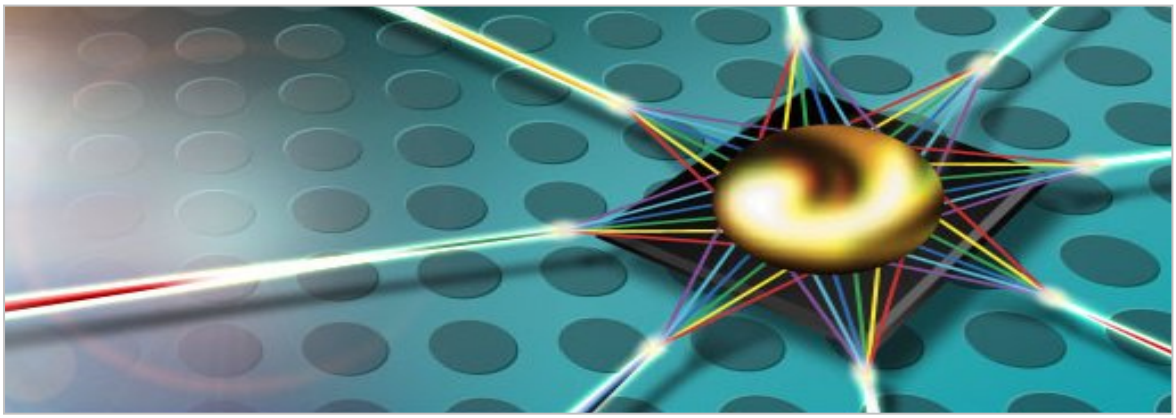
NPS-400

CPE Developer's Guide

Control Plane Environment Developer's Guide for NPS-400 Network Processors

Document Version 1.9



Document Number: 27-8221-04

The information contained is proprietary and confidential.

EZchip Semiconductor Ltd.
1 Hatamar Street, P.O. Box 527
Yokneam 20692, Israel

EZchip Semiconductor Inc.
2700 Zanker Road, Suite 150
San Jose, CA 95134, USA

Preface

EZchip Semiconductor Inc.
2700 Zanker Road, Suite 150,
San Jose, CA 95134, USA
Tel: (408) 520-3700, Fax: (408) 520-3701

EZchip Semiconductor Ltd.
1 Hatamar Street, PO Box 527,
Yokneam 20692, Israel
Tel: +972-4-959-6666, Fax: +972-4-959-4166

Email: info@ezchip.com, Web: www.ezchip.com

Please send comments regarding the documentation to supportNP@ezchip.com.

About this Manual

This manual is intended for software developers who plan to develop control plane applications for products using the EZchip NPS-400 network processor.

Related Documents

For additional information refer to:

DOCUMENT	***CONTENT***
NPS-400 EZcp Reference Manual	Describes the EZchip Control Plane Service library (EZcp) and its related APIs. The EZcp library provides an API for control-plane applications for the NPS network processor, abstracting the complexities of the underlying hardware.
NPS-400 EZdev Reference Manual	Describes the EZchip Device Access Layer library (EZdev) and its related APIs. The EZdev library defines and implements the services required for accessing NPS devices, such as detecting the devices on the PCI Express bus, mapping the devices to the CPU address space, performing memory accesses to the devices and handling interrupt event notifications from the devices.
NPS-400 EZenv Reference Manual	Describes the EZchip Environment library (EZenv) and its related APIs. The EZenv library provides a shared runtime infrastructure for all Control Plane Environment (CPE) libraries.

This Document

The following is a brief description of the contents of each section:

CHAPTER	NAME	DESCRIPTION
Section 1	**Introduction**	Provides an overview of the Control Plane Environment Libraries.
Section 2	**Folder Structure and Contents**	Folder structure and contents of the Control Plane Environment Libraries.
Section 3	**Building**	Information on building the Control Plane Environment Libraries.
Section 4	**Porting**	Provides guidelines for porting the Control Plane Environment Libraries to various platforms (OS, CPU and/or target board).
Section 5	**Control Plane Environment Libraries**	Describes each of the Control Plane Environment Library components.
Section 6	**Sample Control Plane Applications**	Describes the sample control plane applications supplied.
Appendix A: Summary of Compilation Flags		Summarizes the compilation flags used in the Control Plane Environment Libraries.

