

Intel[®] EP80579 Software for Security Applications on Intel[®] QuickAssist Technology for FreeBSD*

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	8
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 FreeBSD 7.1	20
3.3	Installing FreeBSD	20
4.0	Rebuilding FreeBSD Kernel	23
4.1	Build Environment Requirements	23
4.2	Unpacking EP80579 FreeBSD Security Package	23
4.3	Patching the Kernel for PCI Device Recognition	23
4.4	Enabling IPSec in the FreeBSD Kernel (Optional)	24
4.5	Rebuilding the FreeBSD Kernel	25
4.6	Configure Loader Configuration File	25
4.7	Rebooting and Verifying	25
5.0	Building EP80579 Security Software on a Target Development Board	26
5.1	Environment Setup	26
5.2	Build Options	26
5.3	Build Using Top Level Make	26
5.4	Installation of Build Output	27
5.4.1	Statistics Collection	27
5.5	Uninstalling Embedded and Security Kernel Modules	29
6.0	Runtime Configuration	30
6.1	Loading Opencrypto Driver	30
6.2	OpenSSL Updates	30
6.3	Using the Intel® QuickAssist Technology Cryptographic API	31



7.0	Racoon* Installation and Configuration	32
7.1	Racoon Installation	32
7.2	Racoon Configuration	33
7.2.1	racoon.conf	33
7.2.2	psk.txt	34
7.2.3	spd.sh	34
7.2.4	spd_flush.sh	34
7.2.5	ipsec.conf	35
7.3	Configuring Sample VPN Application	35
7.3.1	Set Up Left Development Board	36
7.3.2	Set Up Right Development Board	36
7.3.3	Set Up VPN	37
7.4	IPSec Performance	38
8.0	Building Components Individually	39
8.1	Building Components Individually Using Top-Level Make	39
9.0	Building, Installing and Loading Individual EP80579 Embedded Software Drivers	40
9.1	Controller Area Network (CAN) Driver	40
9.1.1	FreeBSD Compilation Instructions	40
9.1.2	FreeBSD Module Load/Unload Instructions	40
9.1.3	FreeBSD Sample Codelet	41
9.2	Enhanced Direct Memory Access (EDMA) Driver	41
9.2.1	FreeBSD Compilation Instructions	41
9.2.2	FreeBSD Module Load/Unload Instructions	41
9.2.3	FreeBSD Sample Codelet	42
9.3	Watchdog Timer (WDT) Driver	42
9.3.1	FreeBSD Compilation Instructions	42
9.3.2	FreeBSD Module Load/Unload Instructions	42
9.3.3	FreeBSD Sample Codelet	43
9.4	General Purpose I/O (GPIO) Driver	43
9.4.1	FreeBSD Compilation Instructions	43
9.4.2	FreeBSD Module Load/Unload Instructions	43
9.5	IEEE 1588 Hardware Assist Driver	44
9.5.1	FreeBSD Compilation Instructions	44
9.5.2	FreeBSD Module Load/Unload Instructions	44
9.5.3	FreeBSD Sample Codelet	45
9.6	Global Configuration Unit and Gigabit Ethernet Drivers	45
9.6.1	FreeBSD Compilation Instructions	45
9.6.2	FreeBSD Module Load/Unload Instructions	46
9.7	System Management Bus (SMBus) Driver	46
9.7.1	FreeBSD Compilation Instructions	46
9.7.2	FreeBSD Module Load/Unload Instructions	46
10.0	Building and Using Crypto Tools	48
10.1	Using Crypto Tools	48
11.0	Sample Applications	49
12.0	Pre-Boot Firmware (BIOS)	50
12.1	Pre-Boot Firmware Setup Menu	50
12.1.1	Serial Console Redirection	51
12.1.2	Changing the Boot Device	51
12.1.3	Maximum Memory Speed Setup	51
12.1.4	Coherent and Non-Coherent Memory Allocation	52
12.1.5	Legacy and AHCI SATA Mode	52
12.2	Pre-Boot Firmware Image Reflashing Instructions	53



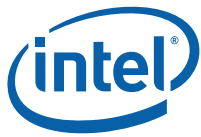
12.2.1 Aptio Flash Update Utility (AFUEFI)	53
13.0 Uninstalling the Software	54
13.1 FreeBSD Modules/Driver Dependencies	54
14.0 Troubleshooting	55
14.1 Using a Graphics Card	55
14.2 Using the Serial Port	55
15.0 Glossary	57

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.....	35

Tables

1 Development Board - Key Components and Connectors Legend.....	17
2 Output Files Created in the \$ICP_BUILD_OUTPUT Directory	27
3 Pre-Boot Firmware Setup Main Menu	50
4 Pre-Boot Firmware Setup Program Action Keys	50
5 Serial Console Redirection Default Settings.....	51
6 FreeBSD Module/Driver Dependencies	54



Revision History

Date	Revision	Description
November 2009	001	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 marked 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.

Note: Note that copying FreeBSD commands with quoted strings from this pdf file to a shell script may result in incorrect quotes. The user should edit the shell scripts appropriately.

1.2 Additional Information on Software

The EP80579 security software release package for FreeBSD has been validated with FreeBSD v7.1.

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 FreeBSD* 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
FreeBSD 7.1 ISO	ftp://ftp-archive.freebsd.org/pub/FreeBSD-Archive/old-releases/i386/ISO-IMAGES/7.1
FreeBSD Installation Guide, Readme, etc	ftp://ftp-archive.freebsd.org/pub/FreeBSD-Archive/old-releases/i386/7.1-RELEASE

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

Note: Hashing Algorithms have been verified to work when Racoon is configured to use them to pass traffic in kernel space. The algorithms have not been validated in user space.

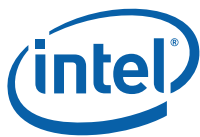
- Authentication Schemes:
 - AES-XCBC-MAC-96
 - HMAC-MD-5
 - HMAC-SHA-1
 - HMAC-SHA-2 (224, 256, 384, 512)
- Chaining Cipher and Authentication algorithms

- Opencrypto Driver

This is a kernel module that communicates with the FreeBSD Opencrypto framework 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 Opencrypto 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

- 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
- 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

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/sample_code/ directory.

```
./Acceleration
./Acceleration/firmware
./Acceleration/include
./Acceleration/include/dcc
./Acceleration/include/lac
./Acceleration/drivers
./Acceleration/drivers/icp_asd
./Acceleration/library
./Acceleration/library/common
./Acceleration/library/icp_debug
./Acceleration/library/icp_utils
./Acceleration/library/icp_utils/OSAL
./Acceleration/library/icp_services
./Acceleration/library/icp_crypto
./Acceleration/library/icp_crypto/QATAL
./Acceleration/library/icp_crypto/QAT_FW
./Acceleration/library/icp_crypto/look_aside_crypto
./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/freebsd
./Acceleration/library/icp_crypto/look_aside_crypto/src/sample_code
./Acceleration/library/icp_crypto/look_aside_crypto/include
./Acceleration/library/icp_debugmgmt
./Acceleration/shims/
./Acceleration/shims/OCF_Shim
./Embedded
./Embedded/src
./Embedded/src/patches
./Embedded/src/WDT
./Embedded/src/GCU
```



```
./Embedded/src/EDMA
./Embedded/src/CAN
./Embedded/src/GPIO
./Embedded/src/1588
./Embedded/src/GbE
./Embedded/codelet
./Embedded/codelet/WDT
./Embedded/codelet/EDMA
./Embedded/codelet/CAN
./Embedded/codelet/1588
./FreeBSDConfigFiles
./FreeBSDConfigFiles/left
./FreeBSDConfigFiles/right
./build_system
./build_system/build_files
./OpenSourcePatches/
```



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.

