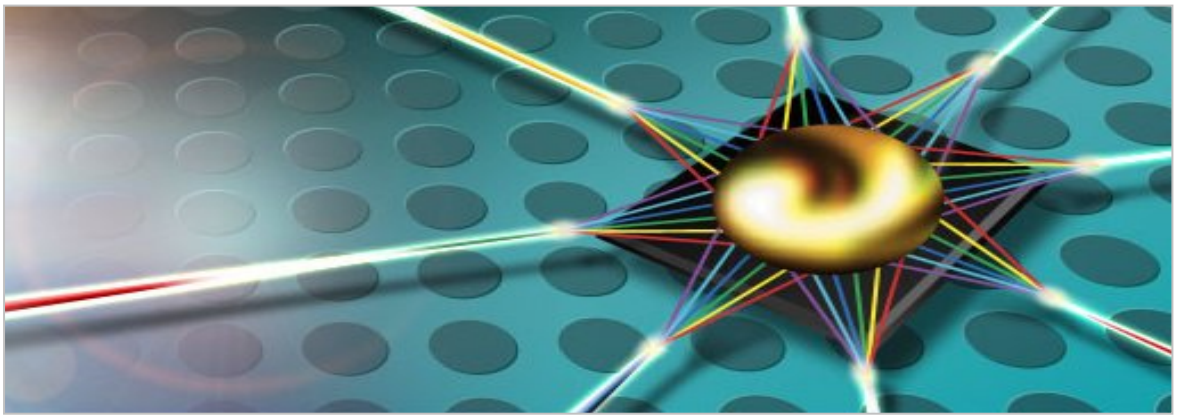
NPS-400

EZenv Reference Manual

Environment Library for NPS-400 Network Processors

Document Version 1.9



Document Number: 27-8215-04

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

EZchip welcomes your comments on this publication. Please address them to: supportNP@ezchip.com.

About this Manual

This document 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 manual is intended for software developers who plan to develop boards based on the EZchip NPS-400 Network Processor.

This Document

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

CHAPTER	*NAME*	*DESCRIPTION*
Section 1	**Overview**	Introduction to the EZenv library, its architecture and implementation.
Section 2	**Reference**	Describes the routines and calls relating to the EZenv functionality.
Appendix A: Preprocessor Definitions		Lists the preprocessor definition may be used to control EZenv related functionality.

Revision History

REVISION	*DATE*	*DESCRIPTION OF MODIFICATION*
1.9	Sept. 7, 2015	Relates to EZdk version 1.9a. New routines for time measurement enabling you to print the current time and to calculate the difference between times. See Time Measurement Routines (EZosTime.h). Remote device infrastructure removed.
1.8	March 4, 2015	Relates to EZdk version 1.8a.
1.7	Nov. 9, 2014	Relates to EZdk version 1.7a. New routines: EZlog_OpenAdditionalLogFile( ), EZlog_CloseAdditionalLogFile( ), EZlog_SetSubComponentLogFile( ), EZosIO_GetNativeDevicePtrs( ). Updated routines: EZlog_SetLog( ), EZlog_IsLogEnabled( ), EZosIO_ChangeDevicePtrs( ) return.
1.6	July 17, 2014	Relates to EZdk version 1.6a. Update to EZlog_SetLog in EZlog.h file.
1.5	Mar. 10, 2014	Initial release. Relates to EZdk version 1.5a.