Revision History

REVISION	DATE	DESCRIPTION OF MODIFICATION
1.9	Sept. 7, 2105	Relates to EZdk version 1.9a. Porting section updated. Removed EZtbs.
1.8	Mar. 26, 2015	Relates to EZdk version 1.8a. Removed EZvpci and VxWorks.
1.7	Nov. 5, 2014	Relates to EZdk version 1.7a. Appendix A: Summary of Compilation Flags: EZ_CPU_TYPE removed.
1.6	July 17, 2014	Relates to EZdk version 1.6a. No changes to document.
1.5	Mar. 10, 2014	Initial release. Relates to EZdk version 1.5a.

Terminology and Conventions

General

The following terminology is used throughout this document:

TERM	DESCRIPTION
NPS / NP	Refers to the EZchip NPS-400 network processor device and/or software simulator.
Channel	Refers to an NPS-400 device and/or simulator in the system
Control Plane Application	Refers to a customer-developed application responsible for configuration and management of the NPS device.
Control Plane CPU	Refers to the CPU on which the control plane application resides. This may be an external host CPU or the NPS CTOPs.

Typographical Conventions

The following typographical conventions are used in this manual. Routine (or function/call) names are written with a parenthesis, e.g. EZapi_Create( ).

📖 Refer to the section or document referenced here for additional information on the topic.

- *Notes provide additional information that is not necessarily mandatory.*

Important: Contains information that is mandatory for proper confirmation and/or operation that should not be overlooked.

Contents

Preface 2
About this Manual 2
Related Documents 2
This Document 3
Revision History 3
Terminology and Conventions 4
Figures and Tables 5
1. Introduction 6
2. Folder Structure and Contents 7
3. Building 8
4. Porting 9
4.1 Data Types 9
4.2 CPU Endianness 9
4.3 CPU Alignment 10
4.4 CPU Address (32 bit vs. 64-bit) 10
4.5 OS Abstraction 10
4.6 NPS Device Access 11
5. Control Plane Environment Libraries 12
5.1 EZcp 12
5.2 EZdev 13
5.3 EZenv 13
5.4 EZagt 14
5.5 EZspy 14
6. Sample Control Plane Applications 15
6.1 EZware 15
7. Appendix A: Summary of Compilation Flags 16