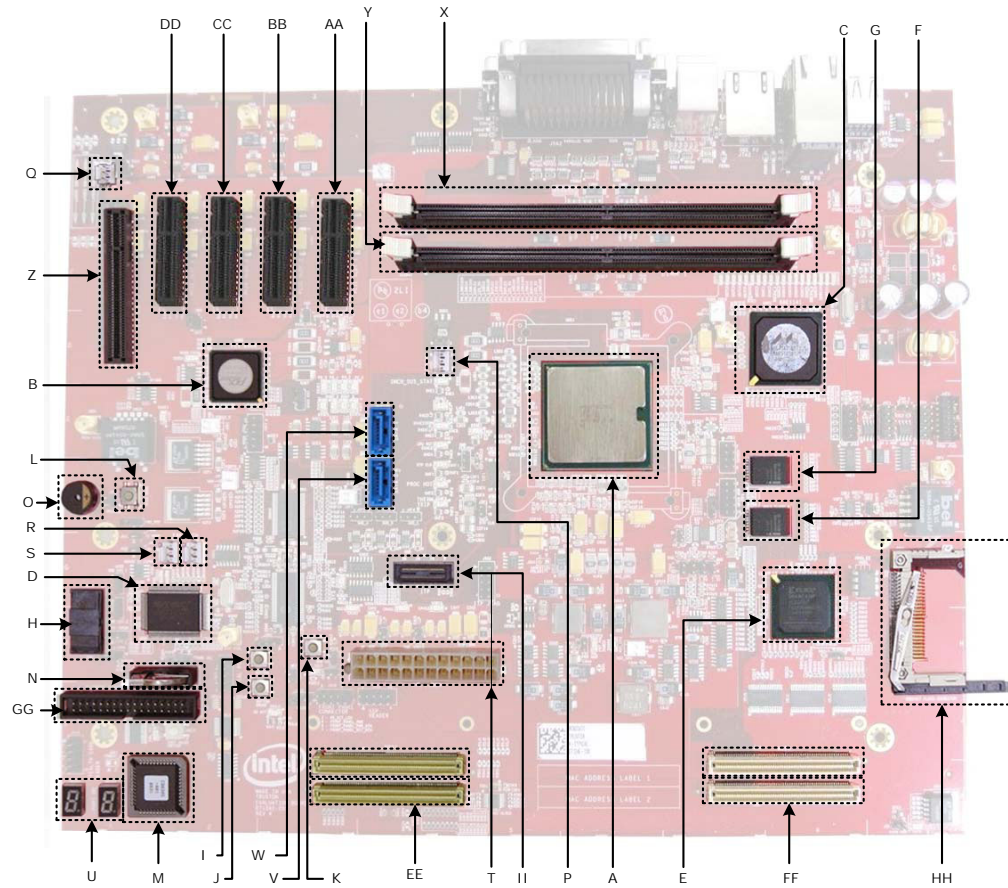
Use the following procedure to power up the development board:

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](#).

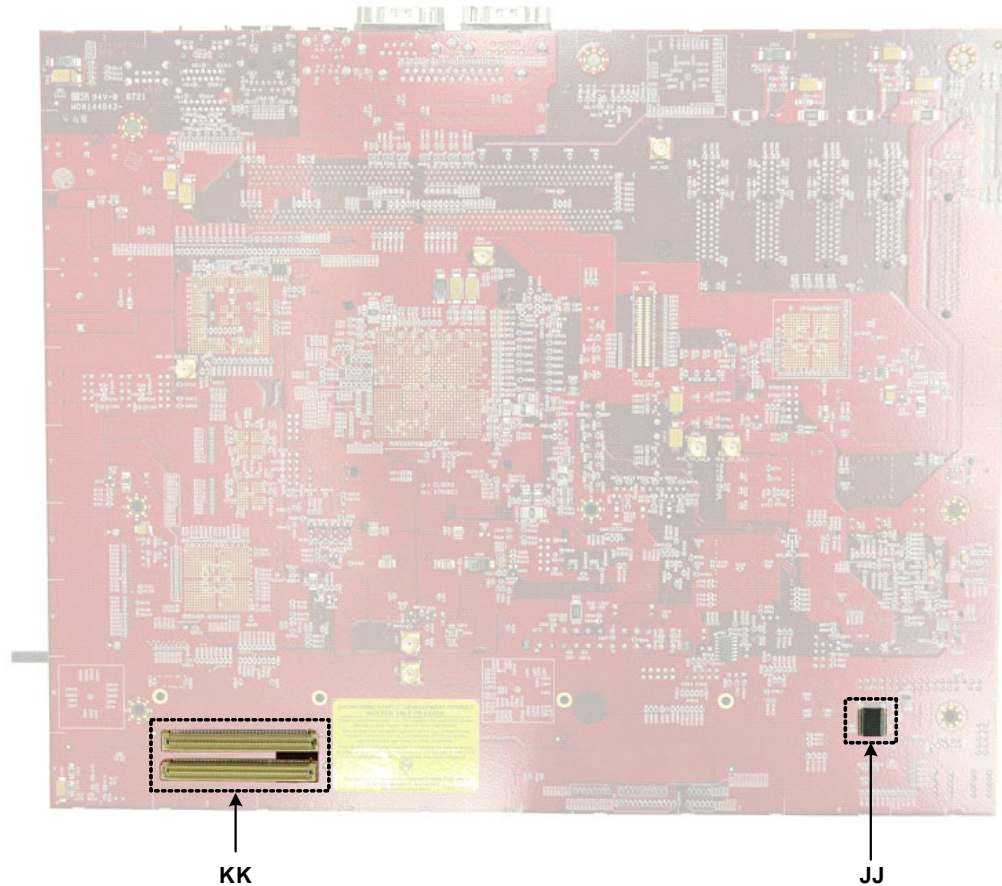


Figure 1. Development Board - Top View of Component and Connector Locations



B6607-02

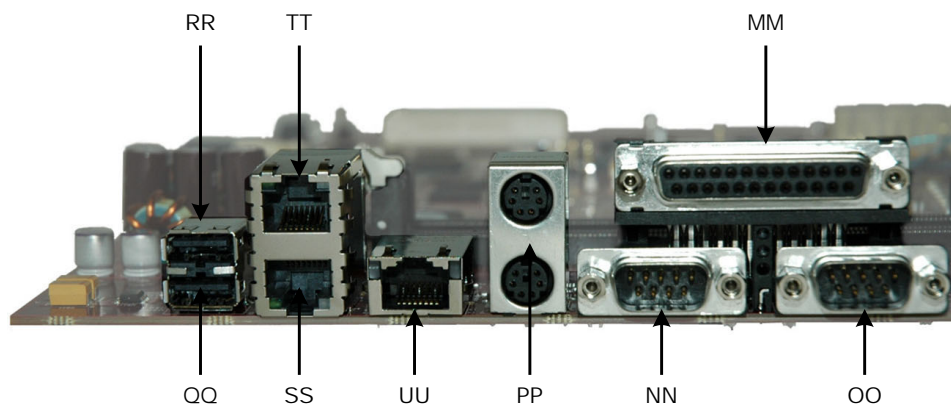
Figure 2. Development Board - Bottom View of Component and Connector Locations



