module that communicates with OCF at the top layer and the Intel® EP80579 Software for Security Applications on Intel® QuickAssist Technology Cryptographic API at the bottom layer and thus enabling OCF to use EP80579 Crypto Accelerators). The following features have been validated:
 - Null Cipher
 - DES-CBC, 3DES-CBC, AES-CBC, ARC4
 - SHA1, SHA256, SHA384, SHA512, MD5 and HMAC of the same
 - Modular Exponentiation and Chinese Remainder Theorem
 - DSA RS Sign and Verify



- Diffie-Hellman secret key generation
- Daemon to supply random numbers to kernel /dev/random pool
- IEEE 1588 Hardware Assist Driver
- Controller Area Network (CAN) Hardware Access Driver
- Advanced Host Controller Interface Software Support for SATA for Native Command Queuing and Hot Plug Capability
- SMBus Driver
- General Purpose I/O (GPIO) Hardware Access Driver
- Enhanced Direct Memory Access (EDMA) Hardware Assist Driver
- Watchdog Timer Hardware (WDT) Access Driver
- CompactFlash Driver

1.5.2 List of Files in Release

The Bill of Materials, sometimes referred to as the BOM, is included as a text file in the released software package. This text file is labeled “filelist” and is located at the top directory level for each release.

1.5.3 Package Release Structure

The high-level package release directory structure is shown as follows:

Note: The sample code is contained in the ./Acceleration/library/icp_crypto/look_aside_crypto/src/sample_code/ directory.

```
./Acceleration
./Acceleration/drivers
./Acceleration/drivers/icp_asd
./Acceleration/firmware
./Acceleration/include
./Acceleration/include/dcc
./Acceleration/include/lac
./Acceleration/library
./Acceleration/library/common
./Acceleration/library/common/include
./Acceleration/library/icp_crypto
./Acceleration/library/icp_crypto/look_aside_crypto
./Acceleration/library/icp_crypto/look_aside_crypto/include
./Acceleration/library/icp_crypto/look_aside_crypto/src
./Acceleration/library/icp_crypto/look_aside_crypto/src/common
./Acceleration/library/icp_crypto/look_aside_crypto/src/linux
./Acceleration/library/icp_crypto/look_aside_crypto/src/sample_code
./Acceleration/library/icp_crypto/QATAL
./Acceleration/library/icp_crypto/QAT_FW
./Acceleration/library/icp_debug
./Acceleration/library/icp_debugmgmt
./Acceleration/library/icp_debugmgmt/MIL
./Acceleration/library/icp_services
./Acceleration/library/icp_services/ResourceManager
./Acceleration/library/icp_services/RuntimeTargetLibrary
./Acceleration/library/icp_utils
./Acceleration/library/icp_utils/OSAL
./Acceleration/shims
./Acceleration/shims/OCF_Shim
./build_system
./Embedded
./Embedded/src
./Embedded/src/1588
```



```
./Embedded/src/CAN
./Embedded/src/CF
./Embedded/src/EDMA
./Embedded/src/GbE
./Embedded/src/GPIO
./Embedded/src/WDT
./Embedded/codelet
./Embedded/codelet/1588
./Embedded/codelet/CAN
./Embedded/codelet/EDMA
./Embedded/codelet/WDT
./OpenSourcePatches
./OpenswanConfigFiles
```



2.0 Configuration Requirements

2.1 Development Board Configuration

The Intel® EP80579 Integrated Processor with Intel® QuickAssist Technology Development Kit User's Guide contains complete details on the development board. The document describes the design, structure, and function of all development board features.

To facilitate quick start of the Software for Intel® EP80579 Integrated Processor product line, relevant sections from the development kit User's Guide have been included in this chapter. Please follow the directions in [Section 1.2.1](#) to access the full User's Guide.

2.1.1 Package Components

The development kit includes the following:

- A development board containing the Intel® EP80579 Integrated Processor with Intel® QuickAssist Technology
- ATX12V power supply
- One DDR2-800 DIMM
- PCIe* graphics card
- SATA hard drive with cable
- SATA DVD-ROM with cable
- Two Controller Area Network cable connectors
- Power Cord (USA power cord supplied)

The following items are required, but not supplied by Intel:

- Mouse
- Keyboard
- Monitor
- Power Cord (if country or region-specific power cord is required)

Note: Additional items may be required but are not supplied by Intel.

2.1.2 Development Kit Setup

Ensure that all components listed in [Section 2.1.1](#) arrive together. Once all components have been identified and located, installation and setup can begin. This section describes how to set up the development board for operation.

Note: This document assumes that the user is familiar with the basic concepts required to install and configure hardware for a PC system.



2.1.3 Safety

The development board is shipped as an open system allowing for maximum flexibility in changing hardware configurations and peripherals in a lab environment. Since the board is not in a protective chassis, the user is required to take safety precautions when handling and operating the board. Some assembly is required before use.

Ensure a safe and static-free work environment before removing any components from their anti-static packaging. The development board is susceptible to electrostatic discharge that may cause failure or unpredictable operation. The development board must be operated on a flame-retardant surface because a chassis is not included with the board.

Caution: Connecting the wrong cable or reversing a cable may damage the board and may damage the device being connected. Since the board is not in a protective chassis, use caution when connecting cables to the board.

Caution: The power supply cord provides the main connection to AC power. The socket outlet should be installed near the equipment and should be readily accessible. To avoid shock, ensure that the power cord is connected to a properly-wired and grounded receptacle. Do not connect/disconnect any cables or perform installation/maintenance of the boards in this product during an electrical storm. Ensure that any equipment to which this product will be attached is also connected to properly-wired and grounded receptacles.

Note: Ensure that the step to set up the ATX power supply is the final step performed in the process of assembly.

2.1.4 Connecting the Serial ATA Hard Drive and Cable

The development board provides two Serial ATA (SATA) connectors. Connect cables to the appropriate drive sequentially, starting from Port 0 to Port 1. See [Figure 1](#) and [Table 1](#) for the location and identification of the SATA connectors.

Note: Intel recommends connecting the boot drive to SATA port 0.

2.1.5 Connecting the Keyboard and Mouse

Connect a PS/2 mouse and keyboard to the stacked PS/2 connector on the rear panel of the board. The bottom connector is the keyboard connector and the top connector is the mouse connector. Alternatively, a USB keyboard and a USB mouse can be connected to the USB connectors on the development board.

Note: The mouse and keyboard are not supplied by Intel.

Note: The serial redirection feature can be enabled to remotely access the board through a serial cable without attaching a keyboard or mouse to the development board. Refer to the "Connecting the Serial Cable for Console Redirection" section of the Intel® EP80579 Integrated Processor with Intel® QuickAssist Technology Development Kit User's Guide for more information.

2.1.6 Connecting the PCI Express* Video Card

Populate the PCIe* graphics card in any one of the PCIe slots.



2.1.7 Connecting the Serial ATA DVD-ROM Drive (Optional)

Connect the Serial ATA DVD-ROM drive to SATA Port 1 utilizing the cable that comes with the DVD-ROM drive. See [Figure 1](#) and [Table 1](#) for the location and identification of the SATA connectors.

2.1.8 Connecting the Power Cables

Use the following procedure to connect the power cables:

1. The board supports the use of ATX12V power supplies with either 2 x 10 or 2 x 12 main power cables.
2. Plug the main connector into the board. Ensure that the plug clip lines up with the clip lock and the connector pins easily fit into their appropriate slots. When using a power supply with a 2 x 10 main power cable, attach that cable to the right-most part of the main power connector, leaving pins 11, 12, 23 and 24 (labeled on the board) unconnected.
3. Plug in the power connectors from each of the SATA drives.

2.1.9 Powering Up the System

Warning: Ensure the steps in the previous sections were strictly followed before powering up the system.

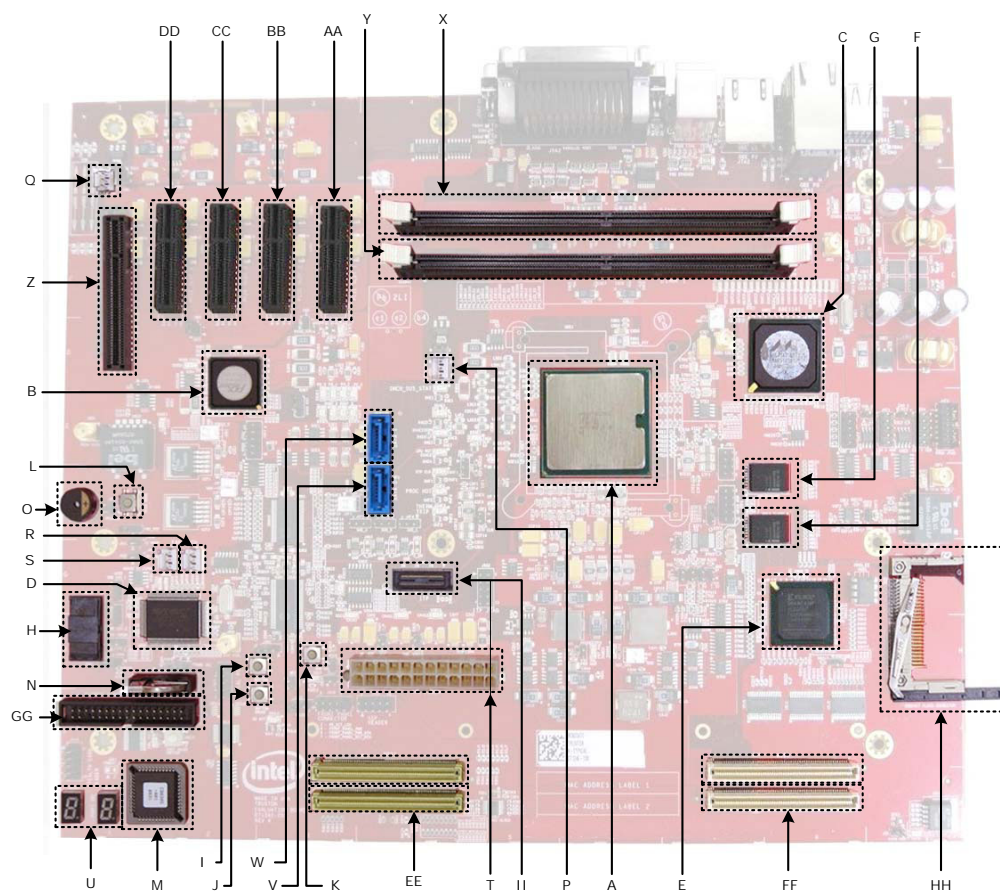
Use the following procedure to power up the development board:

1. Ensure that the processor heat sink and the fan are installed according to the procedure in the “Connecting the Processor Heatsink and Fan” section of the Intel® EP80579 Integrated Processor with Intel® QuickAssist Technology Development Kit User’s Guide.
2. Leaving the On/Off switch in the OFF position, connect the power cable into the back of the power supply.
3. Once the board is set up, plug the cord into the power source.
4. Switch on the power supply.
5. Press the power-on button to start the system. Refer to [Figure 1](#) for the location of the power-on button (item I, lower-left).

Note: [Table 1](#) is a legend for key items labeled in [Figure 1](#), [Figure 2](#) and [Figure 3](#).

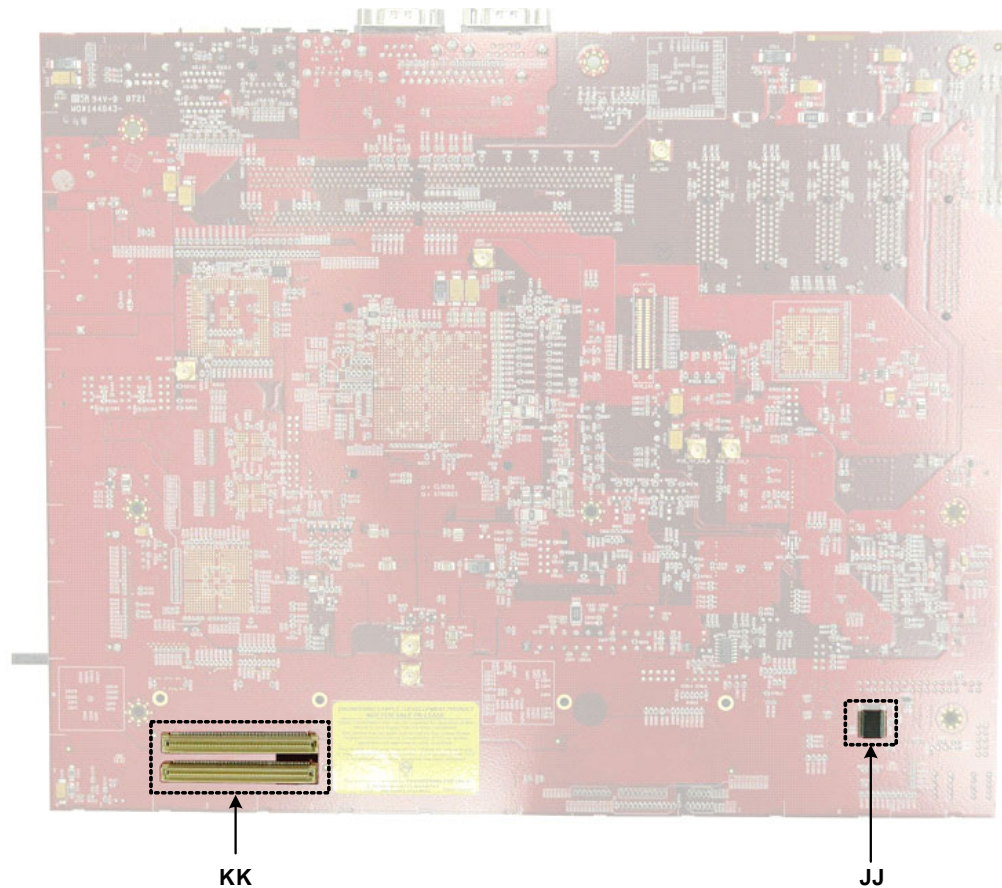


Figure 1. Development Board - Top View of Component and Connector Locations



B6607-02

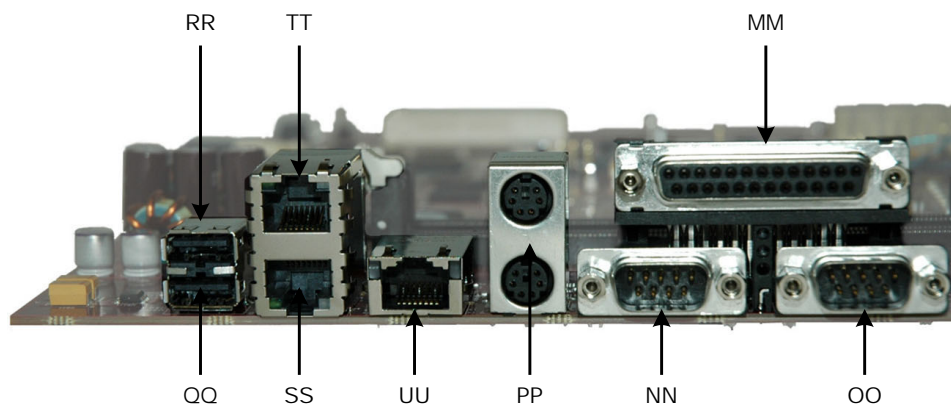
Figure 2. Development Board - Bottom View of Component and Connector Locations