Figures and Tables

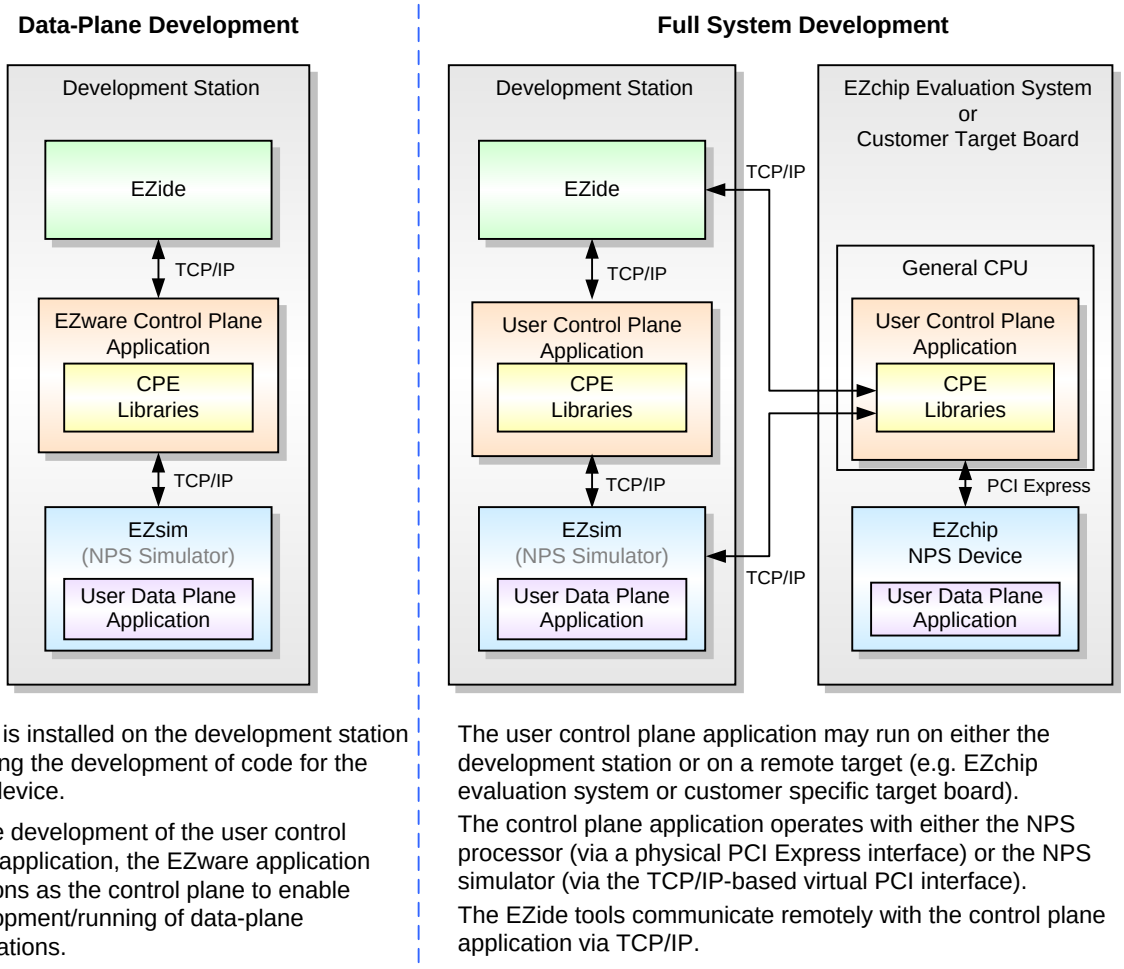
Figure 1. Typical development environment for an NPS-based product 6
Figure 2. The Device Access Layer (EZdev) 11
Figure 3. Main software components in NPS control plane applications 12

1. Introduction

The EZdk Control Plane Environment (CPE) provides a set of libraries used to development control plane applications for systems based on the NPS network processors.

The following diagram illustrates a typical development environment for an NPS-based product:

Figure 1. Typical development environment for an NPS-based product



EZide is installed on the development station enabling the development of code for the NPS device.

Before development of the user control plane application, the EZware application functions as the control plane to enable development/running of data-plane applications.

The user control plane application may run on either the development station or on a remote target (e.g. EZchip evaluation system or customer specific target board).

The control plane application operates with either the NPS processor (via a physical PCI Express interface) or the NPS simulator (via the TCP/IP-based virtual PCI interface).

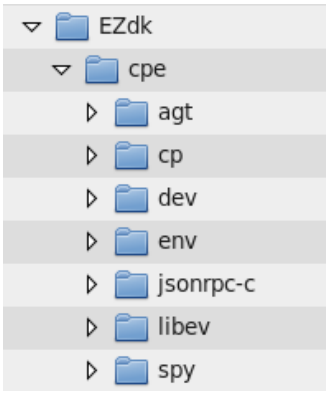
The EZide tools communicate remotely with the control plane application via TCP/IP.

The system is designed for a seamless transition from the development of data-plane and control-plane applications using the NPS software simulator (in the initial design/pre-silicon stages) to using the 'real' NPS device on a target board. The NPS device replaces the software simulator and a physical PCI Express interface replaces the virtual PCI with no need to modify the control plane application and/or CPE libraries' source code. Note that the NPS simulator is typically, but not necessarily, run on the development station.