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)**

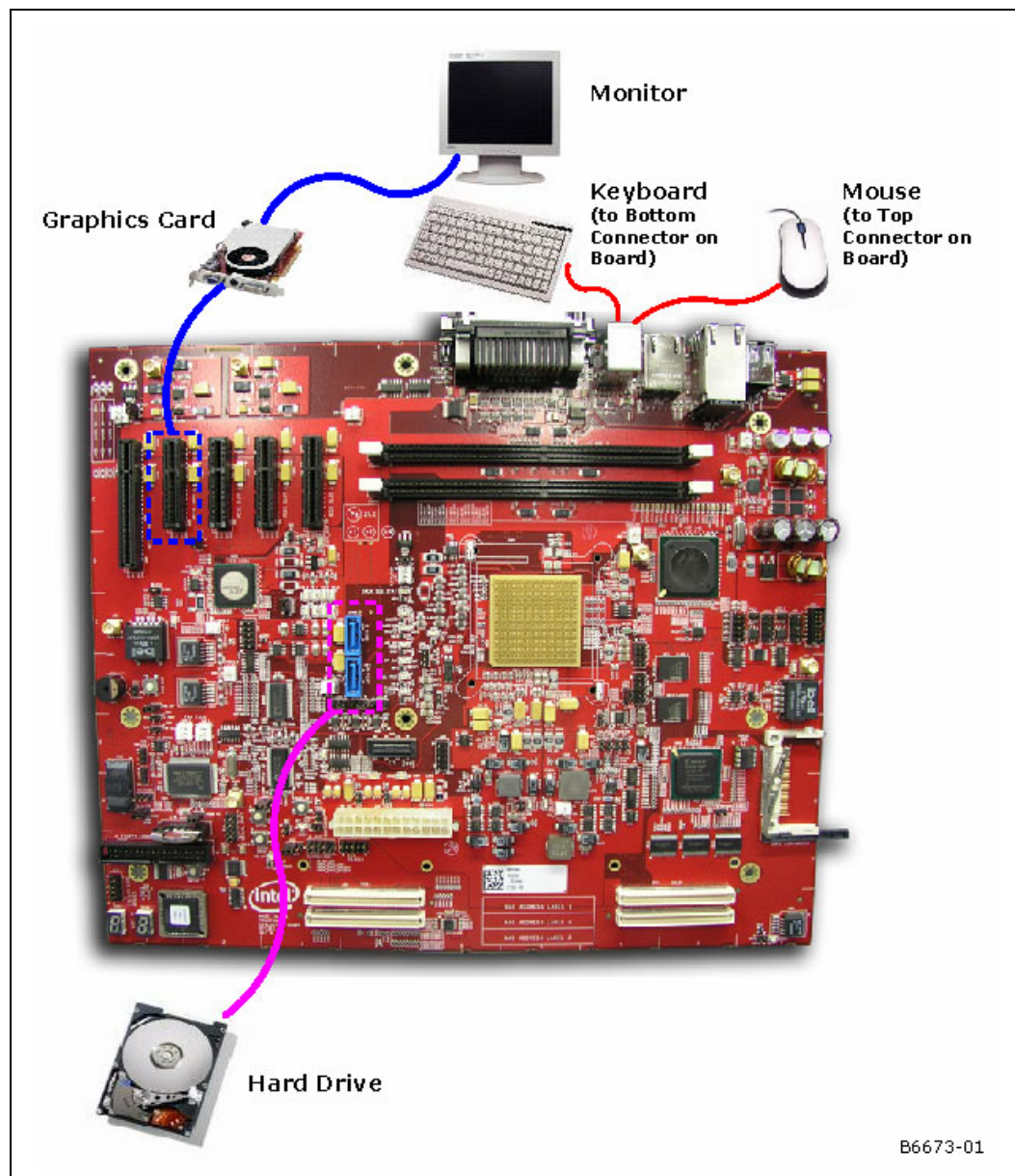
Callout	Component/Connector
V	SATA port 0
W	SATA port 1
X	DDR2 DIMM0
Y	DDR2 DIMM1
Z	Slot 0 x8 connector 4 lanes PCI Express
AA	Slot 1 x4 connector 1 lane PCI Express
BB	Slot 2 x4 connector 1 lane PCI Express
CC	Slot 3 x4 connector 1 lane PCI Express
DD	Slot 4 x4 connector 1 lane PCI Express
EE	Mezzanine 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 FreeBSD* 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

3.1 System Requirements

Please consult the FreeBSD minimum hardware requirements in the online documentation at <http://www.freebsd.org/releases/7.1R>. The development board meets these requirements.

Execute the following three steps:

- Get the latest BIOS image and install it as described in [Section 12.2.1, “Aptio Flash Update Utility \(AFUEFI\)” on page 53](#)
- Set the Legacy or AHCI SATA mode as described in [Section 12.1.5, “Legacy and AHCI SATA Mode” on page 52](#)
- Allocate the Coherent and Non-Coherent Memory as described in [Section 12.1.4, “Coherent and Non-Coherent Memory Allocation” on page 52](#)

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

3.2 Acquiring FreeBSD 7.1

FreeBSD* download software can be acquired from <ftp://ftp-archive.freebsd.org/pub/FreeBSD-Archive/old-releases/i386/ISO-IMAGES/7.1>.

For complete installation instructions and the FreeBSD 7.1 release Readme document, please refer to FreeBSD’s web site at the following locations:

- <ftp://ftp-archive.freebsd.org/pub/FreeBSD-Archive/old-releases/i386/7.1-RELEASE>

3.3 Installing FreeBSD

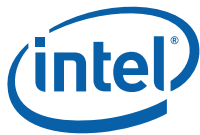
The following are the instructions that Intel has used to install FreeBSD on the development board.

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 [Section 12.0, “Pre-Boot Firmware \(BIOS\)” on page 50](#).
2. Power on the system with FreeBSD 7.1 disc #1 in the CDROM. The system should begin to boot from the FreeBSD installation disc #1.
3. The user is shown a list of Boot options. Select: 1 - Boot FreeBSD [default]
4. Select the appropriate Country from the next list.



5. The user is shown a list of options for software installation. Select: "Standard - Begin a Standard installation (recommended)".
6. In the next few screens, the user will be asked to create a DOS-Style partition on the disk. There may be warnings about geometry being incorrect; however, these error messages can be ignored. Select: "A = Use Entire Disk"; and then confirm by selecting: Q = Finish.
7. Next install a Boot Manager. Select: "Standard - Install a standard MBR (no boot manager)".
8. Create default sized BSD partitions inside the fdisk partition, created in a previous step. Select: A = Auto Defaults; and then confirm by selecting: Q = Finish.
9. Choose the distribution to use. Select: "4 Developer -- Full sources, binaries and doc but no games".
10. Answer [Yes] to the question: Would you like to install the FreeBSD ports collection?
11. Note the user is taken to the previous screen showing the various distributions. Distribution was selected in previous step so select: Exit - Exit this menu (returning to previous menu) to continue the installation.
12. Identify the installation media. Select: "1 CD/DVD - Install from a FreeBSD CD/DVD".
13. The system warns you of loss of data in the disk. To continue with the installation, select "Yes". File systems are created and files copied. This step will take a few minutes to complete and display a Congratulations message at the end.
14. Answer [No] to the question: "Would you like to configure any Ethernet or SLIP/PPP network devices?"
15. Answer [Yes] to the question: "Do you want this machine to function as a network gateway?"
16. In the next few screens inetd and related services can be enabled. Do not edit the default configuration file. The questions can be deceptive. Select Yes and No carefully.
17. Answer [Yes] to the question: "Would you like to enable the SSH login?"
18. Answer [No] to the question: "Do you want to have anonymous FTP access to this machine?"
19. Answer [No] to the question: "Do you want to configure the machine as an NFS server?"
20. Answer [No] to the question: "Do you want to configure this machine as a NFS client?"
21. Answer [No] to the question: "Would you like to customize your system console settings?"
22. In the next few screens, the machines time zone is set. Carefully select Yes and No to these questions. Some are counter intuitive and install does not give the ability go back and correct.
23. Answer [No] to the question: "Would you like to enable Linux binary compatibility?"
24. In the next few screens, the Mouse can be configured. When finished, exit and go back to previous screen to continue.
25. The system informs you of the availability of ready-to-run applications. Select YES to select applications from the following:

Note: If the list of applications is not displayed, your installation sources may be incomplete. These items are required.