Terminology and Conventions

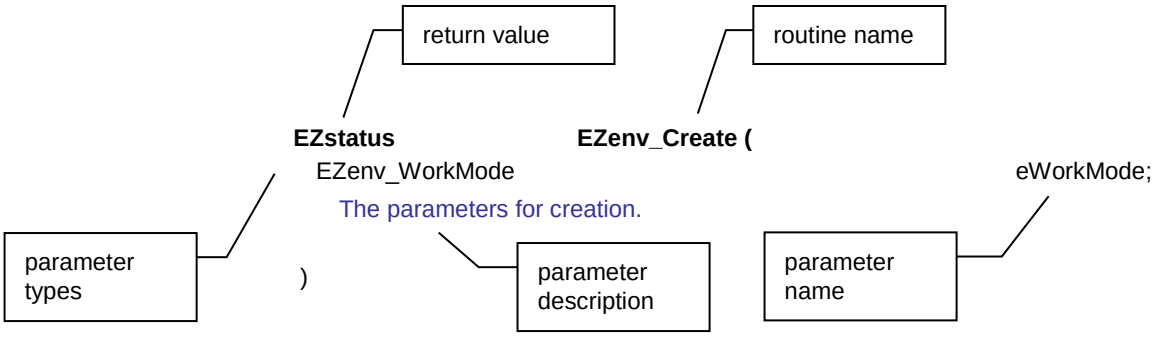
General

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.

Conventions Used for Routines

Routines, or functions/calls, are designated by the keyword EZstatus followed by the routine name enclosed in parentheses. Parameters are listed in parentheses.



Typographical Conventions

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

List of Figures and Tables

1. Overview

The EZchip Environment library (EZenv) provides a shared runtime infrastructure for all Control Plane Environment (CPE) libraries.

The EZenv library provides several types of services:

- **CPU Abstraction** – Infrastructures to abstract CPU dependencies, allowing to easily port the CPE libraries to various CPU architectures. See section 1.1.
- **OS Abstraction** – Infrastructures to abstract OS dependencies, allowing to easily port the CPE libraries to various operating systems and runtime environments. See section 1.2.
- **Utilities and Infrastructures**– Utility functions, modules and infrastructures which are utilized by the CPE libraries. See section 1.3.

The following sections provide more details on each of these topics.

1.1 CPU Abstraction

The EZenv library provides infrastructures to abstract CPU dependencies, allowing to easily port the NPS Control Plane Environment (CPE) libraries to various CPU architectures.

1.1.1 Data Types

The EZenv library defines abstractions for all basic datatypes (EZtype.h). All CPE libraries use the types defined in *EZtype.h*, allowing to change the mapping of CPElibrary types to the matching CPU specific types in a single location, without modifying the CPE libraries.

The following basic datatypes are defined in EZtype.h:

- **EZui32** – Unsigned 32 bit variable (prefix ui)
- **EZi32** – Signed 32 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)
- **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).

- *The CPE libraries do not use 64 bit numeric datatypes on 32-bit platforms.*
- *The CPE libraries do not use floating point operations. Floating point operations are done using floating point emulation services (see EZfloat32.h in the EZenv include directory).*

1.1.2 CPU Endianess

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 memory via the PCI perform a swap (either by the operating system or by the underlying hardware):

- EZ_PCI_NO_SWAP – The operating system or hardware passed PCI data to the NPS as is (default).
- EZ_PCI_SWAP – The operating system or hardware performs a swap on each 4 bytes of PCI data passed to the NPS.

1.1.3 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.

1.1.4 CPU Address (32 bit vs. 64 bit)

The CPE libraries support both 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 (default).
- EZ_CPU_ADDRESS_64_BIT – Compile for 64-bit CPU.

1.1.5 CPU Type (External vs. Embedded)

The CPE libaries support both an external CPU operating via the PCI Express interface and the embedded CPUs within the NPS device.

The typeof the CPU (external vs. embedded) is defined using preprocessor definitions (EZdef.h):

- EZ_CPU_TYPE_EXTERNAL – Compile for external CPU (default).
- EZ_CPU_TYPE_EMBEDDED – Compile for embedded CPU.

1.2 OS Abstraction

The EZenv library provides infrastructures to abstract OS dependencies, allowing to easily port the NPS Control Plane Environment (CPE) librariesto 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.

The following OS services are defines:

- **EZosMem** – Memory services (malloc, free, memset, memcpy, memmove, memcmp, etc.).
- **EZosIO** – IO services (fopen, fclose, fprintf, fflush, etc.).
- **EZosTask** – Task management services (task spawn, task id, task delay, mutexes, semaphores, etc.).
- **EZosSocket** – Socket services (create, listen, connect, accept, send, receive, etc.).
- **EZosMsgQ** – Message queue services (create send, receive, etc.).
- **EZosStdarg** – Standard argument services (valist, start, end, etc.).
- **EZosMisc** – Additional/miscellaneous services (geterrno, clock, etc.).