B6606-03



Figure 3. Development Board - Side View of the Board Connectors



B6605-01

Table 1. Development Board - Key Components and Connectors Legend (Sheet 1 of 2)

Callout	Component/Connector
A	Intel® EP80579 Integrated Processor with Intel® QuickAssist Technology
B	PEX PCIe Switch Chip
C	Marvell* 88ME1141 Quad PHY
D	Super IO Controller
E	FPGA
F	Flash memory 0
G	Flash memory 1
H	FWH
I	Power button
J	Reset button
K	Sleep button
L	PCIe Wake button
M	Port 80 IC
N	CMOS battery
O	On-board speaker
P	CPU FAN connector
Q	AUX FAN connector
R	AUX0 FAN connector
S	AUX1 FAN connector
T	ATX power connector
U	Two 7-segment display (Port 80)

**Table 1. Development Board - Key Components and Connectors Legend (Sheet 2 of 2)**

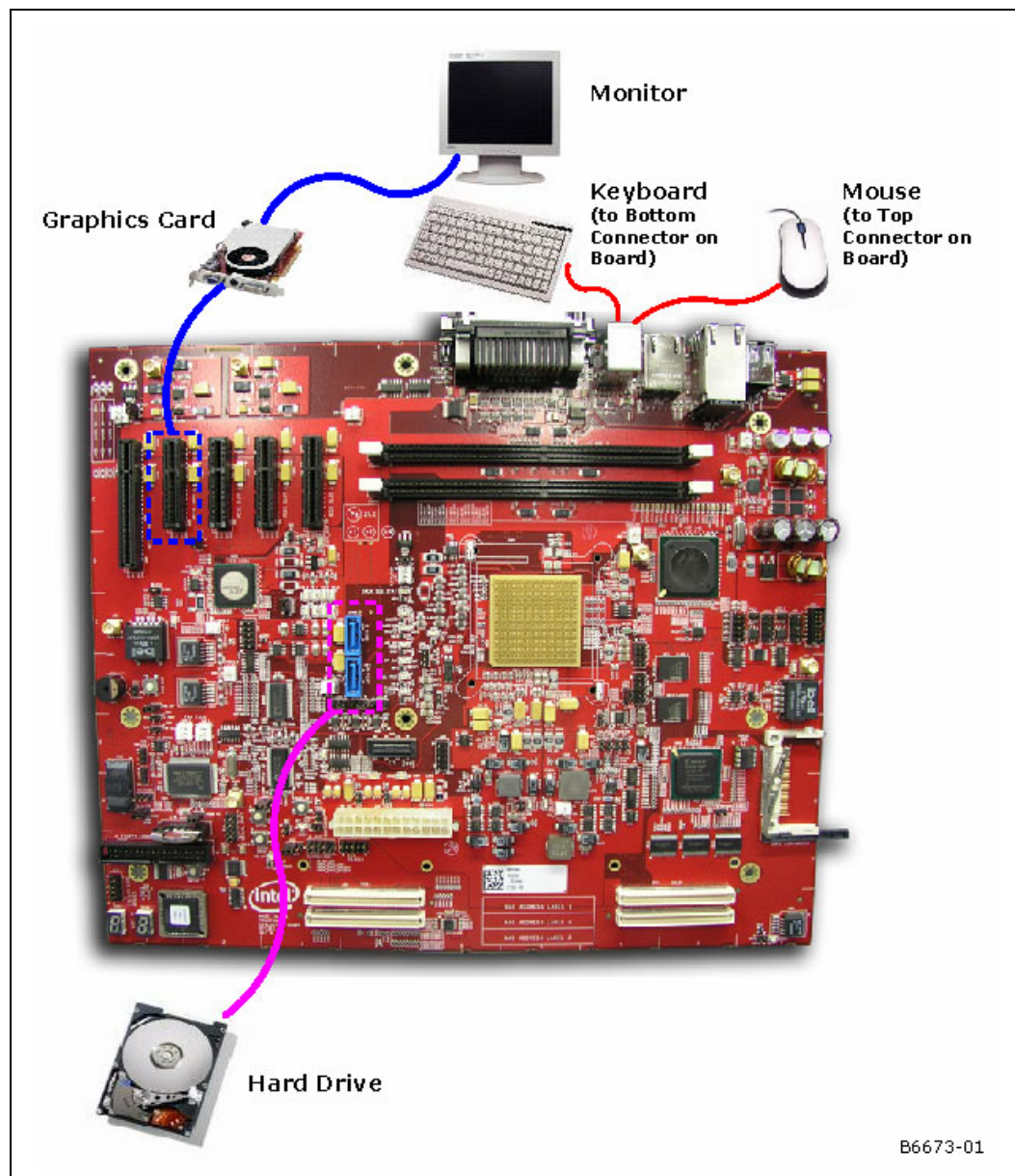
Callout	Component/Connector
V	SATA port 0
W	SATA port 1
X	DDR2 DIMM0
Y	DDR2 DIMM1
Z	Slot 0 x8 connector 4 lanes PCI Express
AA	Slot 1 x4 connector 1 lane PCI Express
BB	Slot 2 x4 connector 1 lane PCI Express
CC	Slot 3 x4 connector 1 lane PCI Express
DD	Slot 4 x4 connector 1 lane PCI Express
EE	Mezzanine connector 1
FF	Mezzanine connector 0
GG	Floppy Connector
HH	CF connector
II	ITP-XDP connector
JJ	Trusted Platform Module
KK	Mezzanine connector 2
MM	Parallel port
NN	COM1
OO	COM2
PP	PS/2 mouse (top)/keyboard (bottom)
QQ	USB port 0
RR	USB port 1
SS	RJ-45 Ethernet port 0
TT	RJ-45 Ethernet port 1
UU	RJ-45 Ethernet port 2



2.2 Development Board Setup Requirements

Figure 4 shows the system setup when the target development board is also used for build and install.

Figure 4. Development Board System Setup





3.0 Installing the OS on a Development Board

Installing the OS on a development board involves the installation of CentOS 5.2 from the CDs and the rebuilding of the kernel and module files after patching the kernel for the following:

- Built-in PCI device recognition
- OpenBSD Cryptographic Framework (OCF) patch

Note: System commands given in this chapter assume that the user is issuing commands from a bash shell. This is the default shell. Use the “echo \$0” command to verify use of the bash shell or run “/bin/bash” to switch to the bash shell.

3.1 System Requirements

Please consult the CentOS minimum hardware requirements in the online Installation Guide at http://www.centos.org/docs/5/html/5.2/Installation_Guide. The development board meets these requirements.

Execute the following three steps:

- Get the latest BIOS image and install it as described in [Section 13.2.1, “Aptio Flash Update Utility \(AFUEFI\)” on page 68](#)
- Set the Legacy or AHCI SATA mode as described in [Section 13.1.5, “Legacy and AHCI SATA Mode” on page 67](#)
- Allocate the Coherent and Non-Coherent Memory as described in [Section 13.1.4, “Coherent and Non-Coherent Memory Allocation” on page 67](#)

Note: The SATA mode (Legacy or AHCI) cannot be changed after OS installation.

3.2 Acquiring CentOS 5.2 Linux

The software package has been validated with CentOS 5.2, and validated with GCC 4.1 and the Glibc 2.5 tool chain. The software package does not include a distribution of CentOS 5.2 or any Linux distribution. The package includes Linux device driver sources developed by Intel for EP80579 integrated processor features. The device driver source may be installed with a CentOS 5.2 distribution. Please acquire a source distribution of CentOS 5.2 Linux from <http://vault.centos.org/5.2/>.

3.3 Installing CentOS 5.2 Linux

Note: You will have the option to install Linux in either text mode or graphical mode. The instructions in this guide are based on selecting graphical mode. Installation instructions may differ if text mode is selected.

For complete CentOS 5.2 installation instructions, please refer to the online Installation Guide at http://www.centos.org/docs/5/html/5.2/Installation_Guide. The following are some basic instructions. For the purposes of this Getting Started Guide, it is assumed that the installation is from CD images.



Warning: Decide to use the SATA drive in Legacy mode or AHCI (Advanced Host Controller Interface) mode prior to installing any Operating System. Once the SATA mode is set and the Operating System is installed, changing the SATA mode will prevent the Operating System from booting. Selecting to use SATA in AHCI mode will improve performance of the system, provided the hard drive being used supports AHCI mode. Refer to [Section 13.1.5, “Legacy and AHCI SATA Mode” on page 67](#) for instructions on selecting the desired SATA mode in the BIOS setup selection menu.

1. Prepare the system to boot from CDROM by using the boot selections in the BIOS Setup Menu. Details of the BIOS setup options are available in [Chapter 13.0, “Pre-Boot Firmware \(BIOS\).”](#)

2. Power on the system with disc #1 in the CDROM. The system should begin to boot from the CentOS installation disc #1.

Note: It is recommended that all CentOS installation discs be verified at least once prior to an installation of the OS.

3. Upon booting from disc #1, the installation process prompts if the CD media should be tested.
4. Review the CentOS Release Notes during the installation if desired. Continue with the installation process whenever prompted with the “Next” button.
5. Select the language to use during the CentOS installation process.
6. Select the appropriate keyboard for the system.
7. Select the option to perform a fresh install of CentOS on the system.
8. The hard drive that comes with the development board is initially unformatted. Partition this hard drive.
9. Select the time zone in which the system is located.
10. Select the city nearest to the system’s located.
11. Enter the desired root password for the system in the box labeled “Root Password:” and reaffirm the root password by entering the password in the box labelled “Confirm:”.

IMPORTANT: You must select the “Customize Now” button in step 12. to get access to the “Software Development” options required to complete the installation successfully.

12. Select “Customize Now” and see [Section 3.3.1](#), which provides “Development” options that enable GNU Compiler Collection (GCC) and glibc Tool Set installation. Select the desired sets of software to include. Selecting the “Software Development” software set ensures that GCC and glibc Tool Set are installed. These are necessary to allow the compilation of the EP80579 embedded software drivers and the rebuilding of the Linux kernel to support the EP80579 integrated processor.

3.3.1 Recommended Installation Customizations

1. Package customization is available through these selections:
 - Desktop Environments
 - Applications
 - Development
 - Servers
 - Base System
 - Virtualization
 - Clustering
 - Cluster Storage



— Languages

The recommended minimum package installation selections are given in the tips below. Additional detailed customization of each package can be accomplished by selecting a package (for example: Editor), and clicking the “Optional package” button to display the optional and default packages within the main package.

Tip: **Desktop Environments:** Optional. Install Gnome or KDE if you wish to use a Graphical User Interface (GUI).

Tip: **Applications:** Editor. Others are optional.

Tip: **Development:** Development Libraries and Development Tools are recommended to be installed. Others are optional.

Tip: **Servers:** Depending on the end use connectivity, none to all server types may apply. Some common selections are DNS Name Server, FTP Server, Legacy Network Server, Network Servers or Web Servers.

Tip: **Base System:** Administration Tools, Base and System Tools are recommended to be installed.

Tip: **Virtualization:** Optional.

Tip: **Clustering:** Optional.

Tip: **Cluster Storage:** Optional.

Tip: **Languages:** Additional language selections vary depending on user location.

2. Following package selection, click the “Next” button when prompted. After a dependency check, the installation process is ready to begin and prompts the user to have the required CentOS CDs available. To begin the installation, click the “Continue” button, click “Back” to change a package selection, or click “Reboot” to abort this installation.
3. If continuing with this installation, when prompted, insert the requested installation CD and click “Next”. When the installation is complete, and a prompt is displayed to reboot the system, remove the final CD from the CDRom and choose to reboot the system.
4. When rebooting, some normal system setup is required, such as setting the time and date. When asked, follow the prompt by clicking the “Forward” button.
5. Disable “Firewall”.
Note: If the Firewall is not disabled, the customer would have to execute appropriate iptable commands to allow VPN traffic.
6. SELinux should be changed from the default “Enforcing” to “disabled”.
7. Set the time and date.
Note: Although you are creating additional users in the next step, please note that the instructions in this document should be executed as the root user.
8. Create additional users.
9. Finally, when prompted, press the “Finish” button.

3.4 Installing CentOS 5.2 Kernel Source

Note: Always login as the root user to carry out the instructions in this document.



1. Follow the instructions in the man page for the “date” command to set the current date. For example:

```
date 072910302008
```

to set the date to July 29th, 2008, 10:30AM.

Note: If the Time, Date and Timezone are not set on the development board, the build command used later in this procedure will go into an infinite loop.

2. Create a directory for file download, build and install activities. For example, create a directory called “EP805XX_release” with the following commands:

```
mkdir /EP805XX_release  
cd /EP805XX_release
```

3. Create the ICP_ROOT environment variable:

```
export ICP_ROOT=$PWD
```

4. Check to see if unifdef rpm is installed using the command:

```
rpm -q unifdef
```

5. If the rpm is not installed, copy unifdef rpm from CentOS Installation Disk 3 to the \$ICP_ROOT directory and install the rpm using the following commands.

- a. Mount the CD-ROM drive as follows:

```
mkdir /mnt/cdrom  
mount -t auto /dev/cdrom /mnt/cdrom
```

- b. With OS Disk 3 inserted in the CD-ROM drive, issue the following command:

```
cp /mnt/cdrom/CentOS/unifdef-1.171-5.fc6.i386.rpm .
```

- c. Unmount the CD-ROM drive:

```
umount /dev/cdrom
```

- d. Install the rpm as follows:

```
rpm -ivh unifdef-1.171-5.fc6.i386.rpm
```

6. If the eth0 port is connected to the internet, you can ftp the rpm given below directly from the target machine. If not, proceed as follows:

Use ftp from another machine to get the kernel source rpm packages:

```
ftp://ftp.redhat.com/pub/redhat/linux/enterprise/5Server/en/os/SRPMS/kernel-2.6.18-92.el5.src.rpm
```

Using a Flash drive or CDROM, copy the kernel source rpm package to the \$ICP_ROOT directory on the target machine.

A Flash drive can be mounted as follows:

```
mount /dev/sda1 /mnt/sda1
```

Execute the following commands:

Note: If the directory is already created, skip the “mkdir” command below.



```
cd $ICP_ROOT
mkdir -p /usr/src/redhat
rpm -ivh kernel-2.6.18-92.el5.src.rpm --nosignature
```

Note: If the “warning: user brewbuilder does not exist - using root” and “warning: group brewbuilder does not exist - using root” message is displayed, it can be ignored.