- a. Select all items under: devel - Software development utilities and libraries
- b. Select preferred editors under: editors - Common text editors
- c. Select: ftp - ftp client and server utilities
- d. Select Perl5 scripting tool under: perl5
- e. Select bash shell under: shells

Note: Numerous additional entries are selected as dependents of above choices.

- 26. Select Install. This will install the packages selected in the previous step.
- 27. Add a user account if required. Set the root password.
- 28. Install will give an opportunity to set any last options. Select and set any options may have missed.
- 29. Select "Exit Install" to complete the installation.
- 30. At this point system will reboot. If the BIOS Boot Priority Option is set to select the Hard Disk in the absence of CD ROM, then the installed FreeBSD image will start and display the login prompt. Login using the root or user password.



4.0 Rebuilding FreeBSD Kernel

4.1 Build Environment Requirements

FreeBSD 7.1 includes GCC 4.2.1. Refer to the FreeBSD 7.1 Release Notes for a list of contributed software. The FreeBSD 7.1 Release Notes for the i386 release can be found at the following location:

<ftp://ftp-archive.freebsd.org/pub/FreeBSD-Archive/old-releases/i386/7.1-RELEASE>

4.2 Unpacking EP80579 FreeBSD Security 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.

```
# mkdir /EP805XX_release
# cd /EP805XX_release
# setenv ICP_ROOT $PWD
```

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:

Tip: To mount a USB memory stick, issue the following commands:

```
# mkdir /mnt/usb (Skip this command if directory exists)
# mount -t msdosfs /dev/da0sl /mnt/usb
# cp /mnt/usb/{Path to security package}/Security.B.1.0.2-xxx.tar.gz /
EP805XX_release
```

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.B.1.0.2-xxx.tar.gz
```

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

Note: The Makefiles in FreeBSD are called BSDMakefile.

4.3 Patching the Kernel for PCI Device Recognition

Execute the following commands to patch the FreeBSD kernel:



```
# cd $ICP_ROOT
# cd ./Embedded/src/patches
# patch -p0 < Intel_EP80579_FreeBSD_71.patch
```

Upon patch application, review the output. The output should contain text similar to:

```
Hunk #1 succeeded at 1814...
```

This indicates that this part of the patch was applied successfully.

The following patch must be applied to ensure the correct device strings are returned by the `pciconf` utility. Copy the `pci_vendors.patch` to the `/usr/share/misc` directory. The patch file can be found in the `$ICP_ROOT/Embedded/src/patches` directory. Apply the patch to the `pci_vendors.patch` file using the following commands:

```
# cp pci_vendors.patch /usr/share/misc
# cd /usr/share/misc
# patch -p0 < pci_vendors.patch
```

Upon patch application, review the output. The output should contain text similar to:

```
Hunk #1 succeeded at 9268...
```

This indicates that the patch was applied successfully.

4.4 Enabling IPsec in the FreeBSD Kernel (Optional)

There are two parts to the FreeBSD IPsec setup: userspace IKE functionality and kernel IPsec functionality. Racoon is a tool for handling Internet Key Exchange (IKE) in IPsec for FreeBSD. Racoon installation is described in [Section 7.0](#).

Tip:

Storing your kernel configuration file directly under `/usr/src` can be a bad idea. If you are experiencing problems, it can be tempting to just delete `/usr/src` and start again, however, you will have deleted your custom kernel configuration file. Also, do not edit `GENERIC` directly, as it may get overwritten the next time you update your source tree, and your kernel modifications will be lost. You might want to keep your kernel configuration file elsewhere, and then create a symbolic link to the file in the `i386` directory. For example:

```
# cd /usr/src/sys/i386/conf
# mkdir /root/kernels
# cp GENERIC /root/kernels/MYKERNEL
# ln -s /root/kernels/MYKERNEL
```

To install the FreeBSD IPsec functionality, perform the following steps:

1. First make a backup copy, then modify the FreeBSD Kernel configuration file (located in `/usr/src/sys/i386/conf/GENERIC`) and add/verify the following options:

```
device      gif
device      crypto
device      enc
options     IPSEC
```

Upon patch application, review the output. The output should contain text similar to:

```
Hunk #1 succeeded at 392.
Hunk #2 succeeded at 516.
Hunk #3 succeeded at 599.
```




```
Hunk #4 succeeded at 631.
```

2. Apply the Opencrypto patches.

```
# patch -p1 -d /usr/src/ < $ICP_ROOT/OpenSourcePatches/  
opencrypto_FreeBSD7.1.patch
```

4.5 Rebuilding the FreeBSD Kernel

Complete instructions on building and installing the FreeBSD kernel are available at the following location:

http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/kernelconfig-building.html

Build and install the kernel by executing the following commands:

```
# cd /usr/src  
# make clean  
# make buildkernel KERNCONF=GENERIC  
# make installkernel KERNCONF=GENERIC
```

4.6 Configure Loader Configuration File

In order to memory map CDRAM and NCDRAM it is necessary to set the parameter: hw.physmem. Edit the file /boot/loader.conf and add the following line:

```
hw.physmem=672M
```

Note: The contents of this file may initially be empty.

4.7 Rebooting and Verifying

Rebooting the system will bring the system up with the newly compiled kernel.

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

```
# uname -r
```

The output should look like:

```
7.1-RELEASE
```



5.0 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.

5.1 Environment Setup

After unpacking the release package (as described in [Section 4.2](#)), 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 sources are untarred at ICP_ROOT.

```
# setenv ICP_ROOT /EP805XX_release
```

Follow the instructions in the man page for the **date** command to set the current date. For example, this command sets the date to November 5th, 2008, 15:23.

```
# date 200811051523
```

5.2 Build Options

[Section 5.3, “Build Using Top Level Make” on page 26](#) describes the instructions for building Security and Embedded components together.

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

5.3 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
# setenv ICP_BUILD_OUTPUT $ICP_ROOT/StagingArea
```

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

The following commands build the EP80579 security software and EP80579 embedded software drivers:

```
# cd $ICP_ROOT
```



```
# make clean
# make
```

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

File Name	Description
*.ko	Kernel modules or 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_freeBSD.sh	Installation scripts
qat_service_freeBSD	Shell script used to Start and Stop kernel modules

5.4 Installation of Build Output

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

```
# make install
```

Note:

Opencrypto Driver kernel module (icp_ocf.ko) is compiled but not installed with the **make install** command. For installation information, refer to [Section 6.1, “Loading Opencrypto Driver”](#) on page 30.

Besides copying the kernel modules to a sub-directory under */boot/modules*, the startup script *qat_service_freeBSD* is added in the */etc/rc.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.

5.4.1 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 setting kernel environmental variables prior to loading the *icp_crypto* kernel module. 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:

```
# kldunload icp_asd.ko
# kldunload icp_crypto.ko
# kldunload mmp_firmware.ko
# kldunload uof_firmware.ko
```

See [Section 5.4.1.4](#) for instructions on loading the modules again.



5.4.1.1 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. Use `kenv` to set this value.

To enable statistics collection:

```
# kldunload icp_crypto.ko
# kenv icp_crypto.statistics.master=all
# kldload icp_crypto.ko
```

When these commands are issued, the module will load with all statistic collection enabled. This is the default setting, so unless this parameter has been altered, it is not necessary to set `icp_crypto.statistics.master` as shown here. The real value of this parameter is observed when used in conjunction with disabling specific statistics gathering as discussed in [Section 5.4.1.2](#).

To disable all statistics collection:

```
# kldunload icp_crypto.ko
# kenv icp_crypto.statistics.master=none
# kldload icp_crypto.ko
```

When these commands are issued, the module will load with all statistics collection disabled. All individual statistics settings discussed in [Section 5.4.1.2](#) are ignored when this parameter is set to none.