The EZos layer defines a common interface for all operating systems. This includes the OS routines, as well as the return codes and OS data types. In addition, the EZos layer provides sample implementations for the supported OSs. In most cases, the OS specific implementation is done is 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 done in the header files, requiring recompilation of the CPE libraries when changed.

Developers wishing to port the OS abstraction implementation 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 (default).
- EZ_OS_LINUX – Linux/Unix variants, user space.
- EZ_OS_LINUX_KERNEL – Linux/Unix variants, kernel space.
- EZ_OS_VXWORKS – VxWorks OS.

See OS Routines (EZos.h) section.

The VxWorks OS implementation is not supplied as part of the EZdk installation. Customers using the VxWorks OS should implement the EZos functionality under the directory EZdk/cpe/env/src/os/vxworks.

1.3 Utilities and Infrastructures

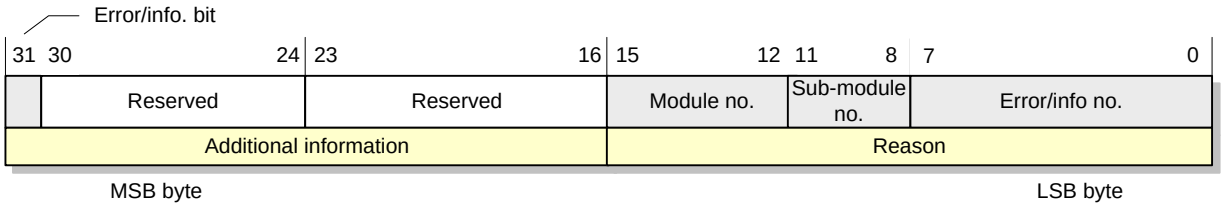
The EZenv library provides several utility functions, modules and infrastructures which are utilized by the Control Plane Environment (CPE) libraries.

1.3.1 Return Codes

The EZenv library defines a common return code infrastructure for all CPE libraries, as well as the division of the the return code address space between the CPE libraries (EZrc.h).

When a CPE library API call returns EZstatus, it will return a 4-byte code according to the following convention:

Figure 1. Return code format



Bit 31	Error indication – If set, the returned value is an error.
Bits 30:16	Reserved (zero).
Bits 15:12	Module number – Indicates the module that returned the error (EZcp, EZdev, etc.).
Bits 11:8	Sub-module number – Indicates the sub-module that returned the error (TM, stats, etc.).
Bits 7:0	Error/info number – Indicates the specific error (or info.) number.

EZenv also defines the following macros which should be used when working with return codes (EZrc.h):

- EZrc_IS_ERROR(retVal) – Indicates if the return code represents an error.
- EZrc_IS_INFO(retVal) – Indicates if the return code represents an information code.

Following is the classification of return codes

- EZok (a value of 0) indicates a successful call, i.e. the action requested by the call was performed fully and successfully.
- A value greater than 0x80000000 (i.e. error bit set) indicates an error, i.e. the action requested by the call has failed. In this case, the reason for the failure is reported by the code.
- A non zero value less than 0x80000000 (i.e. error bit cleared) indicates an information code. i.e. the action succeeded, the requested operation was performed, but additional information was returned.

📖 The return codes for each module are defined in a separate include file located in the module's include directory (EZrc.h, EZdevRC.h, etc.).