This document provides an overview on the EZdk Control Plane Environment (CPE) libraries, as well as information on how to port these to various target platforms.

2. Folder Structure and Contents

The figure and table below show the directory structure for the cpe directory after installing the EZdk software development kit.



FOLDER	CONTENTS
/EZdk	
/cpe	Control-plane environment libraries
/agt	EZchip Agent library (EZagt) The EZagt library implements a remote agent for the EZide tools running on the PC.
/cp	EZchip Control Plane Service library (EZcp). The EZcp library provides an API for control-plane applications for the NPS network processor, abstracting the complexities of the underlying hardware.
/dev	EZchip Device Access Layer library (EZdev). The EZdev library defines and implements the services required for accessing NPS devices, such as detecting the devices on the PCI Express bus, mapping the devices to the CPU address space, performing memory accesses to the devices and handling interrupt event notifications from the devices.
/env	EZchip Environment library (EZenv). The EZenv library provides a shared runtime infrastructure for all Control Plane Environment (CPE) libraries.
/jsonrpc-c	Jason-RPC server (used by the EZagt library)
/libev	Event loop library (used by the jsonrpc-c library)
/spy	EZchip Spy Library (EZspy) EZspy is a troubleshooting tool for NPS device validation and status.

3. Building

This section provides information on how to compile and link the Control Plane Environment (CPE) libraries for the Linux x86-64 bit platform.

Each of the CPE libraries and sample control plane application includes a standalone GNU makefile which can be used to build the library. The makefile is located under the component's directory. The resulting library file is created under each component's /lib directory.

Building a library is done using the command:

```
make –f <makefile> EZDK_BASE=<EZdk installation directory> EZDK_PLATFORM=<EZdk
platform name> -B
```

For example, to completely re-build the EZcp library for Linux x86 64-bit platforms:

```
make –f ~/EZchip/EZdk_1.5a/cpe/cp/Makefile EZDK_BASE=~/EZchip/EZdk_1.9a
EZDK_PLATFORM=linux_x86_64 -B
```

- *Developers porting to the CPE libraries to additional platforms may use the supplied GNU makefiles as references for their platform-specific build environment.*

4. Porting

This section provides guidelines for porting the EZdk Control Plane Environment (CPE) libraries to various platforms (OS, CPU and/or target board).

4.1 Data Types

The EZenv library defines abstractions for all basic data types (EZtype.h). All CPE libraries use the types defined in EZtype.h, allowing to change the mapping of EZdk Control Plane Environment library types to the matching CPU specific types in a single location, without modifying the CPE libraries' source code.

The following basic data types are defined in EZtype.h:

- EZui32 – Unsigned 32 bit variable (prefix ui)
- EZi32 – Signed 32 bit variable (prefix i)
- EZui64 – Unsigned 64 bit variable (prefix ui)
- EZi64 – Signed 64 bit variable (prefix i)
- EZus16 – Unsigned 16 bit variable (prefix us)
- EZs16 – Signed 16 bit variable (prefix s)
- EZuc8 – Unsigned 8 bit variable (prefix uc)
- EZc8 – Signed 8 bit variable (prefix c)
- EZbool – Boolean variable (prefix b)
- EZfloat – Single-precision floating-point variable (prefix f)
- EZdouble – Double-precision floating-point variable (prefix d)
- EZptr – A pointer to something (void*). EZptr is capable of holding any pointer (i.e. pointer to any type). Generally, the size of EZptr is the size of address of the machine. For 64 bit machines, this is an 8 byte variable.
- EZvar – A type that is capable of holding a numeric value of up to the pointer size (e.g. 32 bits for 32-bit platforms or 64 bits for 64-bit platforms).

4.2 CPU Endianness

The CPE libraries support both little endian and big endian CPUs.

The endianness of the CPU in use is defined using preprocessor definitions (EZdef.h):

- EZ_ENDIAN_LITTLE – Compile for little endian CPU (default).
- EZ_ENDIAN_BIG – Compile for big endian CPU.