The RPM .spec file (/usr/src/redhat/SPECS/kernel-2.6.spec) is created. This spec file must be installed using the “rpmbuild” command as indicated in the following step.

7. Execute the following commands:

Note: In the following step, although messages are displayed immediately after the command is entered, there may be delays between messages. Please wait until the prompt is displayed.

```
cd /usr/src/redhat/SPECS
rpmbuild -bp kernel-2.6.spec
```

A new /usr/src/redhat/BUILD/kernel-2.6.18 source directory is created.

8. Set the environment variable KERNEL_SOURCE_ROOT to the new kernel source directory:

```
export KERNEL_SOURCE_ROOT=/usr/src/redhat/BUILD/kernel-2.6.18/linux-2.6.18.i386
```

9. Also, set the following symbolic link to the kernel source directory:

```
mkdir /usr/src/kernels /* if not already created*/
ln -s $KERNEL_SOURCE_ROOT /usr/src/kernels/linux
```

3.5 Unpacking the EP80579 Security Software Linux Package

The EP80579 security software package comes in the form of a tarball. See [Section 1.2.1, “Where to Find Current Software and Documentation” on page 7](#) for the software location. The package can be unpacked at any location on the system, but for the purposes of this Getting Started Guide, a recommendation is provided.

Transfer the gzip file to the EP80579 target system using any preferred method, for example, a USB flash drive, CDROM or network transfer. Unpack the tarball in the \$ICP_ROOT directory using the following commands:

Note: Without the option ‘m’ in the following tar command, the build might get into an infinite loop if the date is set incorrectly.

```
cd $ICP_ROOT
tar -xmvzf Security.L.1.0.2-xxx.tar.gz
tar -xmvzf Security.L.1.0.2-xxx_SW.tar.gz
tar -xmvzf PatchFiles_Security.L.1.0.2-xxx.tar.gz
```

A new directory structure is created under the \$ICP_ROOT directory that contains all EP80579 security software for Linux. See [Section 1.5.3, “Package Release Structure” on page 10](#) for details of the directory structure of the software release.

You will also see a file “filelist” that contains the list of files included in the release.



3.6 Updating PCI Device Names

PCI device names must be updated prior to driver installation. This can be done by performing the following steps:

1. Download the most recent PCI ID list to temporary location from:
<http://pciids.sourceforge.net/pci.ids.bz2>
2. Backup current pci.ids file

```
mv /usr/share/hwdata/pci.ids /usr/share/hwdata/pci.ids.old
```

3. Extract the contents of the archive

```
bunzip2 pci.ids.bz2
```

4. Copy the updated file into the /usr/share/hwdata directory

```
mv pci.ids /usr/share/hwdata
```

The utility “lspci” can be used to verify the changes. The strings returned by the command will contain “Intel Corporation EP80579”.

3.7 Modifications for Use with OCF (Optional)

The modifications described in this section are required only for use with the OpenBSD Cryptographic Framework (OCF).

3.7.1 Applying the OCF Patch

The source for this patch can be obtained from the following URL:

<http://heanet.sourceforge.net/sourceforge/ocf-linux/ocf-linux-26-20071215.patch.gz>

1. Set up environmental variables (if they are not already set)

```
export ICP_ROOT=/EP805XX release
export KERNEL_SOURCE_ROOT=/usr/src/redhat/BUILD/kernel-2.6.18/linux-2.6.18.i386
```

2. Download the patch and copy to the \$ICP_ROOT directory. gunzip it as follows:

```
cd $ICP_ROOT
gunzip ocf-linux-26-20071215.patch.gz
cd $KERNEL_SOURCE_ROOT
patch -p1 < $ICP_ROOT/ocf-linux-26-20071215.patch
```

3. Update line 720 of \$KERNEL_SOURCE_ROOT/drivers/char/random.c. from:

```
- input_pool.entropy_count < random_write_wakeup_thresh);
```

to:

```
+ input_pool.entropy_count <= random_write_wakeup_thresh);
```

For additional information on this change, refer to IXA00239764 and IXA00239767 in the release notes.



4. This patch creates the crypto/ocf folders and modifies a couple of kernel source files. However, it does not modify crypto/Kconfig and crypto/Makefile. These modifications need to be applied manually using the following commands:

- a. Change the directory:

```
cd $KERNEL_SOURCE_ROOT/crypto
```

- b. Add the line 'source "crypto/ocf/Kconfig"' at line 501:

```
sed -i 501i"source\ \"crypto/ocf/Kconfig\""" Kconfig
```

- c. Append the line 'obj-\$(CONFIG_OCF_OCF) += ocf/' to the makefile:

```
echo "obj-$(CONFIG_OCF_OCF) += ocf/" >> Makefile
cd ..
```

3.7.2 Configuring OCF Parameters

1. Make backup copies of the default configuration file .config and execute the following commands to replace the default configuration file:

```
cd $KERNEL_SOURCE_ROOT
make mrproper
cp /boot/config-2.6.18-92.el5 .config
sed -i s/2.6.18-92.el5/2.6.18-EP805XX/ .config
```

2. Edit the .config file using the graphical interface menuconfig

```
make menuconfig
```

3. In the graphical interface, navigate to "Cryptographic Options".
4. Under "OCF Configuration", set "OCF (Open Cryptographic Framework)" to 'M'. 'M' sets the feature to be built as a module.
5. The above setting spawns a larger menu.
Set the following entries to 'M' to enable the random number generation daemon:

```
"crypto random --- harvest entropy for /dev/random"
```

6. The above entry spawns a sub-menu. Verify that the following entry is blank to stop additional checks for randomness. Also, do not build ocf-bench.

```
"enable fips RNG checks"
"ocf-bench (HW crypto in-kernel benchmark)"
```

7. Set the following entries to 'M' to provide support for OCF invocation from user space (cryptodev) and software cryptolibrary (cryptosoft).

```
"cryptodev (user space support)"
"cryptosoft (software crypto engine)"
```

8. Exit the graphical interface by pressing the Escape button several times. Save the new configuration.

3.7.2.1 Copying Include Files for OCF

Execute the following commands to copy the required include files:



```
mkdir -p /usr/include/crypto
cp $KERNEL_SOURCE_ROOT/crypto/ocf/cryptodev.h /usr/include/crypto/
```

3.7.3 Backporting OCF Features and Bug Fixes

OCF EP80579 driver has been tested for both OCF 20070727 and OCF 20071215. Since version OCF 20071215 in combination with Openswan 2.4.11 has a performance issue for high rates of traffic, the recommended versions are OCF 20071215 and Openswan 2.4.9. The following patches fix errors found in OCF20071215. Apply the two patches using the following commands:

```
cd $KERNEL_SOURCE_ROOT
patch -p0 < $ICP_ROOT/OpenSourcePatches/ocf-linux-20071215-driver-removal.patch
cd crypto
patch -p1 < $ICP_ROOT/OpenSourcePatches/ocf-linux-20080917_20071215_pke_dsa_ver_cryptodev.patch
```

3.8 Rebuilding the Kernel

Note: Do NOT execute the commands in [Section 3.8.1](#) if you executed commands in [Section 3.7, “Modifications for Use with OCF \(Optional\)”](#) on page 25. In this case, continue with [Section 3.8.2](#).

3.8.1 Restore the Old Configuration

Note: When the user issues the following commands, the kernel config file is stepped on, OCF has to be turned on again, and the kernel has to be renamed to 2.6.18-EP805XX.

```
cd $KERNEL_SOURCE_ROOT
make mrproper
make oldconfig
```

3.8.2 Modifying the Makefile

Change the directory using the following command:

```
cd $KERNEL_SOURCE_ROOT
```

Modify the Makefile file in the current directory as follows:

Edit the line “EXTRAVERSION = -prep” by editing “-prep” to a name meaningful to the purpose of recompiling the kernel. A simple suggestion is to change “-prep” to “-EP805XX”.

Once the kernel is recompiled, this helps to indicate which vmlinuz kernel files and entries in the /etc/grub.conf file are related to the recompiled kernel.

3.8.3 Building and Installing the Patched Kernel Source

Note: This step will take about an hour and a half to complete.

Change to the directory \$KERNEL_SOURCE_ROOT and execute the following commands to recompile the Linux kernel:

```
make && make modules && make modules_install && make install
```



Note: The make commands will be executed only if the preceding make commands were successful.

Save the configuration of the new 2.6.18-EP805XX kernel in the /boot directory using the following command:

```
cp $KERNEL_SOURCE_ROOT/.config /boot/config-2.6.18-EP805XX
```

The new kernel is compiled and placed in the /boot directory. The new kernel is named vmlinuz-2.6.18-EP805XX. A symbolic link is created from vmlinuz -> vmlinuz-2.6.18-EP805XX.

3.8.4 Making the New Kernel Default

Also, the /boot/grub/grub.conf file is modified to include the possibility of booting to this newly compiled kernel. An entry is created for the new kernel:

```
title CentOS (2.6.18-EP805XX)
  root (hd0,0)
  kernel /vmlinuz-2.6.18-EP805XX ro root=/dev/VolGroup00/LogVol100 rhgb quiet
  initrd /initrd-2.6.18-EP805XX.img
```

1. The title may be changed to be more reflective of this kernel's purpose or name.
2. To boot to this kernel image by default, modify the /boot/grub/grub.conf file's "default" entry. Entries begin from zero (0) and are counted top-down. Edit "default=<kernel entry number>", where <kernel entry number> is the number entry for the new vmlinuz-2.6.18-EP805XX kernel.
3. Skip to step (5) if your platform uses less than 1GB of memory.
4. Add the "mem=832M" (without the quotes, use leading space to demarcate) to the end of the kernel line. A sample entry might look like the following, assuming 1 GB physical memory:

```
kernel /vmlinuz-2.6.18-EP805XX ro root=/dev/VolGroup00/LogVol100 rhgb mem=832M
```

Note: 832M = 1G - 32M(NCDRAM size) - 32M(CDRAM size) -128M).
32M = 0x2000 (BIOS setting for CDRAM and NCDRAM size) * 4K (page size)

Although we do not require 32M of CDRAM and NCDRAM for this release, all validations have been done using the above values. All features have been validated using these BIOS settings.

5. Save the changes.

3.8.5 Rebooting and Verifying

Rebooting the system reboots to the newly compiled and installed kernel.

Execute the following command to ensure that the correct kernel has been loaded:

```
uname -r
```

The output should look like:

```
2.6.18-EP805XX
```



4.0 Building EP80579 Security Software on a Target Development Board

This chapter provides instructions for unpacking the EP80579 security software package, setting up the environment and building/compilation instructions.

Note: System commands given in this chapter assume that the user is issuing commands from a bash shell. This is the default shell. Use the “echo \$0” command to verify use of the bash shell or run “/bin/bash” to switch to the bash shell.

4.1 Environment Setup

After unpacking the release package (as described in [Section 3.5](#)), issue the following commands to restore the environment variables:

Note: All environment variables set for kernel rebuild are lost when the system was rebooted.

Note: As per the instruction in the previous chapters, the kernel source are untarred at KERNEL_SOURCE_ROOT and other sources are untarred at ICP_ROOT.

```
export ICP_ROOT=/EP805XX_release
export ICP_BUILDSYSTEM_PATH=$ICP_ROOT/build_system
export KERNEL_SOURCE_ROOT=/usr/src/redhat/BUILD/kernel-2.6.18/linux-2.6.18.i386
```

Follow the instructions in the man page for the “date” command to set the current date. For example,

```
date 102910302007
```

sets the date to October 29th, 2007, 10:30AM.

4.2 Build Options

[Section 4.3, “Build Using Top Level Make” on page 30](#) describes the instructions for building Security and Embedded components together. The OCF Shim component is always built separately.

Refer to [Chapter 8.0, “Building Components Individually”](#) to install various security and driver components individually.



4.3 Build Using Top Level Make

It is required to set an ICP_BUILD_OUTPUT environment variable that defines where built objects are copied and where the sample load scripts can find the built objects.

For example, to store the resulting files in a directory called \$ICP_ROOT/StagingArea, use the following commands:

```
mkdir $ICP_ROOT/StagingArea
export ICP_BUILD_OUTPUT=$ICP_ROOT/StagingArea
```

Caution: Please make sure that the current date has been set, otherwise the make process goes into an endless loop.

The following command builds the EP80579 security software and EP80579 embedded software drivers:

```
cd $ICP_ROOT
make clean
make
```

Note: The following warnings are displayed when icp_asd.ko module is built. These warnings do not limit/impact functionality in any way, as the error messages are related to symbols that are defined in other modules that are present in the system when icp_asd.ko is loaded and are resolved at load time.

```
WARNING: "icp_AsdCfgLacStop" [/EP805XX_release/Acceleration/drivers/icp_asd/src/
kernel/linux/icp_asd.ko] undefined!
WARNING: "icp_AsdCfgHalStart" [/EP805XX_release/Acceleration/drivers/icp_asd/src/
kernel/linux/icp_asd.ko] undefined!
WARNING: "halAe_Init" [/EP805XX_release/Acceleration/drivers/icp_asd/src/kernel/
linux/icp_asd.ko] undefined!
WARNING: "icp_AsdCfgHalInit" [/EP805XX_release/Acceleration/drivers/icp_asd/src/
kernel/linux/icp_asd.ko] undefined!
WARNING: "icp_AsdCfgHalStop" [/EP805XX_release/Acceleration/drivers/icp_asd/src/
kernel/linux/icp_asd.ko] undefined!
WARNING: "UcLo_MapObjAddr" [/EP805XX_release/Acceleration/drivers/icp_asd/src/
kernel/linux/icp_asd.ko] undefined!
WARNING: "icp_AsdCfgLacShutdown" [/EP805XX_release/Acceleration/drivers/icp_asd/
src/kernel/linux/icp_asd.ko] undefined!
WARNING: "icp_AsdCfgQatalStart" [/EP805XX_release/Acceleration/drivers/icp_asd/src/
kernel/linux/icp_asd.ko] undefined!
WARNING: "QatComms_intr" [/EP805XX_release/Acceleration/drivers/icp_asd/src/
kernel/linux/icp_asd.ko] undefined!
WARNING: "halAe_DelLib" [/EP805XX_release/Acceleration/drivers/icp_asd/src/kernel/
linux/icp_asd.ko] undefined!
WARNING: "UcLo_InitLib" [/EP805XX_release/Acceleration/drivers/icp_asd/src/kernel/
linux/icp_asd.ko] undefined!
WARNING: "icp_AsdCfgQatalStop" [/EP805XX_release/Acceleration/drivers/icp_asd/src/
kernel/linux/icp_asd.ko] undefined!
WARNING: "icp_AsdCfgQatalShutdown" [/EP805XX_release/Acceleration/drivers/icp_asd/
src/kernel/linux/icp_asd.ko] undefined!
WARNING: "QatComms_bh_schedule_register" [/EP805XX_release/Acceleration/drivers/
icp_asd/src/kernel/linux/icp_asd.ko] undefined!
WARNING: "UcLo_DeleObj" [/EP805XX_release/Acceleration/drivers/icp_asd/src/kernel/
linux/icp_asd.ko] undefined!
WARNING: "icp_AsdCfgQatalInit" [/EP805XX_release/Acceleration/drivers/icp_asd/src/
kernel/linux/icp_asd.ko] undefined!
WARNING: "QatComms_bh_handler" [/EP805XX_release/Acceleration/drivers/icp_asd/src/
kernel/linux/icp_asd.ko] undefined!
WARNING: "icp_AsdCfgHalShutdown" [/EP805XX_release/Acceleration/drivers/icp_asd/
src/kernel/linux/icp_asd.ko] undefined!
WARNING: "UcLo_WriteUimageAll" [/EP805XX_release/Acceleration/drivers/icp_asd/src/
kernel/linux/icp_asd.ko] undefined!
```



```
WARNING: "icp_AsdCfgLacInit" [/EP805XX_release/Acceleration/drivers/icp_asd/src/
kernel/linux/icp_asd.ko] undefined!
WARNING: "icp_AsdCfgQatalStart" [/EP805XX_release/Acceleration/drivers/icp_asd/
src/kernel/linux/icp_asd.ko] undefined!
```