Return Code Sample Usage

```
#include EZapiRC.h
...
   EZstatus  uiRetVal;

   /* API execution */
   uiRetVal = EZapiStruct_AddEntry( ... );

   if ( EZrc_IS_ERROR( uiRetVal ) )
   {
      /* The code is error – analyze cause and try to fix */
      switch ( uiRetVal )
      {
      case EZrc_CP_SRH_WRONG_KEY_SIZE:
         /* Change to correct key size */
         break;

      default:
         printf( “Unhandled error %u”, uiRetVal );
         break;
      }
   }
   else if ( EZrc_IS_INFO ( uiRetVal ) )
   {
      /* The code is info – analyze cause and handle */
      switch ( uiRetVal )
      {
      case EZrc_CP_SRH_ENTRY_ALREADY_EXISTS_INF:
         /* Entry already existed and actual action was modify */
         break;

      default:
         break;
      }
   }
   else
   {
      /* The code is EZok – do nothing */
   }
```

1.3.2 Version Information

The EZenv library defines a common version information infrastructure for all CPE libraries (EZversion.h).

The version of each library is comprised of:

- Major version number – Used to represent the EZdk version for an NPS device family (e.g. NPS-400)
- Minor version number – Used to represents EZdk versions for the same NPS device family.
- Version Letter – Used to represent intermediate releases within a minor release.
- Build – used to represent internal/patch builds.

Each of the CPE libraries contains a global version information structure which holds the current version information for that component. In addition, each CPE library also provides a routine to obtain the version information. The version information infrastructure then provides a utility fuctions to translate the version information structure to generate a common displayable string. Control plane applications can use these infrastructures to query and report the various component versions.

Version Information Sample Usage

```
#include "EZversion.h"
...
   EZuc8          aucVerInfo[ EZversionInfo_MAX_STRING_LENGTH ];
   EZversionInfo  *psVerInfo;

   /* CP version. */
   psVerInfo = EZapiCP_GetVersionInfo();

   EZversionInfo_GetVersionString( psVerInfo, aucVerInfo );

   EZlog_Print( EZlog_COMP_USER,
                EZlog_SUB_COMP_LOG_ALL,
                EZlog_LEVEL_ERROR,
                "%s",
                aucVerInfo );

   /* DEV version. */
   psVerInfo = EZdev_GetVersionInfo();

   EZversionInfo_GetVersionString( psVerInfo, aucVerInfo );

   EZlog_Print( EZlog_COMP_USER,
                EZlog_SUB_COMP_LOG_ALL,
                EZlog_LEVEL_ERROR,
                "%s",
                aucVerInfo );
```

1.3.3 Logging

The EZenv library defines a logging infrastructure used by the Control Plane Environment (CPE) libraries (EZlog.h) for logging/tracing.

The logging infrastructure defines 7 levels of logging message. Levels are hierarchical, thus, for example, selecting warning level will also print all error and fatal error messages.

- Fatal – Fatal (unrecoverable) error conditions.
- Error – Error condition.
- Warning – Event which may indicate an error (but may also be OK under some scenarios).
- Trace – Inputs/outputs between components (API), flow through code (function call stack) and major events in the system (main state changes such as "channel is created", "memory partition is loaded", etc.).
- Info – Detailed information such as calculation results, detailed internal state, flow of code/decision points within functions, etc.).
- Debug – Highly detailed information (addresses allocated/freed, etc.).

The logging infrastructure also defines four output streams:

- Standard Error
- Standard Output
- File
- Memory

Finally, the logging infrastructure defines components representing each CPE library (EZcp, EZdev, etc.) and sub-components. Sub-components are logical divisions specific to each module/library (for example, the EZcp library component includes a search sub-componen for all search structure functionality, a statistics sub-component for all statistics functionality, etc.).

The logging infrastructure allows users to configure the level of logging output to each of the output streams (separately). This configuration can be performed globally (for all components and subcomponents) per component and/or per sub-component.

Thus, for example, users can configure standard error to show error level messages from the CP librarycomponent (including all sub-components), and at the same time configure the standard output to show information level messages for the EZcp and EZdev components.

The logging infrastructure also provides control on how messages are displayed. This includes control to display a timestamp with each message, display the task ID which invoked each message as well as control to display the component and/or subcomponent which invoked the message. In addition, the logging infrastructure provides control of the error messages displayed (error messages may be displayed with or without information of the source code file and line where the error was detected). Finally, the logging infrastructure also provides control if to force immediate flushing of outputs.