In addition, preprocessor definitions are used to define if accesses to the NPS device for registers and memory read/write should perform a swap (used in EZdev):

- EZdev_VAL_SWAP and EZdev_VAL_NO_SWAP – Define to EZdev whether to swap value when written/read.
- EZdev_BUF_SWAP and EZdev_BUF_NO_SWAP – Define to EZdev whether to swap buffer when written/read.

4.3 CPU Alignment

The CPE libraries support CPUs which require aligned access to memory.

The alignment requirements of the CPU in use are defined using preprocessor definitions (EZdef.h):

- EZ_CPU_NOT_ALIGNED – Compile for CPU with no requirements on alignment of access to memory (default).
- EZ_CPU_ALIGNED – Compile for CPU that requires aligned accesses to memory.

4.4 CPU Address (32 bit vs. 64-bit)

The CPE libraries support 32-bit and 64-bit CPUs.

The address/pointer size of the CPU in use is defined using preprocessor definitions (EZdef.h):

- EZ_CPU_ADDRESS_32_BIT – Compile for 32-bit CPU.
- EZ_CPU_ADDRESS_64_BIT – Compile for 64-bit CPU.

4.5 OS Abstraction

The EZenv library provides infrastructures to abstract OS dependencies, allowing to easily port the CPE libraries to various operating systems and runtime environments.

The EZenv library contains an OS abstraction layer which implements all OS dependencies, as well as all other external dependencies (standard library functions, runtime library functions, etc.). All CPE libraries use the abstraction layer, allowing to change the implementation to match any OS in a single location, without modifying the CPE libraries' source code.

The EZos layer defines a common interface for all operating systems. In addition, the EZos layer provides sample implementations for all supported OSs. In most cases, the OS specific implementation is located in an OS specific c file, allowing to provide a new implementation or modify an existing implementation without needing to recompile the CPE libraries. However, in some performance critical areas, the OS specific implementation is performed in the header files.

Developers wishing to port the CPE libraries to additional OSs can either modify the existing implementations, or add additional parallel implementations.

The OS in use is defined using preprocessor definitions (EZdef.h):

- EZ_OS_WIN – Windows variants.
- EZ_OS_LINUX – Linux/Unix variants, user space.
- EZ_OS_LINUX_KERNEL – Linux/Unix variants, kernel space.

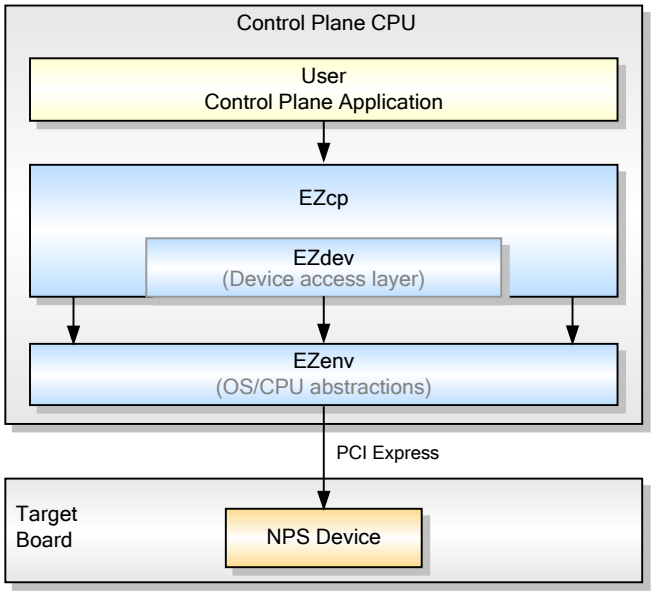
The implementation of the Linux kernel portions of the CPE libraries use the Linux kernel OS services directly and do not use the EZos abstraction layer. This is done to remove unnecessary overhead and simplify the reference implementation for Linux kernel developers. The EZos module thus does not supply a Linux kernel implementation.