5.4.1.2 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 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 kernel environmental variable value to off before loading the `icp_crypto` module. For example, the following commands will enable all statistics gathering except for the `dsa` module:

```
# kldunload icp_crypto.ko
# kenv icp_crypto.statistics.master=all
# kenv icp_crypto.statistics.dsa=off
# kldload icp_crypto.ko
```



5.4.1.3 Verbose Output

The parameter `icp_crypto.verbose` can be turned on (1) to display the selected statistics gathering options. Remember if `icp_crypto.statistics.master` is set to all, then all statistics are enabled, unless otherwise noted under the individual statistics settings. When the `icp_crypto` kernel module is loaded, the statistics logging options selected will be displayed.

```
# kldunload icp_crypto.ko
# kenv icp_crypto.statistics.master=all
# kenv icp_crypto.verbose=1
# kenv icp_crypto.statistics.dsa=off
# kldload icp_crypto.ko
Loading DCC Kernel Module ...
QATAL Driver Module Loaded.
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
```

5.4.1.4 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:

```
# kldload mmp_firmware.ko
# kldload uof_firmware.ko
# kldload icp_crypto
# kldload icp_asd
# /bin/asd_ctl
```

5.5 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/rc.d/qat_service_freeBSD stop
```



6.0 Runtime Configuration

6.1 Loading Opencrypto Driver

All EP80579 Integrated Processor kernel modules as well as Opencrypto's kernel object (ocf.ko) must be loaded into the kernel before loading the EP80579 Opencrypto Driver. If the instructions in [Section 4.4, "Enabling IPsec in the FreeBSD Kernel \(Optional\)"](#) on [page 24](#) were completed, Opencrypto kernel object was compiled into the kernel.

```
# kldload $ICP_ROOT/StagingArea/icp_ocf.ko
```

For additional information on the Opencrypto driver, refer to the README file located in the `$ICP_ROOT/Acceleration/shims/OCF_Shim/src/FREEBSDREADME.txt`.

The following system variable should be set in order to make sure userspace programs can make calls to Opencrypto:

```
# sysctl kern.userasymcrypto=1
```

'cryptodev.ko', the module which handles userspace requests to Opencrypto, should be inserted automatically when the icp_ocf module is loaded.

kldstat can be used to check whether the driver is loaded.

```
# kldstat |grep cryptodev
```

If it is not loaded automatically, it can be loaded manually.

```
# kldload cryptodev
```

6.2 OpenSSL Updates

OpenSSL 0.9.8e is released with FreeBSD 7.1. It may be found in:

```
/usr/src/crypto/openssl
```

There is a patch to get DSA functionality working correctly for the OpenSSL cryptodev engine with this release. To apply the patch:

```
# cd /usr/src/crypto/openssl
# patch -p1 < $ICP_ROOT/OpenSourcePatches/ocf-openssl-
0.9.8_verE_and_verG_PKE_fix.patch
# cd /usr/src/secure/
# make clean
# make
# make install
```

When EP80579 Opencrypto Driver and QuickAssist technology modules are inserted, running the following command shows the accelerated functionality:

```
# openssl engine -c
```



The output of this command should look similar to:

```
(cryptodev) BSD cryptodev engine
[RSA, DSA, DH, DES-CBC, DES-EDE3-CBC, AES-128-CBC]
(padlock) VIA PadLock (no-RNG, no-ACE)
(dynamic) Dynamic engine loading support
```

6.3 Using the Intel® QuickAssist Technology Cryptographic API

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

[Section 7.0, “Racoon* Installation and Configuration” on page 32](#) describes how to build and configure the Racoon IPSec network stack component on the development board.

[Section 7.3, “Configuring Sample VPN Application” on page 35](#) provides instructions for configuring a VPN application using development boards as VPN gateways.



7.0 Racoon* Installation and Configuration

Racoon is a tool for handling Internet Key Exchange (IKE) in IPsec for FreeBSD. For important information regarding IPsec performance on FreeBSD, see [Section 7.4, “IPsec Performance”](#) on page 38.

Note: It is not required to install Racoon to use the EP80579 security software Cryptographic API.

This chapter contains the following:

- instructions to continue the Racoon installation process started in [Section 4.4, “Enabling IPsec in the FreeBSD Kernel \(Optional\)”](#) on page 24
- instructions to configure Racoon on the development board
- description of how the EP80579 security software thus configured can be used to set up a sample VPN application

7.1 Racoon Installation

Note: Before you continue, you must have performed the steps in [Section 4.4, “Enabling IPsec in the FreeBSD Kernel \(Optional\)”](#) on page 24.

1. Install ipsec-tools package.
 - a. go to: http://sourceforge.net/project/showfiles.php?group_id=74601&package_id=74949
 - b. download the file “ipsec-tools-0.7.1.tar.bz2”
 - c. place it in `/usr/ports/distfiles`
2. Go to the folder `/usr/ports/security/ipsec-tools` in the FreeBSD Ports collection and build the ipsec-tools package.

```
# cd /usr/ports/security/ipsec-tools
# make clean
# make
```

Note: When the **make** command is executed, the config options may be displayed. If this occurs proceed to step 4.

3. Run ‘make config’

```
# make config
```
4. Ensure that only the items listed below are enabled.
Items not included on the list that were enabled by default must be disabled.

```
DEBUG
IPV6
ADMINPORT
STAT
DPD
FRAG
```




```

HYBRID
PAM
LDAP
RC5
IDEA

```

5. After you make your selections in the GUI, select 'OK'. The system will execute the 'make install' operation without having to type the command.

```
# make install
```

Racoon binary is placed in the folder `"/usr/local/sbin"`

7.2 Racoon Configuration

A number of configuration files are required for Racoon as it does not provide automatic keying. Default security policies are specified in the `ipsec.conf` file and referenced in the `racoon.conf` file. If `racoon.conf` is enabled appropriately, certain `spd` scripts can be called by Racoon.

A description of the scripts as well as relevant sections of sample scripts provided with EP80579 security software package are included below. The sample script files can be used to recreate the VPN application discussed in ["Configuring Sample VPN Application" on page 35](#). Sample scripts can be found in the `$ICP_ROOT/FreeBSDConfigFiles` directory.

7.2.1 racoon.conf

Configuration for Racoon wherein SA information is defined. It also allows for configuration of network interfaces and ports for ISAKMP requests. For further information, enter 'more racoon.conf' or open `racoon.conf` in a text editor to display the file contents.

Sample racoon.conf

```

...
# if no listen directive is specified, racoon will listen to all
# available interface addresses.
listen
{
    #isakmp ::1 [7000];
    isakmp 192.168.10.1 [500];
    #admin [7002];          # administrative's port by kmpstat.
    #strict_address;       # required all addresses must be bound.
}
...
# IKE exchange with FreeS/WAN PLUTO client
#
remote anonymous
{
    exchange_mode main;
    doi ipsec_doi;
    situation identity_only;
    my_identifier address;

    script "/usr/local/etc/racoon/spd.sh" phase1_up;
    script "/usr/local/etc/racoon/spd_flush.sh" phase1_down;

    nonce_size 16;
    lifetime time 10 min;    # sec,min,hour

```



```
initial_contact on;
proposal_check obey;      # obey, strict or claim

proposal {
    encryption_algorithm aes;
    hash_algorithm sha1;
    authentication_method pre_shared_key ;
    dh_group 2 ;
}

sainfo anonymous
{
    #pfs_group 1;
    lifetime time 3600 sec;
    encryption_algorithm aes;
    authentication_algorithm hmac_sha1;
    compression_algorithm deflate;
}
```

7.2.2 psk.txt

Contains Preshared key (PSK) or RSA keys. This is equivalent to the Openswan ipsec.secrets configuration file. The permissions for this file must be set to read-only or errors will be encountered that prevent Racoon from using the key information specified in the file. The psk.txt requires only the destination IP address and the key to be used to be specified.

Sample psk.txt