The OCF Shim can be built optionally after building the Acceleration kernel modules using the following command:

```
export ICP_OCF_SRC_DIR=$KERNEL_SOURCE_ROOT/crypto/ocf
make ocf
```

Table 2. Output Files Created in the \$ICP_BUILD_OUTPUT Directory

File Name	Description
*.ko	Kernel modules
*.bin	ASU microcode files
icp_asd.conf	Installed in /etc and is the configuration file used by ASD
asd_ctl	User space utility that downloads the /etc/icp_asd.conf file to ASD during initialization
debugmgr	Proprietary debug command to get software version information and data dump
install_security.sh	Installation scripts
qat_service	Shell script used to Start and Stop kernel modules

4.4 Installation of Build Output

Execute the following command to install the Embedded and Accelerated kernel modules:

```
make install
```

Besides copying the kernel modules to a sub-directory under /lib, a startup script is added in the rcx.d directory to make the accelerated kernel modules persistent (that is, they are available for use after a system reset). The kernel modules are loaded to enable EP80579 embedded software drivers and EP80579 security software.

If you built the optional OCF Shim using the 'make ocf' command, copy the kernel module to /lib/... using the following command:

```
cp -a $ICP_ROOT/StagingArea/icp_ocf.ko /lib/modules/2.6.18-EP805XX/
depmod -a
```

4.4.1 Statistics Collection

The Cryptographic API supports statistics retrieval and display for the individual symmetric and asymmetric components. Statistic collection requires the use of atomic operations which are expensive, therefore disabling this functionality results in significant performance gains.

Configuration is performed by including parameters when the icp_crypto kernel module is loaded. As such, the module must be unloaded and then re-loaded for these changes to take effect. Before unloading the icp_crypto module, other security modules must be unloaded. The modules can be unloaded using the following instructions:

```
rmmod icp_asd
rmmod icp_crypto
```

See [Section 4.4.1.4](#) for instructions on loading the modules again.



4.4.1.1 Enable/Disable All Statistics Collection

By default, all statistics collection is enabled when `icp_crypto` module is loaded. The parameter `icp_crypto_statistics_master` can be used to enable or disable all statistics collection. These commands must be executed in the directory where the `icp_crypto` kernel object is located.

To enable statistic collection:

```
# modprobe icp_crypto icp_crypto_statistics_master=all
```

When this command is issued, the module will load with all statistics collection enabled. This is the default setting, so it is not necessary to pass in statistics master setting shown here. The real value of this parameter is observed when used in conjunction with disabling specific statistics gathering as discussed in [Section 4.4.1.2](#).

To disable all statistic collection:

```
# modprobe icp_crypto icp_crypto_statistics_master=none
```

When this command is issued, the module will load with all statistics collection disabled. All individual statistics settings discussed in [Section 4.4.1.2](#) are ignored when this parameter is set to none.

4.4.1.2 Disabling Individual Statistics Gathering

As mentioned previously, it is possible to control which statistics are collected. When `icp_crypto_statistics_master` is set to all, all statistics are collected unless they are individually disabled. The following statistics can be individually disabled:

- `icp_crypto_statistics_dsa`
- `icp_crypto_statistics_dh`
- `icp_crypto_statistics_ln`
- `icp_crypto_statistics_prime`
- `icp_crypto_statistics_rsa`
- `icp_crypto_statistics_key`
- `icp_crypto_statistics_cb`
- `icp_crypto_statistics_alg_chain`
- `icp_crypto_statistics_random`
- `icp_crypto_statistics_msgs`

To disable specific statistics, set the parameter value to off when loading the `icp_crypto` module. For example, the following command will enable all statistics gathering except the `dsa` module:

```
# modprobe icp_crypto icp_crypto_statistics_master=all  
icp_crypto_statistics_dsa=off
```

4.4.1.3 Verbose Output

The parameter `icp_crypto_verbose` can be turned on to inform the user of the statistics gathering options selected at load time. Remember if `icp_crypto_statistics_master` is set to all, all statistics collecting is enabled, unless otherwise noted under the individual statistics settings. Use the command “`dmesg`” to view the statistics gathering settings after loading `icp_crypto` kernel module.

```
# modprobe icp_crypto icp_crypto_verbose=1 icp_crypto_statistics_master=all
```




```
icp_crypto_statistics_dsa=off
# dmesg
...
Loading LAC Module
statistics_master = all --> All statistics ON
Collecting statistics:
  statistics_master = all
  statistics_dsa    = off
  statistics_dh     = on
  statistics_ln     = on
  statistics_prime  = on
  statistics_rsa    = on
  statistics_key    = on
  statistics_cb     = on
  statistics_alg_chain = on
  statistics_random = on
  statistics_msgs   = on
```

4.4.1.4 Reloading Security Modules

After updating the statistics gathering options of the `icp_crypto` kernel module, it is necessary to re-load the other security modules. This can be accomplished by performing the following steps:

```
# modprobe icp_asd
# /etc/init.d/asd_ctl
```

4.5 Uninstalling Embedded and Security Kernel Modules

The following `uninstall` option is available to unload the kernel modules and uninstall the kernel modules from the relevant `/lib` sub-directory.

```
make uninstall
```

The user can load and unload the kernel modules without deleting them from the `/lib` sub-directory using the command:

```
/etc/init.d/qat_service stop
```



5.0 Runtime Configuration

A user working in an X-Windows environment may want to execute the following command to disable X-Windows and work in a command line environment:

```
init 3
```

Note: The user can switch amongst six screens by entering the key sequence <ALT><F1| F2| F3| F4| F5| F6>. The “init 5” command takes the user back into X-Windows mode.

5.1 Loading OCF and OCF Shim

Execute the following commands to install OCF for use with the crypto accelerators on the Intel® EP80579 Integrated Processor with Intel® QuickAssist Technology:

```
modprobe ocf
modprobe cryptodev
modprobe icp_ocf
```

At this point, the user should be able to see the following processes:

- ocf-random (OCF thread that fills entropy pool in Linux)
- icpwq (thread used by OCF shim for session deregistration)

by executing the following commands:

```
ps -ef | grep "ocf"
ps -ef | grep "icp"
```

5.2 Using the Intel® QuickAssist Technology Cryptographic API

At this point, the user can configure and use a Network crypto application such as VPN.

[Chapter 6.0, “Build and Install Openswan”](#) describes how to build and install the Openswan IPsec network stack component on the development board.

[Chapter 7.0, “Configuring Sample VPN Application”](#) provides instructions for configuring a VPN application using the development board as one of the VPN gateways. The VPN application has been configured and validated with another IA computer running the CentOS 5.2 OS and Openswan.



6.0 Build and Install Openswan

This chapter provides instructions for building and installing Openswan on the development board. The next chapter describes how the EP80579 security software thus configured can be used to set up a sample VPN application.

Note: It is not required to install Openswan to use the Intel® EP80579 Software for Security Applications on Intel® QuickAssist Technology Cryptographic API.

6.1 Environment Setup

Define the following environment variables:

```
export ICP_ROOT=/EP805XX_release
export KERNEL_SOURCE_ROOT=/usr/src/redhat/BUILD/kernel-2.6.18/linux-2.6.18.i386
```

Note: As per the instruction in the previous chapters, the kernel source are untarred at KERNEL_SOURCE_ROOT and other sources are untarred at ICP_ROOT.

6.2 Building and Installing the GMP Library

Openswan uses a large number and arbitrary arithmetic precision library called GMP for certain mathematical functions.

1. Download the GMP library from the following location:
<http://ftp.sunet.se/pub/gnu/gmp/gmp-4.2.1.tar.gz>
2. Copy the file to the \$ICP_ROOT directory and extract the downloaded GMP package:

```
cd $ICP_ROOT
tar -xvzf gmp-4.2.1.tar.gz
```

3. Change directory to the newly created GMP directory:

```
cd gmp-4.2.1
```

4. Run the autoconfigure script:

```
./configure
```

5. Compile the code:

```
make
```

6. Run the required tests:

```
make check
```



7. If successful, install the package:

```
make install
```

8. Add the following configuration to recognize libraries in /usr/local/lib:

```
echo "/usr/local/lib" > /etc/ld.so.conf.d/gmp-4.2.1.conf
```

9. Update the shared libraries:

```
ldconfig
```

10. Execute the following command to verify that the gmp library is recognized:

```
ldconfig -p | grep gmp
```

6.3 Building and Installing OpenSSL

Note:

When Openswan is patched to work with the OCF, it requires the “bignum” library from OpenSSL. OpenSSL is installed simply to access the “bignum” library that comes with it. OpenSSL cryptography features can be accelerated using QuickAssist technology by using the “-engine cryptodev” option.

1. Download the OpenSSL package from the following location:

<http://www.openssl.org/source/openssl-0.9.8g.tar.gz>

Note: The downloaded package has the extension tar.tar and not tar.gz.

2. Download the OCF patch for Linux from the following location:

<http://heanet.dl.sourceforge.net/sourceforge/ocf-linux/ocf-linux-20080917.tar.gz>

3. Extract the OCF patch for openssl-0.9.8g from the above tarball using the following command:

```
tar -xvzf ocf-linux-20080917.tar.gz ocf-linux-20080917/openssl-0.9.8g.patch
```

Note: Only the patch file for OpenSSL is extracted from this tarball. Rest of the document still refers to OCF patches for the release OCF 20070727.

4. Copy the OpenSSL patch file to the \$ICP_ROOT directory and extract the downloaded OpenSSL package. Execute the following commands:

```
cd $ICP_ROOT
tar -xvf openssl-0.9.8g.tar.tar
cd openssl-0.9.8g
```

5. Apply the patch using the following commands:

```
patch -p1 < ../ocf-linux-20080917/openssl-0.9.8g.patch
patch -p1 < $ICP_ROOT/OpenSourcePatches/ocf-openssl-
0.9.8_verE_and_verG_PKE_fix.patch
```

6. Run the “autoconfigure” script with some additional parameters related to the shared library. Be aware that the filename is “Configure” with a capital C and there is a space between “ssl_0.9.8g linux-elf”

Note: The option “--with-cryptodev-digests” should **not** be used.

```
./Configure threads shared --with-cryptodev
--prefix=/usr/local/ssl_0.9.8g linux-elf -g
```



7. Compile and install the package:

```
make
make install
```

8. To ensure that the libraries are correctly installed, copy the libcrypto.so and libssl.so files to the /usr/lib directory and add the libraries into ldconfig's path:

```
echo "/usr/local/ssl_0.9.8g/lib" >> /etc/ld.so.conf.d/qt-i386.conf
ldconfig
```

9. Execute the following commands to ensure that the openssl-0.9.8g library is recognized:

```
ldconfig -p | grep libssl
ldconfig -p | grep libcrypto
```

10. Execute the following command to ensure that openssl binary is using the correct libraries:

```
ldd /usr/local/ssl_0.9.8g/bin/openssl
```

The output should look like the following:

```
libssl.so.0.9.8 => /usr/local/ssl_0.9.8e/lib/libssl.so.0.9.8 (.....)
libcrypto.so.0.9.8 => /usr/local/ssl_0.9.8e/lib/libcrypto.so.0.9.8
(.....)
```

11. Execute the following commands so that the newly installed openssl is properly recognized by other components:

```
cd /usr/local
ln -f -s ssl_0.9.8g openssl
ln -f -s ssl_0.9.8g ssl
export PATH=/usr/local/ssl_0.9.8g/bin:${PATH}
export LD_LIBRARY_PATH=/usr/local/ssl_0.9.8g/lib:${LD_LIBRARY_PATH}
```

12. To ensure the OpenSSL headers are correctly installed, copy the contents of the ./include/openssl/ directory into the /usr/include/openssl directory:

```
cd $ICP_ROOT/openssl-0.9.8g
mkdir -p /usr/include/openssl
cp -L ./include/openssl/* /usr/include/openssl
```

Note: Overwrite existing files if required.

13. Verify that the correct version of OpenSSL has been installed using the following command:

```
openssl version
```

The output should be similar to the following:

```
OpenSSL 0.9.8g 23 Oct 2007
```

14. Install the OCF stack as described in [Chapter 5.0, "Runtime Configuration."](#) Verify the OCF access from the OpenSSL engine using the following commands:

```
openssl engine -t
```

The output should look like the following:

```
(cryptodev) BSD cryptodev engine [available]
```



15. Check the crypto operations supported by the OCF engine using the following command:

```
openssl engine -c
```

The output should look like the following:

```
(cryptodev) BSD cryptodev engine [RSA, DSA, DH, DES-CBC, DES-EDE3-CBC, AES-128-CBC, AES-256-CBC]
```

16. Verify that an OpenSSL speed test can be run by executing the following command:

```
openssl speed -evp des3 -elapsed -engine cryptodev
```

Note: Loading the module cryptosoft.ko only will result in OCF using crypto software library. However, loading the module icp_ocf.ko will result in OCF using the hardware accelerator.

6.4 Building and Installing Openswan

6.4.1 Prepare to Build Openswan

1. The Openswan key management daemon is called Pluto. To use OCF for crypto functionality, Pluto requires certain flags (HAVE_OCF and HAVE_OPENSSL) to be enabled. Execute the following commands:

```
export KERNELSRC=$KERNEL_SOURCE_ROOT
cd $KERNELSRC
cp ./include/linux/autoconf.h ./include/linux/autoconf.h.sav
echo "#define CONFIG_KLIPS_OCF 1" >> ./include/linux/autoconf.h
make modules_prepare
```

2. Copy the updated file into the system include directory as follows:

```
cp ./include/linux/autoconf.h /usr/include/linux/autoconf.h
```

Note: You may need to create some of the referenced directories first.