Logging Infrastructure Sample Usage

```
#include "EZlog.h"
...
   EZstatus    retVal = EZok;

   /* Set the log file name (and optional path). */
   retVal = EZlog_SetFileName( pcFileName );
   if ( EZrc_IS_ERROR( retVal ) )
   {
      return retVal;
   }
```

```
/* Open the log file. */
retVal = EZlog_OpenLogFile();
if ( EZrc_IS_ERROR( retVal ) )
{
   return retVal;
}

/* Force flushing of outputs to the file. */
EZlog_SetForceFlush( TRUE );

/* Configure to output to the file error level messages from all
 * components and sub-components. */
retVal = EZlog_SetLog( EZlog_OUTPUT_FILE,
                       EZlog_COMP_ALL,
                       EZlog_SUB_COMP_LOG_ALL,
                       EZlog_LEVEL_ERROR );
if ( EZrc_IS_ERROR( retVal ) )
{
   return retVal;
}

/* Configure to not output anything to the standard error (from all
 * components and sub-components). */
retVal = EZlog_SetLog( EZlog_OUTPUT_STDERR,
                       EZlog_COMP_ALL,
                       EZlog_SUB_COMP_LOG_ALL,
                       EZlog_LEVEL_NONE );
if ( EZrc_IS_ERROR( retVal ) )
{
   return retVal;
}

/* Configure to output to the standard output error level messages from all
 * components and sub-components. */
retVal = EZlog_SetLog( EZlog_OUTPUT_STDOUT,
                       EZlog_COMP_ALL,
                       EZlog_SUB_COMP_LOG_ALL,
                       EZlog_LEVEL_ERROR );
if ( EZrc_IS_ERROR( retVal ) )
{
   return retVal;
}

...

/* Print an info level message under the user component,
 * user1 sub-component. */
EZlog_Print( EZlog_COMP_USER,
             EZlog_SUB_COMP_USR_1,
             EZlog_LEVEL_INFO,
             "Hello World!\n" );

...

/* Close the log file. */
retVal = EZlog_CloseLogFile();
if ( EZrc_IS_ERROR( retVal ) )
{
   return retVal;
}
```

📖 See Logging Routines (EZlog.h) section.

1.3.4 Development Level

The EZenv library defines a common development level infrastructure used by the Control Plane Environment (CPE) libraries (EZdevL.h).

The development level infrastructure defines several levels of compilation, allowing to use a specified level of debug/tracing code during the development stages, and remove it when moving to production stage, removing non-critical code to reduce runtime and code space.

Four hierarchical levels of compilation are defined:

1. User level – Production, no tracing/logging.
   Contains the regular code to be used in the final production version.
2. Note level – Production, with tracing/logging.
   Contains code required only for logging/tracing capability.
   This code may have a slight (but not substantial) impact on performance and code space. Most production environments should compile at this level to allow tracing/debugging capabilities in the field. Compiling user level can remove the tracing code, at the expense of losing debugging/tracing capabilities in the field.
3. Maintenance level – Development , sanity checks.
   Contains code which checks correctness of the code – such as sanity checks, extra validity checks, etc. – which serves the development team to maintain integrity/testing code. This level will have an impact on performance and code space, and thus should ordinarily not be used on production systems.
4. Debug level – Development, debugging code.
   Contains code for low-level/detailed debugging. This is similar to sanity level, but usually contains code has a high impact on performance, and thus will impact standard development/running if included.

The CPE library projects are supplied with debug builds compiled under maintenance development level (include production, tracing and sanity code), while release builds compile under note development level (includes production and tracing code).

The development level to use is defined using preprocessor definitions (EZdevL.h):

- EZdevL_USER_LEVEL – User level
- EZdevL_NOTE_LEVEL – Note level
- EZdevL_MAINTENANCE_LEVEL – Maintenance level
- EZdevL_DEBUG_LEVEL – Debug level

1.3.5 Messaging

The EZenv library defines a messaging infrastructure used by the Control Plane Environment (CPE) libraries (EZmsg.h).