```
192.168.10.2 TestPSK
```

7.2.3 spd.sh

Optional script file that contains information required to initialize the security policies that will be used by Racoon. If used, racoon.conf must be configured to use this script.

In the example scripts provided, racoon.conf has been configured to call this script.

Sample spd.sh

```
setkey -c << EOF_SETKEY
flush;
spdflush;
#TRANSPORT MODE
# spdadd 192.168.10.1/24[any] 192.168.10.2/24[any] any -P out ipsec esp/
transport//require ;
# spdadd 192.168.10.2/24[any] 192.168.10.1/24[any] any -P in ipsec esp/
transport//require ;
#TUNNEL MODE
spdadd 9.0.0.0/8[any] 11.0.0.0/8[any] any -P out ipsec esp/tunnel/
192.168.10.1-192.168.10.2/require ;
spdadd 11.0.0.0/8[any] 9.0.0.0/8[any] any -P in ipsec esp/tunnel/192.168.10.2-
192.168.10.1/require ;
EOF_SETKEY
```

7.2.4 spd_flush.sh

Optional script file that contains the **setKey -PF** command which is used to flush all policies. If used, racoon.conf must be configured to use this script.



In the example scripts provided, racoon.conf has been configured to call this script.

Sample spd_flush.sh

```
#!/bin/sh

#flush;
#spdflush;
setkey -PF;
```

7.2.5 ipsec.conf

Sets the default security policies between two hosts. The location of this file is specified in the rc.conf file. Note that the structure of this file differs in structure from the Openswan version of the file with the same name. As Racoon cannot put the policies in the SPD, this file allows the policies to be set and also removes the need to manually set them using setkey upon each reboot.

Sample ipsec.conf

```
flush;
spdflush;

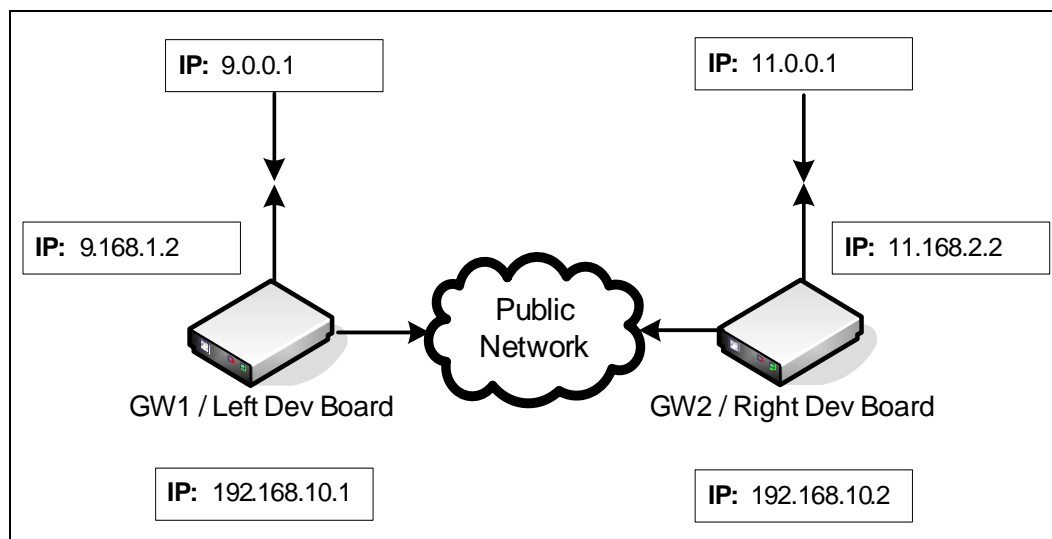
spdadd 11.0.0.0/8 9.0.0.0/8 any -P in ipsec esp/tunnel/192.168.10.1-
192.168.10.2/require;
spdadd 9.0.0.0/8 11.0.0.0/8 any -P out ipsec esp/tunnel/192.168.10.2-
192.168.10.1/require;
```

7.3 Configuring Sample VPN Application

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

The configuration is shown in Figure 5 and it assumes that the two development boards are running FreeBSD*.

Figure 5. VPN Example





7.3.1 Set Up Left Development Board

1. Follow the instructions in this document on the left development board:
 - [Section 3.0, “Installing the OS on a Development Board”](#) on page 20
 - [Section 5.0, “Building EP80579 Security Software on a Target Development Board”](#) on page 26
 - Be sure to complete [Section 4.4, “Enabling IPSec in the FreeBSD Kernel \(Optional\)”](#) on page 24.

2. Create /usr/local/etc/racoon directory if it does not already exist.

```
# mkdir /usr/local/etc/racoon
```

3. Configure Racoon. Configuration files that can be used to replicate the VPN Example in [Figure 5](#) are available in the `$ICP_ROOT/FreeBSDConfigFiles/left` directory. Copy the files to the appropriate directory listed below:

```
# setenv ICP_ROOT /EP805XX_release
# cd $ICP_ROOT/FreeBSDConfigFiles/left
# cp ipsec.conf /etc/
# cp psk.txt /usr/local/etc/racoon/
# chmod 0400 /usr/local/etc/racoon/psk.txt
# cp racoon.conf /usr/local/etc/racoon/
# cp spd.sh /usr/local/etc/racoon/
# cp spd_flush.sh /usr/local/etc/racoon/
```

Note: To enable debug messages, add the entry 'log debug' after the entry 'log notify' on line 18 in the file /usr/local/etc/racoon/racoon.conf on both the left and right systems. The entry 'log debug2' will generate additional debug messages.

4. Edit “/etc/rc.conf” file and add the following lines.

Note: IP addresses for the IPSec tunnel to be created must be specified in the rc.conf file.

```
ifconfig_iegbe0="inet 192.168.10.1 netmask 255.255.255.0"
ifconfig_iegbe1="inet 9.168.1.2 netmask 255.0.0.0"

route add -net 9.0.0.0/8 9.168.1.2
route add -net 11.0.0.0/8 192.168.10.2

gateway_enable="YES"
linux_enable="YES"
racoon_enable="YES"
ipsec_enable="YES"
ipsec_file="/etc/ipsec.conf"
ipsec_encdebug_enable="YES"
```

Note: Exercise caution when updating rc.conf. Errors in this file can lead to system booting in read-only mode. If this occurs, the following commands will mount the file system and open the rc.conf for editing.

```
mount -at ufs
/usr/bin/vi /etc/rc.conf
```

5. Reboot the system for the configuration to take effect.

7.3.2 Set Up Right Development Board

1. Follow the instructions in this document on the right development board:



- Section 3.0, “Installing the OS on a Development Board” on page 20
- Section 5.0, “Building EP80579 Security Software on a Target Development Board” on page 26
- Be sure to complete Section 4.4, “Enabling IPsec in the FreeBSD Kernel (Optional)” on page 24.

2. Create /usr/local/etc/racoon directory if it does not already exist.

```
# mkdir /usr/local/etc/racoon
```

3. Configure Racoon. Configuration files that can be used to replicate the VPN Example in Figure 5 are available in the `$ICP_ROOT/FreeBSDConfigFiles/right` directory. Copy the files to the appropriate directory listed below:

```
# setenv ICP_ROOT /EP805XX_release
# cd $ICP_ROOT/FreeBSDConfigFiles/right
# cp ipsec.conf /etc/
# cp psk.txt /usr/local/etc/racoon/
# chmod 0400 /usr/local/etc/racoon/psk.txt
# cp racoon.conf /usr/local/etc/racoon/
# cp spd.sh /usr/local/etc/racoon/
# cp spd_flush.sh /usr/local/etc/racoon/
```

Note: To enable debug messages, add the entry 'log debug' after the entry 'log notify' on line 18 in the file /usr/local/etc/racoon/racoon.conf on both the left and right systems. The entry 'log debug2' will generate additional debug messages.

4. Edit “/etc/rc.conf” file and add the following lines.

Note: IP addresses for the IPsec tunnel to be created must be specified in the rc.conf file.