6.4.2 Building and Installing Openswan

1. Download the Openswan package and OCF-related patch from the following locations:
<http://www.openswan.org/download/old/openswan-2.4/openswan-2.4.9.tar.gz>
<http://heanet.dl.sourceforge.net/sourceforge/ocf-linux/ocf-openswan-2.4.9-20070727.patch.gz>
2. Copy the files to the \$ICP_ROOT directory and extract the downloaded Openswan package. Execute the following commands:

```
cd $ICP_ROOT
tar -xvzf openswan-2.4.9.tar.gz
gunzip ocf-openswan-2.4.9-20070727.patch.gz
```

3. Change directory to the newly created openswan-2.4.9 directory and apply the patch to enable OCF usage:

```
cd openswan-2.4.9
```



```
patch -p0 < ../ocf-openswan-2.4.9-20070727.patch
```

4. Again, to backport features and bug fixes, apply the following patch:

```
patch -p1 < $ICP_ROOT/OpenSourcePatches/ocf-openswan-2.4.9-20070727-session-
migration-backport.patch
```

5. Apply patch to allow openswan-2.4.9 to be built for CentOS 5.2:

```
patch -p1 < $ICP_ROOT/OpenSourcePatches/ocf-openswan-2.4.9-20070727-
centos5.2.patch
```

6. Disable the batch mode in OCF by editing the file `./linux/net/ipsec/ipsec_ocf.c`. Set the constant `USE_BATCH` to zero. For example:

```
#define USE_BATCH 0
```

7. Enforce packet ordering in Openswan.

Note: This is an optional step. Applying this patch will result in an average 10% to 15% lower throughput for an IPSec connection; however, packets are guaranteed to be sent in the order they are received.

```
patch -p1 < $ICP_ROOT/OpenSourcePatches/ocf-openswan-2.4.9-20070727-packet-
ordering-issue.patch
```

8. Build Openswan.

Note: The `$KERNELSRC` variable must be set as described in [Section 6.4.1](#) before the build will work.

Note: The `make mininstall` command installs KLIPS permanently and avoids a change between KLIPS and the Netkey stack.

```
make clean
make CONFIG_KLIPS_OCF=y DECLARE_TASKLET=y module >build_klips.log 2>&1
make mininstall
make HAVE_OCF_AND_OPENSSL=y programs >build_pluto.log 2>&1
make HAVE_OCF_AND_OPENSSL=y install
```

This first compiles and installs the kernel module, then compiles and installs the userland component.

9. Update any new libraries and update the kernel module dependencies:

```
ldconfig
depmod -a
```

10. Openswan installs the `S47ipsec` startup file in the `/etc/rc3.d` directory. To prevent it auto-starting on every reboot, execute the following command:

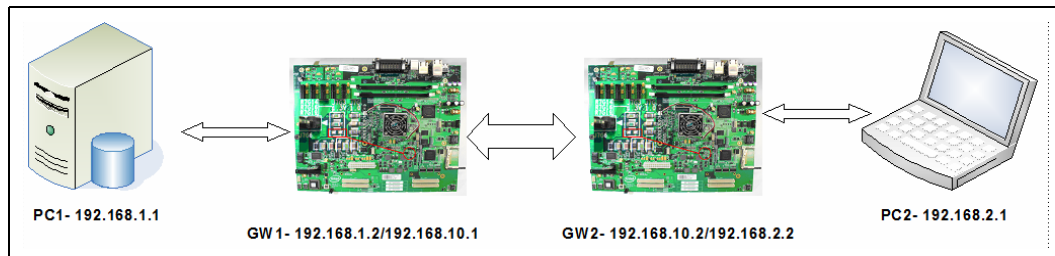
```
chkconfig ipsec off
```

7.0 Configuring Sample VPN Application

This chapter describes how to accelerate an Openswan* IPSec tunnel between an EP80579 gateway GW1 and another IA gateway GW2 using OCF and the OCF Shim for the EP80579 with QuickAssist. The resulting offload of the crypto function can be used to increase the Openswan throughput or release cycles for other user applications.

The configuration is shown in [Figure 5](#) and it assumes that the two PCs, PC1 and PC2, are running Linux*.

Figure 5. VPN Example



7.1 Configure IPSec on Both Gateways

Perform the following steps to set up an IPSec on gateways GW1 and GW2:

1. If an existing system with IPSec implementation is not available, then install one as described in [Section 7.4, "Installing the OS and Openswan on Gateway GW2" on page 45](#).
2. Install two Gigabit NICs on gateway GW2.
3. Connect the gateways GW1, GW2 and the PCs PC1 and PC2 as shown in [Figure 5](#) using crossover CAT5 cables. Alternatively, the user could use different hubs or switches to make the connections.
4. Build and install Openswan on gateway GW2.
5. Configure the IP addresses for the EP80579 internal Gigabit ports and enable IP forwarding on GW1 using the following commands:

```
ifconfig eth0 192.168.10.1/24 up
ifconfig eth1 192.168.1.2/24 up
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Note: In Linux, the Ethernet ports TT, SS and UU in [Figure 3 on page 17](#) appears as eth0, eth1 and eth2 respectively.

6. Configure the IP addresses for the IA Gigabit ports and enable IP forwarding on gateway GW2 using the following example commands:

```
ifconfig eth0 192.168.10.2/24 up
ifconfig eth1 192.168.2.2/24 up
echo 1 > /proc/sys/net/ipv4/ip_forward
```




7. Configure networking on PC1 using the following example commands:

```
ifconfig eth0 192.168.1.1/24 up
route add default gw 192.168.1.2
```

8. Configure networking on PC2 using the following example commands:

```
ifconfig eth0 192.168.2.1/24 up
route add default gw 192.168.2.2
```

9. If the network is configured properly then the user should be able to ping each of the Ethernet ports configured so far from PC1, PC2, GW1 and GW2.
10. Start OCF on GW1 as described in [Section 5.1, "Loading OCF and OCF Shim" on page 34](#).
11. Edit the /etc/ipsec.conf file on both Gateways based on details provided in [Section 7.3, "Configuring Crypto Parameters in ipsec.conf" on page 42](#).
12. Stop any IPSec service on GW1 and verify that it has stopped:

```
service ipsec stop
service ipsec status
```

13. Restart IPSec and verify success as follows:

```
service ipsec start
service ipsec status
```

14. Stop, then restart IPSec on GW2.

7.1.1 Testing if IPSec is Configured Correctly

1. The route command on each Gateway should show an additional entry for device ipsec0. Execute the following command:

```
route
```

2. Execute the following command to display the supported ciphers:

```
cat /proc/net/pf_key_supported
```

In the right-most column, a list of ciphers prefixed with "ocf-" (for example, "ocf-sha1hmac") is displayed. This indicates that Openswan has successfully detected OCF and can use it for cryptographic calls. If every entry in the right-most column reads "unknown", the installation failed.

3. Check the IPSec setup on each gateway using the following command:

```
ipsec verify
```

This command indicates the IPSec version, if "Pluto" is running and which IPSec network stack is being used -- NETKEY or KLIPS.

Note: If the output indicates that Pluto is not running, then execute the following commands as a workaround:

```
ln -s /usr/local/lib/libgmp.so.3 /lib/libgmp.so.3
```

Note: Another troubleshooting technique to narrow down an IPSec connectivity issue is to load OCF and the associated crypto software library instead of the EP80579 OCF shim. This helps to isolate the issue to Openswan or OCF Shim and LAC software. Execute the following commands to install OCF to use crypto software library:



```
modprobe ocf
modprobe cryptodev
modprobe cryptosoft
```

7.2 Set Up a Tunnel Between Gateways

Note: Execute the following commands on every reset in order to support greater than 1000 VPN tunnels and prevent ARP table from overflowing:

```
echo 2048 > /proc/sys/net/ipv4/neigh/default/gc_thresh1
echo 4096 > /proc/sys/net/ipv4/neigh/default/gc_thresh2
echo 8192 > /proc/sys/net/ipv4/neigh/default/gc_thresh3
```

1. The following command can be used to set up, bring down and tear down a tunnel:

```
ipsec auto --up <name_of_tunnel>
ipsec auto --down <name_of_tunnel>
ipsec auto --delete <name_of_tunnel>
```

where <name_of_tunnel> is defined in the /etc/ipsec.conf file. See [Section 7.3.2.1](#) for more information.

Note: This command to enable (up) need be executed only on one Gateway. However, the command to disable (down) and tear down the tunnel should be executed on both Gateways.

2. If the tunnel has been set up, the route command should show two entries with the interface set to device ipsec0.
3. Execute the following commands to trace the packet flow across the tunnel:

```
tcpdump -i eth<x>
tcpdump -i ipsec0
```

If PC1 pings PC2 across the tunnel, a packet trace of eth<x> should show ESP packets being sent across and ipsec0 should show ICMP messages being sent.

4. The number of packets sent across the tunnel can be counted using the following commands. The second command updates the count every <x> seconds.

```
ipsec eroute
watch -n <x> secs ipsec eroute
```

Using a web search engine, search for “openswan ipsec.conf” for details on each field in the file.

7.3 Configuring Crypto Parameters in ipsec.conf

The Openswan VPN application uses two configuration files:

- ipsec.secrets contains the secret keys used for authentication
- ipsec.conf contains all other configuration information for the tunnel setup

IPSec tunnels can be configured to run in either tunnel mode or transport mode. Each mode has its own particular uses and care should be taken to ensure the proper mode is selected.

In Transport mode, only the the payload of the IP packet is encrypted and/or authenticated. The routing is intact, since the IP header is neither modified nor encrypted. Transport mode is used for host-to-host communications.



In Tunnel mode, the entire IP packet (data and IP Header) is encrypted and/or authenticated. It is then encapsulated into a new IP packet with a new IP header. Tunnel mode is used to create Virtual Private Networks for network-to-network communications (such as link between routers), host-to-network communications (such as remote user access), and host-to-host communications (such as private chat).

The “authby” parameter in the ipsec.conf file provides two ways for the IKE on each host to authenticate each other:

- “rsasig”, which uses the RSA algorithm
- “secret”, which uses a Pre-Shared Key (PSK)

RSA is a little more difficult to set up, but it is the preferred method because it is more secure. PSK is slightly easier since it uses a shared secret as a key with the associated security risks and manual key exchange issues.

The next two sections describe how to create and incorporate secret keys into the configuration files and the last section provides a usable configuration file that uses PSK authentication.

7.3.1 RSA SIG

To use RSA, execute the following on each of gateway GW1 and GW2:

1. Set “authby=rsasig” in the ipsec.conf file.
2. Run the following command to generate the RSA key:

```
ipsec rsasigkey --verbose 2192 > rsakey.tmp
```

The following is a sample output:

```
# 1024 bits, Fri Feb 4 05:05:19 2000
# for signatures only, UNSAFE FOR ENCRYPTION
#pubkey=0x010395daee1be05f3038ae529ef2668afd79f5ff1b16203c9ceaeaf801c..
Modulus:0x95daee1be05f3038ae529ef2668afd79f5feaf801cecfb51a6ecc08890..
PublicExponent: 0x03
# everything after this point is secret
PrivateExponent:0x63e74967eaea2025c98c69f6ef0753a6a3ff676415771324dd3..
Prime1:0xc5b471a88b025dd09d4bd7b61840f9c5d45546bea3ccc7b744254f6f0b84..
Prime2:0xc20a99feeafe79763f2f45b0e96cb4aef8918ca333a326d3f6dc2c72b753..
Exponent1:0x83cdallb0756e93e1674fff4512e8d8e2f29c2888524d818df9f5d02f..
Exponent2:0x815c66a9f1fefba44b6c2b1246f5061086ccd176f37f9e81dalcf8ceb..
Coefficient:0x10d954c9e2b8d11f4db1b233ef37ff0a3c5d30186520f1753a7e325..
```

3. Copy the contents of rsakey.tmp to /etc/ipsec.secrets:

```
cp -f rsakey.tmp /etc/ipsec.secrets
```

4. Without deleting any contents, edit the /etc/ipsec.secrets file to insert a new line at the beginning and at the end.

The edited file should look like the following:

```
: RSA {
# 1024 bits, Fri Feb 4 05:05:19 2000
# for signatures only, UNSAFE FOR ENCRYPTION
#pubkey=0x010395daee1be05f3038ae529ef2668afd79f5ff1b16203c9ceaeaf801c..
Modulus:0x95daee1be05f3038ae529ef2668afd79f5feaf801cecfb51a6ecc08890..
PublicExponent: 0x03
# everything after this point is secret
PrivateExponent:0x63e74967eaea2025c98c69f6ef0753a6a3ff676415771324dd3..
Prime1:0xc5b471a88b025dd09d4bd7b61840f9c5d45546bea3ccc7b744254f6f0b84..
```



```
Prime2:0xc20a99feeafe79763f2f45b0e96cb4aef8918ca333a326d3f6dc2c72b753..  
Exponent1:0x83cda11b0756e93e1674fff4512e8d8e2f29c2888524d818df9f5d02f..  
Exponent2:0x815c66a9f1fefba44b6c2b1246f5061086ccd176f37f9e81da1cf8ceb..  
Coefficient:0x10d954c9e2b8d11f4db1b233ef37ff0a3c5d30186520f1753a7e325..  
}
```

Note: The following items should be noted:

- Line 1: ':' is on column 1; space between ':' and 'RSA' ; space between 'RSA' and '{'
 - Last Line: '}' should NOT be in column 1.
 - The comment lines after '#' are important. For example, the public key is stored as a comment.
5. The public key information from /etc/ipsec.secrets on gateway GW1 should be copied to the field 'leftsasigkey' in /etc/ipsec.conf on both GW1 and GW2. Similarly, the public key information from /etc/ipsec.secrets on gateway GW2 should be copied to the field 'rightsasigkey' in /etc/ipsec.conf on both GW1 and GW2.

Since the public key is a large string and manual copy is error prone, use the following commands to extract the exact string.

- a. Run the following command on GW1 to extract the public key:

```
ipsec showhostkey --left
```

- b. Run the following command on GW2 to extract the public key:

```
ipsec showhostkey --right
```

After the edits, the file /etc/ipsec.conf on both gateways GW1 and GW2 should be identical and the relevant fields should look like the following:

```
.....  
conn gw-to-gw  
left=192.168.10.1  
leftsubnet=192.168.1.0/24  
leftsasigkey=0x010395dae1be05f3038ae529ef2668afd79f5ff1b16203c9....  
right=192.168.10.2  
rightsubnet=192.168.2.0/24  
rightsasigkey=0x01037631b81f00d5e6f888c542d44dbb784cd3646f084ee....  
esp=3des-sha1-96  
authby=rsasig  
type=tunnel  
.....
```

Note: When entering the left and right rsasigkeys into the ipsec.conf file, be careful not to include any spaces after the key, as this will result in error when trying to start the ipsec service.

7.3.2 PSK

Note: Sample PSK secret files and corresponding config files are available in the software package. See [Section 7.3.2.1](#) for details.

To use PSK, execute the following steps:

1. Set the following in /etc/ipsec.conf on both gateways:

```
authby=secret
```



2. Generate a secret on any one of the gateways by executing:

```
ipsec ranbits --continuous 128 > somerandommonkey.tmp
```

3. Add the new secret to “/etc/ipsec.secrets” on gateway GW1. The resulting file should look like:

```
192.168.10.1 192.168.10.2 : PSK 0xb2463f384577f95448052aad46496b7ec
```

4. Add the new secret to “/etc/ipsec.secrets” on gateway GW2. The resulting file should look like:

```
192.168.10.2 192.168.10.1 : PSK 0xb2463f384577f95448052aad46496b7ec
```

Note: The PSK secret key should be the same on both gateways, but the IP address pairs should be reversed.

7.3.2.1 Sample Usable Configuration and Secret Files

Usable ipsec.conf and ipsec.secrets file can be found in \$ICP_ROOT/OpenswanConfigFiles.

Execute the following command to copy the files to GW1:

```
cp $ICP_ROOT/OpenswanConfigFiles/ipsec.conf /etc/
cp $ICP_ROOT/OpenswanConfigFiles/ipsec.left.secrets /etc/ipsec.secrets
```

Execute the following command to copy the files to GW2:

```
cp $ICP_ROOT/OpenswanConfigFiles/ipsec.conf /etc/
cp $ICP_ROOT/OpenswanConfigFiles/ipsec.right.secrets /etc/ipsec.secrets
```

Note: The sample ipsec.conf included with the software package configures the tunnel in Tunnel Mode. For more information on configuring IPSec tunnels, enter the command `man ipsec.conf`

7.4 Installing the OS and Openswan on Gateway GW2

If an existing system with an IPSec implementation is not available, then perform the following steps to install a CentOS 5.2 and Openswan on top of it:

1. Select a system that satisfies the basic hardware requirements to install a CentOS 5.2 system.
2. Follow the instructions in [Chapter 3.0, “Installing the OS on a Development Board”](#), but do not install any patches.
3. Install the gmp library by following the instructions in [Section 4.1, “Environment Setup”](#) on page 29 and [Section 6.4, “Building and Installing Openswan”](#) on page 38.
4. Download the Openswan package from the following location:
<http://www.openswan.org/download/old/openswan-2.4.9.tar.gz>

to \$ICP_ROOT and extract it using the following commands:

```
cd $ICP_ROOT
tar -xvzf openswan-2.4.9.tar.gz
```



5. Build Openswan using the following commands:

```
cd openswan-2.4.9
make module mininstall programs install
cd $ICP_ROOT
ldconfig
depmod -a
```



8.0 Building Components Individually

This chapter describes how to build EP80579 security software and EP80579 embedded software drivers individually.

8.1 Building Components Individually Using Top-Level Make

The EP80579 embedded software drivers can be built separately using the following commands:

```
# cd $ICP_ROOT
# make Embed_clean
# make Embed
# make Embed_install
```

The Acceleration kernel modules can be built separately using the following commands:

```
# cd $ICP_ROOT
# make Accel_clean
# make Accel
# make Accel_install
```

The corresponding uninstall commands are:

```
# make Embed_uninstall
# make Accel_uninstall
```



9.0 Building, Installing and Loading Individual EP80579 Embedded Software Drivers

Before building and loading EP80579 embedded software drivers, define the following environment variable:

```
export ICP_ROOT=/EP805XX
```

Note: As per the instruction in the previous chapters, the kernel source are untarred at KERNEL_SOURCE_ROOT and other sources are untarred at ICP_ROOT.

9.1 Controller Area Network (CAN) Driver

9.1.1 Linux Compilation Instructions

All source files for the Linux release of the Controller Area Network (CAN) driver are located in the following directory within the Linux compatible EP80579 security software release:

```
$ICP_ROOT/Embedded/src/CAN
```

Compilation of the Linux CAN driver separately from the rest of the software package is possible. Change to the *\$ICP_ROOT/Embedded/src/CAN* directory and execute the following commands:

```
make clean
make
```

The CAN driver compiles and the resulting can.ko file is placed in the *\$ICP_ROOT/Embedded/src/CAN/build/linux_2.6/kernel_space* directory.

9.1.2 Linux Module Load/Unload Instructions

To load the Linux CAN driver, execute the following command from the *./build/linux_2.6/kernel_space/* directory:

```
insmod can.ko
```

Note: The driver can also be installed for persistence using the “make install” command from the *\$ICP_ROOT/Embedded/src/CAN* directory.

To unload the Linux CAN driver, execute the following command:

```
rmmmod can.ko
```

The lsmod command may be used to confirm if a module has been loaded or unloaded:

```
lsmod | grep can
```




The output of the above command lists modules loaded in the system containing “can” as part of their name.

9.1.3 Linux Sample Codelet

A codelet is provided to demonstrate how a sample application interfaces with the Controller Area Network driver. This codelet is intended to run on an Intel® EP80579 Integrated Processor with Intel® QuickAssist Technology Development Board with CentOS installed. The EP80579 Embedded Software kernel should also be installed, but the codelet works even if it is not. Run the codelet as follows:

1. Login as root. Root permissions are required for all operations.
2. Open a terminal window, change directory to `$ICP_ROOT/Embedded/codelet/CAN` and execute the script. Do not move the script to another location.

```
[root@localhost ~]# ./install.bash
```

The script checks for the Controller Area Network driver and client drivers, runs all of the makefiles in the correct order, loads the client driver, and runs the application. The application output can be viewed in the terminal window.

3. Client driver output can be viewed in a second terminal window by printing `/var/log/messages` as follows:

```
[root@localhost ~]# tail -f /var/log/messages
```

4. For information about the CAN codelet, refer to the Readme document located at `$ICP_ROOT/Embedded/codelet/CAN`.

9.2 Enhanced Direct Memory Access (EDMA) Driver

9.2.1 Linux Compilation Instructions

All source files for the Linux release of the Enhanced Direct Memory Access (EDMA) driver are located in the following directory within the Linux compatible EP80579 security software release:

```
$ICP_ROOT/Embedded/src/EDMA
```

Compilation of the Linux EDMA driver separately from the rest of the software package is possible. Change to the `$ICP_ROOT/Embedded/src/EDMA` directory and execute the following commands:

```
make clean  
make
```

The EDMA driver compiles and the resulting `edma.ko` file is placed in the `$ICP_ROOT/Embedded/src/EDMA/build/linux_2.6/kernel_space` directory.

9.2.2 Linux Module Load/Unload Instructions

To load the Linux EDMA driver, execute the following command from the directory where the compiled executable resides:

```
insmod dma.ko
```

Note: The driver can also be installed for persistence using the “make install” command from the `$ICP_ROOT/Embedded/src/EDMA` directory.

To unload the Linux EDMA driver, execute the following command:



```
rmmod dma.ko
```

The `lsmod` command may be used to confirm if a module has been loaded or unloaded:

```
lsmod | grep dma
```

The output of the above command lists modules loaded in the system containing “dma” as part of their name.

9.2.3 Runtime Configuration of EDMA

Runtime configuration of the Enhanced Direct Memory Access driver is performed from a client driver communicating through the published API set as described in the Intel® EP80579 Software Drivers for Embedded Applications Programmer’s Guide and API Reference Manual. Intel leaves the design and development of a client driver to the customer since it is implementation-specific.

9.2.4 Linux Sample Codelet

A codelet is provided to demonstrate how a sample application interfaces with the EDMA driver. This codelet is intended to run on an Intel® EP80579 Integrated Processor with Intel® QuickAssist Technology Development Board with CentOS installed. The EP80579 Embedded Software kernel should also be installed, but the codelet works even if it is not. Run the codelet as follows:

1. Login as root. Root permissions are required for all operations.
2. Open a terminal window, change directory to `$ICP_ROOT/Embedded/ codelet/ EDMA` and execute the script. Do not move the script to another location.

```
[root@localhost ~]# ./install.bash
```

The script checks for the EDMA and client drivers, runs all of the makefiles in the correct order, loads the client driver, and runs the application. The application output can be viewed in the terminal window.

3. Client driver output can be viewed in a second terminal window by printing `/var/log/messages` as follows:

```
[root@localhost ~]# tail -f /var/log/messages
```

4. For information about the EDMA codelet, refer to the Readme document located at `$ICP_ROOT/Embedded/codelet/EDMA`.

9.3 Watchdog Timer (WDT) Driver

9.3.1 Linux Compilation Instructions

All source files for the Linux release of the Watchdog Timer (WDT) driver are located in the following directory within the Linux compatible EP80579 security software release:

```
$ICP_ROOT/Embedded/src/WDT
```

Compilation of the Linux WDT driver separately from the rest of the software package is possible. Change to the `$ICP_ROOT/Embedded/src/WDT` directory and execute the following commands:

```
make clean
make
```



The Watchdog Timer driver compiles and the resulting wdt.ko file is placed in the `$ICP_ROOT/Embedded/src/WDT/build/linux_2.6/kernel_space` directory.

9.3.2 Linux Module Load/Unload Instructions

To load the Linux Watchdog Timer driver, execute the following command from the directory where the compiled executable resides (`$ICP_ROOT/Embedded/src/WDT/build/linux_2.6/kernel_space`):

```
insmod wdt.ko
```

To unload the Linux Watchdog Timer driver, execute the following command:

```
rmmod wdt.ko
```

Note: The driver can also be installed for persistence using the “make install” command from the `$ICP_ROOT/Embedded/src/WDT` directory.

The `lsmod` command may be used to confirm if a module has been loaded or unloaded:

```
lsmod | grep wdt
```

The output of the above command lists modules loaded in the system containing “wdt” as part of their name.

9.3.3 Runtime Configuration of WDT

Runtime configuration of the Watchdog Timer driver is performed from a client driver communicating through the published API set as described in the Intel® EP80579 Software Drivers for Embedded Applications Programmer’s Guide and API Reference Manual. Intel leaves the design and development of a client driver to the customer since it is implementation-specific.

9.3.4 Linux Sample Codelet

A codelet is provided to demonstrate how a sample application interfaces with the Watchdog Timer driver. This codelet is intended to run on an Intel® EP80579 Integrated Processor with Intel® QuickAssist Technology Development Board with CentOS installed. The EP80579 Embedded Software kernel should also be installed, but the codelet works even if it is not. Run the codelet as follows:

1. Login as root. Root permissions are required for all operations.
2. Open a terminal window, change directory to `$ICP_ROOT/Embedded/codelet/WDT` and execute the script. Do not move the script to another location.

```
[root@localhost ~]# ./install.bash
```

The script checks that the Watchdog Timer driver is loaded, runs all of the makefiles in the correct order, and runs the application. The application output can be viewed in the terminal window.

3. For information about the WDT codelet, refer to the Readme document located at `$ICP_ROOT/Embedded/codelet/WDT`.



9.4 General Purpose I/O (GPIO) Driver

9.4.1 Linux Compilation Instructions

All source files for the Linux release of the General Purpose I/O (GPIO) driver are located in the following directory within the Linux compatible EP80579 security software release:

```
$ICP_ROOT/Embedded/src/GPIO
```

Compilation of the Linux GPIO driver separately from the rest of the software package is possible. Change to the `$ICP_ROOT/Embedded/src/GPIO` directory and execute the following commands:

```
make clean
make
```

The GPIO driver compiles and the resulting `gpio.ko` file is placed in the `$ICP_ROOT/Embedded/src/GPIO/build/linux_2.6/kernel_space` directory.

9.4.2 Linux Module Load/Unload Instructions

To load the Linux General Purpose I/O driver, execute the following command from the directory where the compiled executable resides:

```
insmod gpio.ko
```

Note: The driver can also be installed for persistence using the “make install” command from the `$ICP_ROOT/Embedded/src/GPIO` directory.

To unload the Linux General Purpose I/O driver, execute the following command:

```
rmmmod gpio.ko
```

The `lsmod` command may be used to confirm if a module has been loaded or unloaded:

```
lsmod | grep gpio
```

The output of the above command lists modules loaded in the system containing “gpio” as part of their name.

9.4.2.1 Runtime Configuration of GPIO

Runtime configuration of the General Purpose I/O driver is performed from a client driver communicating through the published API set as described in the Intel® EP80579 Software Drivers for Embedded Applications Programmer's Guide and API Reference Manual. Intel leaves the design and development of a client driver to the customer since it is implementation-specific.

9.5 IEEE 1588 Hardware Assist Driver

9.5.1 Linux Compilation Instructions

All source files for the Linux release of the IEEE 1588 Hardware Assist (1588) driver are located in the following directory within the Linux compatible EP80579 security software release:

```
$ICP_ROOT/Embedded/src/1588
```



Compilation of the Linux IEEE 1588 driver separately from the rest of the software package is possible. Change to the `$ICP_ROOT/Embedded/src/1588` directory and execute the following commands:

```
make clean
make
```

The IEEE 1588 Hardware Assist driver compiles and the resulting `timesync.ko` file is placed in the `$ICP_ROOT/Embedded/src/1588/build/linux_2.6/kernel_space` directory.

9.5.2 Linux Module Load/Unload Instructions

To load the Linux IEEE 1588 Hardware Assist driver, execute the following command from the directory where the compiled executable resides:

```
insmod timesync.ko
```

To unload the Linux IEEE 1588 Hardware Assist driver, execute the following command:

```
rmmod timesync.ko
```

Note: The driver can also be installed for persistence using the “make install” command from the `$ICP_ROOT/Embedded/src/1588` directory.

The `lsmod` command may be used to confirm if a module has been loaded or unloaded:

```
lsmod | grep timesync
```

The output of the above command lists modules loaded in the system containing “timesync” as part of their name.

9.5.3 Runtime Configuration of IEEE 1588

Runtime configuration of the IEEE 1588 Hardware Assist driver is performed from a client driver communicating through the published API set as described in the Intel® EP80579 Software Drivers for Embedded Applications Programmer’s Guide and API Reference Manual. Intel leaves the design and development of a client driver to the customer since it is implementation-specific.

9.5.4 Linux Sample Codelet

A codelet is provided to demonstrate how a sample application interfaces with the IEEE 1588 Hardware Assist driver. This codelet is intended to run on an Intel® EP80579 Integrated Processor with Intel® QuickAssist Technology Development Board with CentOS installed. The EP80579 Embedded Software kernel should also be installed, but the codelet works even if it is not. Run the codelet as follows:

1. Login as root. Root permissions are required for all operations.
2. Open a terminal window, change directory to `$ICP_ROOT/Embedded/codelet/1588` and execute the script. Do not move the script to another location.