4.6 NPS Device Access

The EZchip device access layer (EZdev) further extends the EZcp library's portability by defining the services required for accessing NPS devices, such as detecting the devices on the PCI Express bus, mapping the devices to the CPU address space, performing memory accesses to the devices and handling interrupt event notifications from the devices.

The EZcp library is connected to the device access layer using a set of device access callbacks. While the EZcp library is responsible for issuing NPS device access requests, the device access layer provides the platform specific services to execute these requests.

Figure 2. The Device Access Layer (EZdev)



In many cases, the implementations provided in the EZdk software development kit can be used as is. Alternatively, the provided implementation may be used as a reference to assist customers in their development of implementations optimized for specific target platforms.

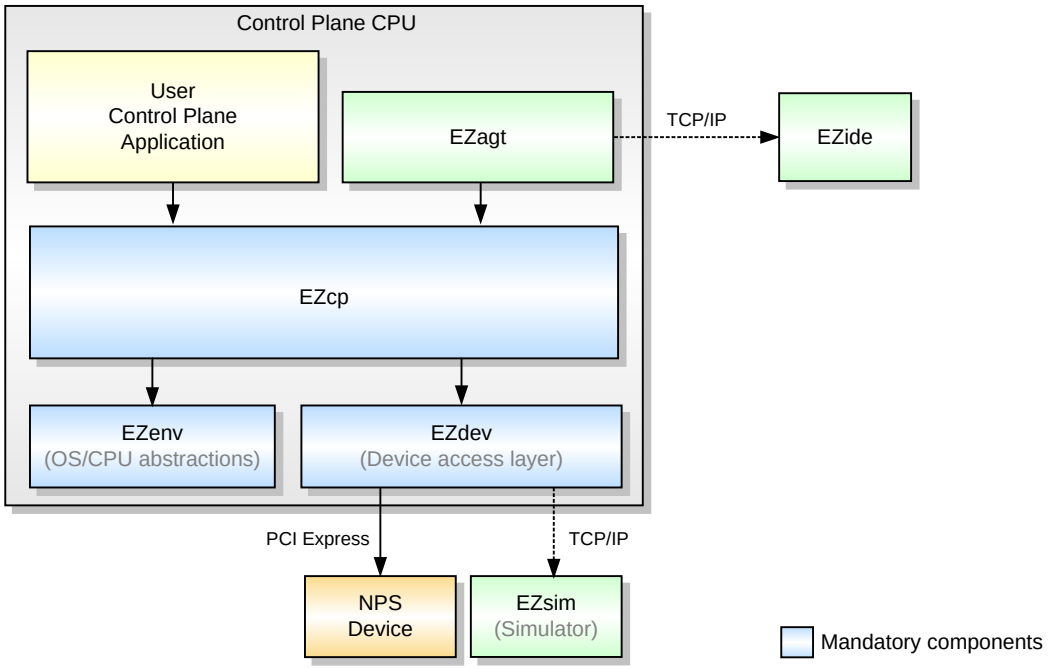
📖 The EZdev library and its API routines are described in the *EZdev Reference Manual*.

5. Control Plane Environment Libraries

The EZdk software development kit provides a set of Control Plane Environment (CPE) libraries for systems utilizing the NPS network processor.

This section provides a short description of each of the CPE libraries, their roles and the interactions between them.

Figure 3. Main software components in NPS control plane applications



5.1 EZcp

The EZcp library provides an API for operation with NPS devices. The control-plane application, as well as other EZcp components, all access the NPS devices through the supplied API.

The EZcp API support includes configuration of the NPS device's CTOPs; configuration of its network interfaces; configuration of its embedded traffic managers; creating and maintaining its lookup structures.

Dependencies: The EZcp library is the main component of the CPE, and is mandatory in any NPS control plane application.

Folder Structure and Content:

Subfolder	Contents
/cp	
/include	EZcp definition files
/lib	EZcp library binaries
/src	EZcp implementation files

📖 The EZcp library and its API routines are described in the *EZcp Reference Manual*.