```
ifconfig_iegbe0="inet 192.168.10.2 netmask 255.255.255.0"
ifconfig_iegbe1="inet 11.168.2.2 netmask 255.0.0.0"

route add -net 9.0.0.0/8 192.168.10.1
route add -net 11.0.0.0/8 11.168.2.2

gateway_enable="YES"
linux_enable="YES"
racoon_enable="YES"
ipsec_enable="YES"
ipsec_file="/etc/ipsec.conf"
ipsec_encdebug_enable="YES"
```

5. Reboot the system for the configuration to take effect.

7.3.3 Set Up VPN

1. Before attempting to bring up a tunnel, use the **netstat** command to ensure that the routing tables are correct for the network scenario depicted in Figure 5.

```
# netstat -r
```

Verify it is possible to ping between all nodes on both machines.

A sample output from the **netstat** command is as follows:

```
Routing Tables
Internet:
Destination      Gateway          Flags    Refs      Use  Netif  Expire
9.0.0.0           192.168.10.1    UGS          0         0  iegbe0
```



11.0.0.0	link#5	UC	0	0	iegbel
11.168.2.2	00:0e:0c:dd:fc:74	UHLW	1	0	lo0
localhost	localhost	UH	0	46	lo0
192.168.10.0	link#4	UC	0	0	iegbel
192.168.10.1	link#4	UHLW	2	0	iegbel

Internet6:	Destination	Gateway	Flags	Netif	Expire
	localhost	localhost	UHL	lo0	
	fe80::%lo0	fe80::1%lo0	U	lo0	
	fe80::1%lo0	link#3	UHL	lo0	
	ff01::3::	fe80::1%lo0	UC	lo0	
	ff02::%lo0	fe80::1%lo0	UC	lo0	

2. Launch Racoon on the left and right development boards using the following commands:

```
# cd /usr/local/etc/racoon
# setkey -PF
# ./spd_flush.sh
# ./spd.sh
# /usr/local/sbin/racoon -F -f /usr/local/etc/racoon/racoon.conf -l /var/log/
racoon.log
```

Note: Logging is specified to assist in the debug of any issues encountered.

Racoon will launch and wait for requests on the specified ports in the racoon.conf file. Once a ping is made from the 9.0.0.1 machine to the 11.0.0.1 machine, Racoon will negotiate and establish a VPN tunnel between both networks.

The **tcpdump** command can be used to verify that ESP packets are exchanged over the tunnel.

Note:

While Racoon is establishing the tunnel, only small data rates should attempt to be passed over the tunnel. Passing larger data rates will cause Racoon to become overwhelmed and the tunnel negotiation will fail as a consequence.

If any changes are made to the racoon configuration files, the command **killall racoon** may be used to stop Racoon. Once stopped, the changes will take effect, then Racoon can be restarted using the above command.

7.4 IPSec Performance

FreeBSD 7.1 should not be used to benchmark accelerated IPSec on the Intel® EP80579 Integrated Processor with Intel® QuickAssist Technology. Accelerated IPSec performance on FreeBSD 7.1 does not compare favorably to Linux (as measured using CentOS 5.2). It is therefore recommended that those wishing to benchmark IPSec on the EP80579 with QuickAssist should do so using CentOS 5.2 and the associated EP80579 Security Software package for Linux.

The root cause of this issue is software resource contention in the FreeBSD 7.1 kernel. This does not pose a problem for cryptographic software implementations or synchronous security offload engines. The asynchronous nature of the EP80579 Integrated Processor's cryptographic offload engine, however, results in increased contention for software resources in the IPSec stack due to the synchronous nature of its design.



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:

```
# setenv ICP_ROOT /EP805XX_release
```

9.1 Controller Area Network (CAN) Driver

9.1.1 FreeBSD Compilation Instructions

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

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

Compilation of the FreeBSD 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
# make install (this will install the driver for persistency)
```

The CAN driver compiles and the resulting `can.ko` file is placed in the `$ICP_ROOT/Embedded/src/CAN` directory. When “make install” is invoked, `can.ko` is placed in the `/boot/modules` directory.

9.1.2 FreeBSD Module Load/Unload Instructions

Note: This step is not necessary if the **make install** command above succeeded. This is only necessary to understand how to load and unload this driver individually.

To load the FreeBSD CAN driver, execute the following command from the directory where the compiled executable resides:

```
# kldload ./can.ko
```

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

```
# kldunload can.ko
```

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

```
# kldstat | grep can
```

The output of the **kldstat** command lists all modules loaded in the system. Look for an entry titled “can”.



9.1.3 FreeBSD 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 FreeBSD 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.

```
# ./install.csh
```

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:

```
# 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 FreeBSD Compilation Instructions

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

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

Compilation of the FreeBSD 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
# make install (this will install the driver for persistency)
```

The EDMA driver compiles and the resulting `dma.ko` file is placed in the `$ICP_ROOT/Embedded/src/EDMA` directory. When “make install” is invoked, `dma.ko` is placed in the `/boot/modules` directory.

9.2.2 FreeBSD Module Load/Unload Instructions

Note: This step is not necessary if the **make install** command above succeeded. This is only necessary to understand how to load and unload this driver individually.

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

```
# kldload ./dma.ko
```

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

```
# kldunload dma.ko
```



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

```
# kldstat | grep dma
```

The output of the **kldstat** command lists all modules loaded in the system. Look for an entry titled “dma”.

9.2.3 FreeBSD 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 FreeBSD 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. Log in as root. Root privileges are required for some of the operations.
2. Open a terminal window, change directory to *\$ICP_ROOT/Embedded/codelet/EDMA* and execute the following script. Do not move the script to another location.

```
# ./install.csh
```

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 and client driver output can be viewed in the terminal window.

3. 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 FreeBSD Compilation Instructions

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

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

Compilation of the FreeBSD WDT driver separately from the rest of the software package is possible. Enter the *\$ICP_ROOT/Embedded/src/WDT* directory and execute the following commands:

```
# make clean
# make
# make install (this will install the driver for persistency)
```

The WDT driver compiles and the resulting *iwdt.ko* file is placed in the *\$ICP_ROOT/Embedded/src/WDT* directory. When “make install” is invoked, *iwdt.ko* is placed in the */boot/modules* directory.

9.3.2 FreeBSD Module Load/Unload Instructions

Note:

This step is not necessary if the **make install** command above succeeded. This is only necessary to understand how to load and unload this driver individually.

To load the FreeBSD Watchdog Timer driver, execute the following command from the directory where the compiled executable resides:

```
# kldload ./iwdt.ko
```



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

```
# kldunload iwdt.ko
```

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

```
# kldstat | grep iwdt
```

The output of the **kldstat** command lists all modules loaded in the system. Look for an entry titled "iwdt".

9.3.3 FreeBSD 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 FreeBSD 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.

```
# ./install.csh
```

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 FreeBSD Compilation Instructions

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

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

Compilation of the FreeBSD GPIO driver separately from the rest of the software package is possible. Enter the *\$ICP_ROOT/Embedded/src/GPIO* directory and execute the following commands:

```
# make clean
# make
# make install (this will install the driver for persistency)
```

The GPIO driver compiles and the resulting gpio.ko file is placed in the *\$ICP_ROOT/Embedded/src/GPIO* directory. When "make install" is invoked, gpio.ko is placed in the */boot/modules* directory.

9.4.2 FreeBSD Module Load/Unload Instructions

Note: This step is not necessary if the **make install** command above succeeded. This is only necessary to understand how to load and unload this driver individually.



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

```
# kldload ./gpio.ko
```

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

```
# kldunload gpio.ko
```

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

```
# kldstat | grep gpio
```

The output of the **kldstat** command lists all modules loaded in the system. Look for an entry titled “gpio”.