```
[root@localhost ~]# ./install.bash
```

The script checks for the IEEE 1588 Hardware Assist driver and client drivers, runs all of the makefiles in the correct order, loads the client driver, and runs the application. The application output can be viewed in the terminal window.

3. Client driver output can be viewed in a second terminal window by printing `/var/log/messages` as follows:

```
[root@localhost ~]# tail -f /var/log/messages
```



4. For information about the 1588 codelet, refer to the Readme document located at `$ICP_ROOT/Embedded/codelet/1588`.

9.6 Global Configuration Unit and Gigabit Ethernet Drivers

Two drivers are required for enabling network connectivity on the Gigabit Ethernet controllers in the EP80579: the Global Configuration Unit (GCU) driver and the Gigabit Ethernet (GbE) driver. The GCU driver controls the MAC and administrative activities. The GbE driver controls the network connectivity. The GbE driver is dependent on the GCU driver.

Note: The Global Configuration Unit driver must be installed prior to installation of the Gigabit Ethernet driver.

9.6.1 Linux Compilation Instructions

All source files for the Linux release of the Global Configuration Unit (GCU) driver and the Gigabit Ethernet (GbE) driver are located in the following directory within the Linux compatible EP80579 security software release:

```
$ICP_ROOT/Embedded/src/GbE
```

Compilation of both the GCU and GbE drivers separately from the rest of the software package is possible. Enter the `$ICP_ROOT/Embedded/src/GbE` directory and execute the following commands:

```
make clean
make
```

The GCU and GbE drivers compile and the resulting `gcu.ko` and `iegbe.ko` files are placed in the `$ICP_ROOT/Embedded/src/GbE/build/linux_2.6/kernel_space` directory.

9.6.2 Linux Module Load/Unload Instructions

To load the Linux Global Configuration Unit driver, execute the following command from the directory where the compiled executable resides (`$ICP_ROOT/Embedded/src/GbE/build/linux_2.6/kernel_space`):

```
insmod gcu.ko
insmod iegbe.ko
```

To unload the Linux Global Configuration Unit driver, execute the following commands:

```
rmmod gcu.ko
rmmod iegbe.ko
```

The `lsmod` command may be used to confirm if a module has been loaded or unloaded:

```
lsmod | grep gcu
lsmod | grep iegbe
```

The output of the above commands list modules loaded in the system containing “gcu” and “iegbe” as part of their name.

9.6.3 Runtime Configuration of GCU and Gigabit Ethernet

Runtime configuration of network support is through traditional `ifconfig` commands, detailed in man pages installed on the system.



Note: In Linux, the Ethernet ports TT, SS and UU in [Figure 3 on page 17](#) appears as eth0, eth1 and eth2 respectively.

9.7 System Management Bus (SMBus) Driver

9.7.1 Linux Compilation Instructions

All source files for the SMBus Linux support are available in the Linux kernel source. Additionally, the `lm_sensors` rpm must be installed to work with additional SMBus hardware on the system board. The `lm_sensors` rpm can be found on Disc 2 of the CentOS 5.2 installation CDs.

Note: Before installing `lm_sensors` rpm, you can check if it has already been installed using the `rpm -q lm_sensors` command.

Install the rpm with the following command:

```
rpm -ivh lm_sensors-2.10.0-3.1.i386
```

Two additional SMBus drivers, which are available within the Linux kernel source, must be installed prior to use of the `i2c-i801` driver.

```
i2c-core  
i2c-dev
```

These drivers are available within the Linux kernel source:

```
$KERNEL_SOURCE_ROOT/drivers/i2c/i2c-core.ko  
$KERNEL_SOURCE_ROOT/drivers/i2c/i2c-dev.ko
```

It is likely that these drivers will be installed during installation of the operating system and kernel. To determine if they are installed, perform an `lsmod` similar to the examples above.

9.7.2 Linux Module Load/Unload Instructions

If the additional SMBus driver files are not installed, install them with either the `modprobe` or `insmod` commands:

```
modprobe i2c-core  
modprobe i2c-dev
```

or

```
insmod /lib/modules/`uname -r`/kernel/drivers/i2c/i2c-core.ko  
insmod /lib/modules/`uname -r`/kernel/drivers/i2c/i2c-dev.ko
```

Installing the `i2c-i801` driver is accomplished using the following command:

```
modprobe i2c-i801
```

or

```
insmod /lib/modules/`uname -r`/kernel/drivers/i2c/busses/i2c-i801.ko
```

To unload the Linux SMBus `i2c-i801` driver, execute the following command:

```
rmmod i2c-i801.ko
```

The `lsmod` command may be used to confirm if a module has been loaded or unloaded:



```
lsmod | grep i2c_i801 (command uses underscore, not dash)
```

The output of the above command lists modules loaded in the system containing "i2c_i801" as part of their name.

9.7.3 Runtime Configuration of SMBus

Runtime configuration of the SMBus driver is performed from a client driver communicating through the published API set as described in the Intel® EP80579 Software Drivers for Embedded Applications Programmer's Guide and API Reference Manual. Examples of already developed client drivers in the kernel source tree are i2cdetect or i2cdump. Intel leaves the design and development of a client driver to the customer since it is implementation-specific.

9.8 CompactFlash* Driver

9.8.1 Linux Compilation Instructions

All source files for the Linux release of the CompactFlash (CF) driver are located in the following directory within the Linux compatible EP80579 embedded software drivers release:

```
$ICP_ROOT/Embedded/src/CF
```

Compilation of the Linux CF driver separately from the rest of the software package is possible. Change to the *\$ICP_ROOT/Embedded/src/CF* directory and execute the following commands:

```
make clean
make
```

The CF driver compiles and the resulting *leb_cf.ko* file is placed in the *\$ICP_ROOT/Embedded/src/CF/build/linux_2.6/kernel_space* directory.

9.8.2 Linux Module Load/Unload Instructions

To load the Linux CF driver, execute the following command from the directory where the compiled executable resides:

```
insmod leb_cf.ko
```

Note: The driver can also be installed for persistency using the "make install" command from the *\$ICP_ROOT/Embedded/src/CF* directory.

To unload the Linux CF driver, execute the following command:

```
rmmod leb_cf.ko
```

The *lsmod* command may be used to confirm if a module has been loaded or unloaded:

```
lsmod | grep leb_cf
```

The output of this *lsmod* command lists modules loaded in the system containing "leb_cf" as part of their name.



9.8.3 Linux Runtime Configuration of CompactFlash Driver

No runtime configuration is necessary for the CompactFlash driver. Once the component is built and installed, the driver will be running. Note that a CompactFlash card must be mounted in order to use as storage medium, using the following commands:

```
mkdir /mnt/cf  
mount /dev/sdal /mnt/cf
```

The development platform can be configured to boot from CompactFlash. For additional information refer to [Section 11.0, "Bootting from CompactFlash*" on page 60](#).



10.0 Building and Using Crypto Tools

- Cryptotest is an open source user space command that invokes crypto operations using OCF.
- Cryptokeytest is a program to test asymmetric crypto functions

Note: Information on Debugmgr has been moved to the Intel® EP80579 Software for Security Applications on Intel® QuickAssist Technology Programmer's Guide.

10.1 Building Crypto Tools

Build the crypto tools as follows:

1. Save the crypto-tools-20071215.tar.gz file from the following location

<http://heanet.dl.sourceforge.net/sourceforge/ocf-linux/crypto-tools-20071215.tar.gz>

to the \$ICP_ROOT directory and change to that directory:

```
cd $ICP_ROOT
```

2. Extract the crypto-tools package using the following command:

```
tar -xvzf crypto-tools-20071215.tar.gz
```

3. Change directory to the newly extracted crypto-tools directory:

```
cd crypto-tools
```

4. Due to a BSD system call in crypto-tools that is not present in Linux, it is necessary to edit any "strncpy" calls to be "strncpy" calls in the cryptotest.c and cryptokeytest.c files. The following commands can be used for this purpose:

```
sed -i s/strncpy/strncpy/g cryptotest.c  
sed -i s/strncpy/strncpy/g cryptokeytest.c
```

5. Compile the code files:

```
make
```

Note: The cryptokeytest may not compile successfully. Even if cryptokeytest compiles, it will not work because it differs from implementation of the feature in the EP80579 integrated processor. The issue with cryptokeytest does not impact cryptotest.



10.2 Using Crypto Tools

The cryptotest command can be found in the \$ICP_ROOT/crypto_tools directory.

Note: The user must load the OCF stack for this command to work.

The following is the cryptotest command description.

Command:

```
cryptotest [-csbv] [-a algorithm] [count] [size ...]
```

Description:

Runs an encrypt+decrypt or mac operation on a buffer of "size" bytes for a "count" number of iterations. A random key and IV are used.

Options:

- -c check the results
- -d dev pin work on device dev
- -v verbose mode
- -b mark operations for batching
- -p profile kernel crypto operations (must be root)
- -t n fork n threads and run test concurrently

Supported Algorithms:

- -3des Triple DES in CBC mode
- -sha1 SHA-1
- -sha1_hmac HMAC SHA-1

Example:

```
cryptotest -a 3des 100 8192
```

Note: 3des option requires test size to be a multiple of 8; it fails silently if not.



11.0 Booting from CompactFlash*

This chapter describes how to setup and boot the development board from a CompactFlash* (CF) card. Instructions are provided for legacy boot and EFI.

During the installation process, the following tasks will be performed:

- Install CentOS to CompactFlash card
- Create initial Ramdisk image

11.1 System Requirements

Before you begin, you must ensure the following system requirements are met:

- CompactFlash Memory Card Reader/Writer with USB interface
- CompactFlash card. 2GB or more is recommended to allow space for the CentOS image.
- Clean installation of CentOS* 5.2 on hard drive
- CentOS 5.2 installation DVD/CDs media
- Memory stick installed on development board

Note: For legacy boots, both 1 GB and 512 KB memory sticks are supported. For EFI boots, 512 KB memory stick is supported (1 GB is **not** supported). Refer to the Embedded Software release notes for additional information.

- SATA or USB DVD/CD-ROM connected to development board
- Network connection to internet (required to obtain elilo if using EFI boot)

11.2 Procedure

Pre-boot Firmware

1. Verify BIOS 64 is installed on the development board. The Project Version in the Main BIOS settings should have the value TRXTG 64.00.
2. Verify SATA mode is set to AHCI mode in the BIOS. This setting is available on the Advanced tab under IDE Configuration.
3. Update BIOS settings to boot from DVD/CD ROM drive. The setting is available on the Boot tab.
4. Shutdown the development board.

Install CentOS 5.2 from DVD/CD Media onto CompactFlash card

5. Plug in the CompactFlash Memory Card Reader/Writer and connect this to USB port on the development board.
6. Disconnect the SATA hard drive from the development board.
7. Place the CentOS DVD or CD #1 in your DVD/CD-ROM drive and boot the development board from the DVD/CD-ROM.



8. Begin installing CentOS onto CompactFlash media.
9. When the message "Installation requires partitioning of your drive." is displayed, select "Remove all partitions on selected drives and create default layout."
10. Click the "Review and modify partitioning layout" checkbox.
11. Click "Next"
12. Remove Logical Volume Manager partitions. This can be done by:
 - Select the VolGroup00 and click Delete button.
 - c. Select /dev/sda2 LVM PV type partition and click Delete button.
 - d. Repeat for any additional LVM partitions on the drive.

Complete these steps (13. through 14.) for EFI Boot

13. Remove the /boot partition. This can be done by:
 - Select /boot partition and click the Delete button.
14. Create "/" and "/boot/efi" partitions. "/boot/efi" partition must have format type set to "vfat". The CentOS CompactFlash drive partition could look similar to:

/dev/sda1	/	ext3	1866
/dev/sda2	/boot/efi	vfat	101

Note: These partition sizes assume a 2 GB CompactFlash card.

Complete these steps (15. through 16.) for Legacy Boot

15. Create "/" partition. This can be done by:
 - a. Select the free space created by deletion of the LVM partition(s).
 - b. Click New.
 - The mount point should be "/", the file system type should be left at "ext3" and then select "Fill to maximum available size".
16. Leave the /boot partition as ext3

Complete these steps (17. through 20.) for both Legacy and EFI Boot

17. Click Next

Note: You will receive a warning notice that a swap partition was not specified. This warning can be ignored. Select "Yes" to continue with your requested partitioning.
18. Select the bootloader when prompted.
 - a. For Legacy Boot, select "The Grub boot loader will be installed"
 - b. For EFI Boot, select "No boot loader will be installed"
19. Use a minimal set of packages – it is likely a KDE or GNOME desktop environment will not fit onto CompactFlash card. It is highly recommended to deselect any additional packages from the install.
20. After installation to the CompactFlash card, reconnect the hard drive and boot CentOS from the hard drive.

Build CompactFlash Driver

Note: These steps can be skipped if the top level make and make install commands were executed in [Section 4.3, "Build Using Top Level Make" on page 30](#).

21. Environmental Setup

```
$ export ICP_ENV_DIR=/EP805XX_release/Embedded
$ export ICP_ROOT=/EP805XX_release
```



22. Build CompactFlash Driver

```
$ cd /EP805XX_release/Embedded/src/CF
$ make
```

23. Copy CompactFlash into Modules.

```
$ mkdir -p /lib/modules/2.6.18-92.el5/kernel/drivers/CF
$ cp /EP805XX_release/Embedded/src/CF/build/linux_2.6/kernel_space/leb_cf.ko /lib/
modules/2.6.18-92.el5/kernel/drivers/CF/
```

Create Initial Ramdisk

24. Make the initial ramdisk with CompactFlash driver.

```
$ cd /EP805XX_release
$ mkinitrd --preload=leb_cf initrd-2.6.18-cf.img 2.6.18-92.el5
```

Populate CompactFlash

25. Plug in CompactFlash Memory Card Reader/Write adapter and connect to development board.

Complete these steps (26. through 30.) for EFI Boot

26. Download elilo-3.6-ia32.efi from: http://sourceforge.net/project/showfiles.php?group_id=91879&package_id=97044

Expand the 3.6 Release to locate file. This download can be performed on the development board.

27. Rename the file to elilo.efi

28. Copy elilo.efi onto /boot/efi (vfat) partition of CompactFlash card.

29. Copy /EP805XX_release/initrd-2.6.18-cf.img onto /boot/efi (vfat) partition of CompactFlash card.

30. Create elilo.conf file on /boot/efi (vfat) partition of CompactFlash.