The messaging infrastructure allows the CPE library tasks to send messages between one another. The infrastructure defines a common message format/header for all messages, and allows to send tasks using logical IDs (irrespective of the actual OS dependent task IDs).

The messaging infrastructure supports messages not only between the tasks running on the target CPU, but also messages to/from remote tasks using TCP/IP sockets. This is used, for example, to communicate between the tasks running on the target CPU and the development tools (EZide) running on a remote PC machine.

The messaging infrastructure implementation uses two tasks for remote task communication:

- **Remote Task Server Thread (RTS)**
  The EZmsg remote task server thread continually listens for incoming socket connections from remote tasks – normally running in the PC development tools (EZide). The server task is responsible for setting up the connections. When the server thread receives new connection, it registers the connections in the EZmsg remote task data structures. In addition, the server task is responsible for receiving messages from the remote tasks on the socket connections and forwarding them to the local task message queues based on the message destination.
- **Remote Task Client Thread (RTC)**
  The EZmsg remote task client thread is responsible for transferring back response messages to the remote tasks – normally running in the PC development tools (EZide).

1.3.6 Utility Functions

The EZenv library includes several basic utility function modules:

- **EZutl** – Basic utility functions (min, max, swap, etc.).
- **EZutlBit** – Bit manipulation functions (set bit, clear bit, write bit, read bit, etc.).
- **EZutlBit32** – Bit manipulation functions on 32 bit numeric variables (user to encode/decode register data).
- **EZutlBitBuf** – Bit manipulation functions on data buffers of arbitrary length (used to encode/decode memory data).

1.3.7 Time Functions

The EZenv library includes time measurement functions. These functions enable you to print the current time and to calculate the difference between times.

1.4 Folder Structure and Contents

The following table details the folder structure and contents of the EZenv library:

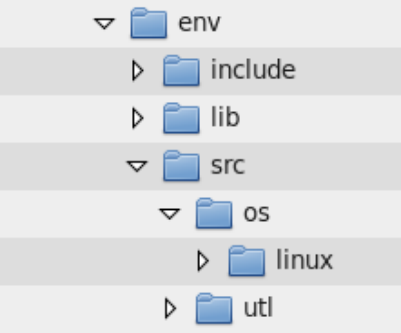


Table 1. Folder structure and contents

SUB-FOLDER	*CONTENTS*
../EZdk	
../cpe	
../env	
../include	Definition files
../lib	Library files.
../src	Libraries for the operating systems and utilities.

2. Reference

2.1 API Routines

This section describes each of the routines used in the EZenv library.

2.1.1 Summary of API Routines

Table 2. Summary of EZenv API routines

2.2 General Routines (EZenv.h)

2.2.1 EZenv_Create( )

Description

Create the EZenv library.

Synopsis

EZstatus EZenv_Create (void)

Precondition

Returns

Notes

See also

2.2.2 EZenv_Delete( )

Description

Delete the EZenv library.

Synopsis

EZstatus EZenv_Delete (void)

Precondition

Returns

Notes

See also

2.2.3 EZenv_GetVersionInfo( )

Description

Get version information.

Synopsis

EZversionInfo *EZenv_GetVersionInfo (void)

Precondition

None, may be called at any time.

Returns

Returns pointer to EZenv version info structure.

Notes

See also

2.3 Logging Routines (EZlog.h)

This module is used by all Control Plane Environment (CPE) libraries to provide logging/execution tracing services.

2.3.1 EZlog_SetLog( )

Description

Set current logging.