5.2 EZdev

The EZdev (device access) layer defines services connecting the EZcp library to the NPS devices such as detecting the devices on the PCI Express bus, mapping the devices to the CPU memory space, accessing the NPS device registers and handling interrupt event notifications from the device.

The EZdev library provided in the EZdk software development kit provides reference implementations for several platforms. In many cases, the implementations provided can be used as is. Alternatively, the provided implementation may be used as a reference to assist customers in their development of implementations optimized for specific target platforms.

Dependencies: An EZdev layer implementation is mandatory for all NPS control plane applications.

Folder Structure and Content:

Subfolder	Contents
/dev	
/include	EZdev definition files
/lib	EZdev library binaries
/src	EZdev implementation files

📖 The EZdev layer interface and reference implementations are described in the *EZdev Reference Manual*.

5.3 EZenv

The EZenv (environment) library implements a shared infrastructure and runtime environment for all EZdk control plane environment (CPE) libraries. The EZenv library centralizes all CPU and OS abstractions, easing the porting process to various target operating systems, runtime environment and CPU architectures. In addition, the EZenv library implements utility functions, modules and infrastructures which are utilized by the various EZcp libraries.

Dependencies: The EZenv library is mandatory in any NPS control plane application.

Folder Structure and Content:

Subfolder	Contents
/env	
/include	EZenv definition files
/lib	EZenv library binaries
/src	EZenv implementation files

📖 The EZenv library and its API routines are described in the *EZenv Reference Manual*.

5.4 EZagt

The EZagt library implements a remote agent for the EZide tools running on the PC. The EZide tools communicate with the EZagt via a TCP/IP connection. The EZagt library receives the messages, and performs the requested services in the control plane application on behalf of the EZide tools. The EZagt library is mandatory for operation with the EZide environment.

Dependencies: The EZagt library is required for working with the EZide environment. It is recommended (but not required) to leave the EZagt library in the final control plane application to allow connecting to the system using the EZide environment.

Folder Structure and Content:

Subfolder	Contents
/agt	
/include	EZagt definition files
/bin	EZagt library binaries
/src	EZagt implementation files

- *The EZagt library uses the jsonrpc-c and the libev open source libraries to implement the TCP connection and Jason-rpc server.*

5.5 EZspy

EZspy is an additional library that allows the user to use the Spy commands for NPS validation and status.

Dependencies: It is recommended (but not required) to leave the EZspy library in the final control plane application to allow use of the EZspy debug capabilities.

Folder Structure and Content:

Subfolder	Contents
/spy	
/include	EZspy definition files
/bin	EZspy library binaries
/src	EZspy implementation files

6. Sample Control Plane Applications

The EZdk software development kit includes sample control-plane applications which can be used as a reference for developing customer-specific control plane applications.

6.1 EZware

The EZdk software development kit includes the source code for the EZware application, used as the initial control plane application when working with the in the EZide development environment (before developing a customer-specific control-plane application). The EZware application may run on the either the development station or a target system (e.g. the EZchip evaluation system), and connects between the EZide tools and the EZsim software simulator or real NPS device.

7. Appendix A: Summary of Compilation Flags

This section summarizes the compilation flags used in the control plane environment (CPE) libraries.

Values in **BOLD** indicate the default behavior when no compilation flags are defined**.**

General (all components)

Operating System (EZdef.h):	EZ_OS_[**WIN**|LINUX|LINUX_KERNEL]
CPU Endian (EZdef.h)	EZ_ENDIAN_[**LITTLE**|BIG]
PCI Swap (EZdef.h)	EZ_PCI_[**NO_SWAP**|SWAP]
CPU alignment (EZdef.h)	EZ_CPU_[**NOT_ALIGNED**|ALIGNED]
CPU address size(EZdef.h)	EZ_CPU_ADDRESS_[**32_BIT**|64_BIT]
Development Level (EZdevL.h)	EZ_DEVL_[USER|**NOTE**|MAINTENANCE|DEBUG]_LEVEL