```
verbose=5
default=linux
image=vmlinuz-2.6.18-92.el5
label=linux
initrd=initrd-2.6.18-cf.img
read-only
root=/dev/cfa1
```

Note: Root is set to /dev/cfa1 because during installation of CentOS 5.2 to CompactFlash the first partition was formatted as /.

Complete these steps (31. through 32.) for Legacy Boot

31. Copy /EP805XX_release/initrd-2.6.18-cf.img onto /boot partition of CompactFlash card.

32. Edit /boot/grub.conf on /boot/grub partition of CompactFlash and replace "initrd/initrd-2.6.18-92.el5.img" with "initrd/initrd-2.6.18-cf.img"

Boot from CompactFlash

33. Shut down the development board and disconnect SATA hard drive from development platform.

34. Plug the CompactFlash card into development board (not the CompactFlash Memory Card Reader/Write adapter).

35. Modify SATA mode to Legacy mode in the BIOS. This setting is available on the Advanced tab under IDE Configuration.



For Legacy Boot the required work is complete. Booting the system now will begin loading CentOS from CompactFlash. Instructions for EFI Boot are contained below.

Complete these steps (36. through 38.) for EFI Boot

36. Boot to system to the EFI shell. You may need to update the Boot order in BIOS to select EFI shell as first boot device. The setting is located on the Boot tab in the BIOS settings.

37. Switch to fs0.

```
Shell> fs0:
```

38. From EFI shell, run elilo.

```
fs0:\> elilo
```

This command initiates the CentOS boot.



12.0 Sample Applications

Sample applications are included with this software release that present examples of how to use cryptographic APIs. They can be found in the directory:
./Acceleration/library/icp_crypto/look_aside_crypto/src/sample_code

The directory structure is as follows:

```
./Acceleration/library/icp_crypto/look_aside_crypto/src/sample_code
.../functional
.../functional/asm
.../functional/asm/diffie_hellman_sample
.../functional/asm/prime_sample
.../functional/common (This folder is FreeBSD-specific. Others occur in both OSes.)
.../functional/include
.../functional/sym
.../functional/sym/alg_chaining_sample
.../functional/sym/cipher_sample
.../functional/sym/hash_sample
.../performance
```

For instructions on using the sample applications, refer to README.txt located in:
./Acceleration/library/icp_crypto/look_aside_crypto/src/sample_code



13.0 Pre-Boot Firmware (BIOS)

The pre-boot firmware is executed when the system is powered up or reset. It initializes and configures system memory, devices and interfaces.

The pre-boot firmware is based on the AMI Aptio* 4.5 core and compliant to EFI v1.1. The firmware is stored in the Firmware Hub (FWH) or Serial Peripheral Interface (SPI) Flash; the FWH or SPI Flash can be updated using a flash utility tool that is provided.

The pre-boot firmware setup menu can be used to view and modify the system settings for the development board. The setup menu is accessed by pressing the key during pre-boot firmware boot up (before the operating system begins). The setup menu bar is shown in [Table 3](#).

13.1 Pre-Boot Firmware Setup Menu

[Table 3](#) shows the pre-boot firmware setup main menu and provides a brief description of each menu option. [Table 4](#) shows the action keys that can be used when navigating and selecting options from pre-boot firmware menus.

Table 3. Pre-Boot Firmware Setup Main Menu

Main	Advanced	Chipset	Security	Boot	Exit
Displays processor and memory configuration Setup for CMOS system date and time	Configures advanced features and settings	Configures different major components	Setup passwords and security features	Selects boot options and configurations	Saves or discards changes to setup program options

Table 4. Pre-Boot Firmware Setup Program Action Keys

Function Key	Description
< or >	Moves cursor left or right in the main menu
^ or v	Moves cursor up or down to select sub-menu items
Enter	Executes command or selects the submenu
F7	Discard changes
F8	Load the fail-safe default
F9	Load the optimal default configuration value for the current menu
F10	Save the current configuration and exit the setup menu
ESC	Exit the setup menu



13.1.1 Serial Console Redirection

The pre-boot firmware supports redirection of both video and keyboard via a serial port. When console redirection is enabled, the remote console terminal sends keystrokes to the development board pre-boot firmware, and the pre-boot firmware redirects the video to the console terminal.

As an option, the development board can be operated without keyboard or video and can run entirely via the remote serial console. This includes accessing the pre-boot firmware setup menu.

Console redirection ends when operating system boot up begins. After boot up begins, the operating system is responsible for continuing the redirection.

Note: The pre-boot firmware console redirection is text only. Graphical data, such as logos, are not redirected.

Table 5 shows the default settings of the serial console redirection.

Table 5. Serial Console Redirection Default Settings

Parameter	Default
Port Number	COM 1
Baud Rate	115200
Data Bits	8
Parity	None
Stop bits	1
Flow Control	None

13.1.2 Changing the Boot Device

Use the following procedure to change the boot device:

1. Press the key during POST to enter the pre-boot firmware setup menu.
2. Use the arrow keys to navigate to the <BOOT> menu.
3. Move the cursor to <Boot Device Priority>.
4. Select the desired booting sequence list.

Note: Follow the instructions on the right side of the pre-boot firmware screen to navigate and change pre-boot firmware settings.

13.1.3 Maximum Memory Speed Setup

The maximum memory speed supported on the development board can be selected using the Maximum Memory Speed Setup option available in the BIOS Setup Menu on the Chipset tab.

Enter the BIOS Setup Menu and select the Chipset tab. Select the North Bridge sub tab. Navigate down to the bottom option, titled Max Memory Speed Support, and select this option using the Enter button. A selection box appears providing the following options:

- 400 MHz
- 533 MHz
- 667 MHz
- 800 MHz



The default setting in the BIOS is 400 MHz. If a higher speed memory DIMM is inserted into the development board, the corresponding memory speed must be selected in the BIOS Setup Menu to support the intended speed. Otherwise, the memory is reduced to the default of 400 MHz.

13.1.4 Coherent and Non-Coherent Memory Allocation

The development board supports allocation of memory regions for coherent and non-coherent use. For more information on these regions, refer to the Intel® EP80579 Integrated Processor Product Line Datasheet, Section 3.0.

Coherent and non-coherent memory use features are for development boards that use the Intel® EP80579 Integrated Processor with Intel® QuickAssist Technology. EP80579 integrated processors **without** Intel® QuickAssist Technology do not make use of the memory set aside for these features.

This software package requires the pre-boot firmware (BIOS) for your hardware to allocate the values for each region.

To allocate memory regions, perform the following steps:

1. Enter the BIOS Setup Menu and select the Chipset tab.
2. Select North bridge, and navigate down towards the bottom to the Coherent Mem Size option and press Enter to select this option. A dialog box is displayed prompting the user to enter a value.
Type the numerical value "2000" and press Enter.
3. Navigate to the next option, Non-Coherent Mem Size, and press Enter to select this option. A dialog box is displayed prompting the user to enter a value.
Type the numerical value "2000" and press Enter.

13.1.5 Legacy and AHCI SATA Mode

The development board supports hard drives in legacy SATA mode and in Advanced Host Controller Interface (AHCI) mode. AHCI mode provides advanced capabilities and improved performance, provided the hard drive supports the following features:

- Hot Plug
- Native Command Queuing
- Speeds up to 3 Gb/s

Refer to the Serial ATA Organization web site for more information:

<http://www.serialata.org/>

The development board pre-boot firmware (BIOS) can be configured in either Legacy or AHCI mode as desired. The BIOS defaults to Legacy mode because not all hard drives support AHCI. To toggle the BIOS to either Legacy or AHCI mode, proceed as follows:

1. Press the key during POST to enter the pre-boot firmware setup menu.
2. Use the arrow keys to navigate to the Advanced menu.
3. Use the arrow keys to navigate to the IDE Configuration option.
4. Select the IDE Configuration option.
5. Use the arrow keys to navigate to the SATA Mode option.
6. Press the Enter key. A SATA Mode popup window appears.
7. Select either Legacy or AHCI as desired. Do not use Native as a selection.



8. Press F4 to save.
9. Choose Yes. The system continues the boot process.

13.2 Pre-Boot Firmware Image Reflashing Instructions

One method is available for updating the pre-boot firmware flash images located on the development board FWH or SPI Flash:

- AFUEFI Flash Recovery

It is possible that updated pre-boot firmware images will become available from Intel for the Intel® EP80579 Integrated Processor with Intel® QuickAssist Technology Development Board. The latest pre-boot firmware image is available from Intel's public web site, <http://www.intel.com/go/soc> located with all other collateral related to the EP80579 integrated processor.

If the pre-boot firmware image should become corrupted on the board, also utilize these instructions to reflash the image.

13.2.1 Aptio Flash Update Utility (AFUEFI)

Use the following instructions to update the development board pre-boot firmware image using a USB flash drive and the Aptio Flash Update Utility from AMI.

Necessary hardware:

- development board
- Socketed Firmware Hub or SPI Flash
- USB flash drive

Necessary software:

- development board pre-boot firmware image
- Aptio Flash Update Utility from AMI - AFUEFI

Steps to reflash the image:

1. Execute the AFUEFI utility onto the USB flash drive.
2. Load the pre-boot firmware image onto the USB flash drive.
3. Change the boot setting in the BIOS Setup Selection to boot from the EFI shell. Boot the development board to the EFI shell.
4. Insert the USB flash drive into the USB port.
5. Once the USB flash drive is recognized on the system (activity can be seen on the USB flash drive LED if present), several commands are available as follows:
 - Type "map -r" to list all devices available.
 - Type "fs0:" to enter USB device.
 - Type "ls" to list all files.
6. Once the "fs0:" command has been initiated, execute the AFUEFI utility. Enter:

```
AFUEFI <pre-boot firmware image name> /X /P /B /N
```

where the <pre-boot firmware image name> will be TRXTG063.ROM or similar

7. Reboot the development board when reflashing has completed.
8. Confirm that the image has been updated to the reflashed image by checking the Flash Image identity in the BIOS Setup.



14.0 Uninstalling the Software

Please refer to instructions on loading and unloading individual modules in [Section 9.1](#) through [Section 9.7](#).

14.1 Linux Module/Driver Dependencies

[Table 6](#) lists the dependencies for the driver modules or patch within the EP80579 security software package. OS installation is assumed.

Table 6. Linux Module/Driver Dependencies

Module/Driver/Patch	Dependency 1	Dependency 2
EP80579 integrated processor Linux Patch	OS Source	None
OCF	OS Source	None
Openswan	OpenSSL	GMP library
cryptosoft	ocf	
cryptodev	ocf	
icp_ocf	ocf	LAC
icp_asd	icp_crypto	icp_debug
icp_crypto	icp_services	icp_debug
icp_services	icp_hal	icp_debug
icp_debugmgmt	icp_debug	
Controller Area Network (CAN)	None	-
Enhanced DMA (EDMA)	None	-
IEEE 1588 Hardware Assist (1588)	None	-
General Purpose IO (GPIO)	None	-
Gigabit Ethernet (GbE)	GCU	None
Global Configuration Unit (GCU)	None	-
SMBus (i2c-i801)	i2c-core	i2c-dev
Watchdog Timer (WDT)	None	-



15.0 Troubleshooting

Refer to the Release Notes for your software package for a list of known errata, implications, and workarounds.

15.1 Using a Graphics Card

Some LCD monitors do not function properly with the development board. In one case, an “Input Not Supported” message was reported upon boot completion. Try an alternative LCD monitor if video is not displayed.

15.2 Using the Serial Port

The user can bring up the system without using the Graphics card. In this case, the IO takes place over the serial port. The serial port is configured as follows:

1. Enable communication over all serial ports in the BIOS setup. Configure ports to 115200, No Parity, 8 bit, 1 stop bit and No Flow Control.
2. Use a female-female 9-pin RS232 cable to connect the development board to a terminal emulator application such as HyperTerminal* in Windows. Configure the terminal emulator to 115200, No Parity, 8 bit, 1 stop bit and No Flow Control.
3. Edit the file /etc/ttys as follows:
 - Locate entries: `tttyd[0123] "/usr/libexec/getty std.9600" dialup off secure`
 - Change the above to: `tttyd[0123] "/usr/libexec/getty std.115200" vt100 on secure`
4. Edit the file /boot/loader.conf and ensure it contains the following entries:
 - `comconsole_speed="115200"`
 - `boot_serial="YES"`
 - `boot_multicons="YES"`
 - `console="comconsole"`
5. Ensure that the file /boot/loader.rc contains the following entries:
 - `include /boot/loader.4th`
 - `start`
6. To enable communication over the serial port, use the command:

```
echo "-Dh" > /boot.config
```
7. Reboot for changes in /boot/loader.conf to take effect
8. Kill -HUP 1 to reread /etc/ttys by getty

Note: Changing from the default sio0 COM device (the one on the left when you face the platform) is complex and requires editing of /etc/make.conf file and rebuilding of bootloader and kernel.



9. Edit grub.conf:

The bootloader enables Grub output on the serial port and console logging via the serial port.

- a. Make a backup of /etc/grub.conf before doing any changes.
- b. Edit that file and just underneath "timeout=5" add:

```
serial --unit=0 --speed=115200 --word=8 --parity=no --stop=1
terminal --timeout=0 serial console
```

At the end of each "kernel" line append:

```
console=tty0 console=ttyS0,115200
```

- c. If the user does not want a graphic card, they can boot without one, provided they modify the grub.conf file and comment out the line:

```
splashimage=(hd0,0)/grub/splash.xpm.gz
```

Otherwise, if no graphics card is available, the system will hang at boot.

10. Edit /etc/inittab (enables shell login from the serial port):

- a. Make a backup of /etc/inittab before making any changes and edit the file.
- b. Just below the line "Run gettys in standard runlevels", add the following line:

```
so:2345:respawn:/sbin/agetty -L -f /etc/issuesserial 115200 ttyS0 v 115200
vt100-nav
```

11. You may have to edit /etc/securetty and add the lines:

```
ttyS0
ttyS1
```

12. Remove the graphics card and restart the system.



16.0 Glossary

AHCI	Advanced Host Controller Interface
ASD	Acceleration System Driver
ASU	Acceleration Services Unit
CAN	Controller Area Network
CDRAM	Coherent DRAM
DIMM	Dual Inline Memory Module
EDMA	Enhanced Direct Memory Access
EFI	Extensible Firmware Interface
FWH	Firmware Hub
GbE	Gigabit Ethernet
GCU	Global Configuration Unit
GPIO	General Purpose Input Output
IEEE	Institute of Electrical and Electronics Engineers
IHS	Integrated Heat Spreader
NCDRAM	Non-Coherent DRAM
OCF	OpenBSD Cryptographic Framework
OCF Patch	Patch file used to include OCF files during kernel rebuild
OCF Shim	A kernel module that allows the OCF module to offload cryptographic functions to EP80579 cryptographic accelerators.
POST	Power On Self Test
RNG	Random Number Generator
SATA	Serial Advanced Technology Attachment
SKU	Stock Keeping Unit
SMBus	System Management Bus
TIM	Thermal Interface Material
WDT	Watchdog Timer