9.5 IEEE 1588 Hardware Assist Driver

9.5.1 FreeBSD Compilation Instructions

All source files for the FreeBSD release of IEEE 1588 Hardware Assist driver are located in the following directory within the FreeBSD compatible EP80579 embedded software drivers release:

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

Compilation of the FreeBSD IEEE 1588 driver separately from the rest of the software package is possible. Enter the *\$ICP_ROOT/Embedded/src/1588* directory and execute the following commands:

```
# make clean
# make
# make install (this will install the driver for persistency)
```

The IEEE 1588 Hardware Assist driver compiles and the resulting timesync.ko file is placed in the *\$ICP_ROOT/Embedded/src/1588* directory. When “make install” is invoked, timesync.ko is placed in the */boot/modules* directory.

9.5.2 FreeBSD Module Load/Unload Instructions

Note:

This step is not necessary if the **make install** command above succeeded. This is only necessary to understand how to load and unload this driver individually.

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

```
# kldload ./timesync.ko
```

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

```
# kldunload timesync.ko
```

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

```
# kldstat | grep timesync
```

The output of the **kldstat** command lists all modules loaded in the system. Look for an entry titled “timesync”.



9.5.3 FreeBSD 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 FreeBSD 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.

```
# ./install.csh
```

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:

```
# 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 FreeBSD Compilation Instructions

All source files for the FreeBSD release of the EP80579 Global Configuration Unit driver and Gigabit Ethernet driver are located in the following directories within the FreeBSD compatible EP80579 embedded software drivers release:

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

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

```
# cd $ICP_ROOT/Embedded/src/GCU  
# make clean  
# make  
# make install (this will install the GCU driver for persistency)  
# cd $ICP_ROOT/Embedded/src/GbE  
# make clean  
# make  
# make install (this will install the GbE driver for persistency)
```



The Gigabit Ethernet driver compiles and the resulting `iegbe.ko` file is placed in the `$ICP_ROOT/Embedded/src/GbE` directory. The GCU driver compiles and the resulting `gcu.ko` file is placed in the `$ICP_ROOT/Embedded/src/GCU` directory. When “make install” is invoked, `iegbe.ko` and `gcu.ko` are placed in the `/boot/modules` directory.

9.6.2 FreeBSD Module Load/Unload Instructions

Note: This step is not necessary if the **make install** command above succeeded. This is only necessary to understand how to load and unload this driver individually.

To load the FreeBSD Gigabit Ethernet driver, execute the following commands from the directory where the compiled executable resides:

```
# kldload ./gcu.ko
# kldload ./iegbe.ko
```

To unload the FreeBSD Gigabit Ethernet driver, execute the following commands (`iegbe` driver must be unloaded before `gcu` driver can be unloaded):

```
# kldunload iegbe.ko
# kldunload gcu.ko
```

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

```
# kldstat | grep iegbe
# kldstat | grep gcu
```

The output of the **kldstat** command lists all modules loaded in the system. Look for an entry titled “`iegbe`” and “`gcu`”.

Configuration of the Gigabit Ethernet ports provided by the EP80579 integrated processor is through the traditional FreeBSD network command, **ifconfig**. Please consult the man pages for **ifconfig** for details.

9.7 System Management Bus (SMBus) Driver

9.7.1 FreeBSD Compilation Instructions

All source files for the SMBus FreeBSD support are available in the FreeBSD kernel's open source.

9.7.2 FreeBSD Module Load/Unload Instructions

Three drivers must be loaded to support SMBus:

- `smb.ko`
- `smbus.ko`
- `ichsmb.ko`

The `smb.ko` and `smbus.ko` drivers are installed using one command:

```
# kldload smb
```

The `ichsmb.ko` driver is installed using the following command:

```
# kldload ichsmb
```

These drivers can be unloaded using the following commands:



```
# kldunload smb.ko
# kldunload smbusr.ko
# kldunload ichsmb.ko
```

To confirm these modules have been loaded or unloaded, execute the following command:

```
# kldstat
```

The output of the **kldstat** command lists all modules loaded in the system. Look for entries entitled "smb", "smbusr", and "ichsmb".



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

10.1 Using Crypto Tools

Testing of the Framework (Opencrypto with EP80579 Opencrypto Driver and QuickAssist technology) using crypto-tools can only be done for symmetric, un-chained algorithms using the 'cryptotest' program. However, a patch is required in order to run tests correctly against the driver because the cryptotest program uses the same cryptographic session for encrypt and decrypt operations. (The EP80579 Opencrypto Driver and QuickAssist technology API require a session to be created for each operation.)

To use cryptotest with the EP80579 Opencrypto Driver, the following actions must be performed:

```
# cd /usr/src/tools/tools/crypto/  
# patch -p1 < $ICP_ROOT/OpenSourcePatches/crypto-tools-FreeBSD7.1.patch  
# make  
# ./cryptotest -va 3des 10 1000
```

The output of this command should look similar to:

```
session = 0x0  
device = icp_ocf0  
count = 10, size = 1000  
iv:  
0000: 68 35 6e 33 6a 6f 32 37 61 65 75 69 68 69 39 32  
cleartext:  
0000: 6e 31 6e 6a 6a 33 37 38 6f 38 32 75 74 74 6a 34  
0010: 62 62 62 21 69 74 6e 6f 61 38 32 39 6a 6f 6f 73  
0020: 6e 31 65 74 73 62 6f 75 69 37 39 73 74 37 35 75  
0030: 6f 73 6e 75 31 6f 68 6e 33 30 35 31 6f 30 68 61  
cleartext:  
0000: 79 52 7e a5 de a4 7f 94 fa c3 32 39 92 19 c8 66  
0010: e9 59 11 fd 36 32 54 64 09 af a1 62 05 07 f9 a4  
0020: 6d 33 60 c7 f0 8f d9 95 da 39 51 bc 9d 50 0a fb  
0030: a0 58 ed 01 94 69 ee 72 b6 6f f0 96 19 92 48 24  
0.001 sec, 20 3des crypts, 1000 bytes, 18674136 byte/sec, 142.5 Mb/sec
```

Note: PKE (Diffie-Hellman, Mod Exp, Mod Exp CRT, DSA) tests are not possible as 'cryptokeytest' was created for a different driver. If you wish to edit this code to test the Opencrypto Driver, it is recommended to use OpenSSL libraries to generate any large number (BIGNUM) parameters.



11.0 Sample Applications

Sample applications are included with this software release that present examples of how to use the 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`



12.0 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](#).

12.1 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



12.1.1 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

12.1.2 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.

12.1.3 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.

12.1.4 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.

12.1.5 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.

12.2 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.

12.2.1 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.



13.0 Uninstalling the Software

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

13.1 FreeBSD Modules/Driver Dependencies

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

Table 6. FreeBSD Module/Driver Dependencies

Module/Driver/Patch	Dependency 1	Dependency 2
EP80579 FreeBSD Patch	OS Source	None
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 (ichsmb)	smb	smbus
Watchdog Timer (WDT)	None	-
mmp_firmware.ko	/boot/firmware.ko	
uof_firmware.ko	/boot/firmware.ko	
icp_hal	mmp_firmware.ko	uof_firmware.ko
icp_debug	icp_hal	
icp_crypto	icp_debug	
icp_asd	icp_crypto	



14.0 Troubleshooting

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

Installing FreeBSD 7.1 on a hard drive which has been used before can cause an issue during boot. The installation of the operating system will seem to succeed, however, upon reboot after installation, the OS may hang at the boot loader. If using a hard drive previously used, it is recommended you first reformat the hard drive.

Upon installation of FreeBSD 7.1, a warning message will be displayed about the geometry of the disk drive. It is OK to accept the disk geometry that FreeBSD is displaying.

14.1 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.

14.2 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: `ttyd[0123] "/usr/libexec/getty std.9600" dialup off secure`
 - Change the above to: `ttyd[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.



15.0 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