Synopsis

```
EZstatus        EZlog_SetLog (
  EZui32                                        uiOutputMask,
    Mask of output streams to configure:
    EZlog_OUTPUT_STDOUT
    EZlog_OUTPUT_STDERR
    EZlog_OUTPUT_FILE
    EZlog_OUTPUT_MEMORY
  EZui32                                        uiCompMask,
    Component to configure:
    EZlog_COMP_TBS
    EZlog_COMP_AGT
    EZlog_COMP_VPCI
    EZlog_COMP_ENV
    EZlog_COMP_DEV
    EZlog_COMP_NL
    EZlog_COMP_USER
    EZlog_COMP_SPY

    EZlog_COMP_CP_CP
    EZlog_COMP_CP_CHANNEL
    EZlog_COMP_CP_FCU
    EZlog_COMP_CP_ICU
    EZlog_COMP_CP_IF
    EZlog_COMP_CP_PRM
    EZlog_COMP_CP_STAT
    EZlog_COMP_CP_STRUCT
    EZlog_COMP_CP_TCAM
    EZlog_COMP_CP_TM
    EZlog_COMP_CP_GEN
  EZui32                                        uiSubCompMask,
    Subcomponents to configure:
    EZlog_SUB_COMP_CP_ALL
    EZlog_SUB_COMP_CP_ALL_API
    EZlog_SUB_COMP_CP_ALL_COR
    EZlog_SUB_COMP_CP_ALL_PRM
    EZlog_SUB_COMP_CP_ALL_GEN

    /* CP_CP sub-component defines */
    EZlog_SUB_COMP_CP_CP_API
    EZlog_SUB_COMP_CP_CP_COR
    EZlog_SUB_COMP_CP_CP_OS

    /* CP_CHANNEL sub-component defines */
    EZlog_SUB_COMP_CP_CHANNEL_API
    EZlog_SUB_COMP_CP_CHANNEL_COR
    EZlog_SUB_COMP_CP_CHANNEL_COR_DP
    EZlog_SUB_COMP_CP_CHANNEL_PRM
    EZlog_SUB_COMP_CP_CHANNEL_PRM_PMU
    EZlog_SUB_COMP_CP_CHANNEL_PRM_PUP
    EZlog_SUB_COMP_CP_CHANNEL_PRM_SER
    EZlog_SUB_COMP_CP_CHANNEL_PRM_CLUSTER
```

```
EZlog_SUB_COMP_CP_CHANNEL_PRM_EMEM
EZlog_SUB_COMP_CP_CHANNEL_PRM_IMEM
EZlog_SUB_COMP_CP_CHANNEL_PRM_BMU
EZlog_SUB_COMP_CP_CHANNEL_PRT
EZlog_SUB_COMP_CP_CHANNEL_PRM_GIM

/* CP_FCU sub-component defines */
EZlog_SUB_COMP_CP_FCU_API
EZlog_SUB_COMP_CP_FCU_COR
EZlog_SUB_COMP_CP_FCU_PRM

/* CP_ICU sub-component defines */
EZlog_SUB_COMP_CP_ICU_API
EZlog_SUB_COMP_CP_ICU_COR
EZlog_SUB_COMP_CP_ICU_PRM

/* CP_IF sub-component defines */
EZlog_SUB_COMP_CP_IF_API
EZlog_SUB_COMP_CP_IF_COR
EZlog_SUB_COMP_CP_IF_PRM
EZlog_SUB_COMP_CP_IF_PRM_IFTDM
EZlog_SUB_COMP_CP_IF_PRM_TMDMA
EZlog_SUB_COMP_CP_IF_PRM_TND
EZlog_SUB_COMP_CP_IF_PRM_SERDES

/* CP_PRM sub-component defines */
EZlog_SUB_COMP_CP_PRM_API
EZlog_SUB_COMP_CP_PRM_COR
EZlog_SUB_COMP_CP_PRM_PRM_PERF_MODULE
/* CP_STAT sub-component defines */
EZlog_SUB_COMP_CP_STAT_API
EZlog_SUB_COMP_CP_STAT_COR
EZlog_SUB_COMP_CP_STAT_PRM

/* CP_STRUCT sub-component defines */
EZlog_SUB_COMP_CP_STRUCT_API
EZlog_SUB_COMP_CP_STRUCT_COR
EZlog_SUB_COMP_CP_STRUCT_COR_ALG_TCAM
EZlog_SUB_COMP_CP_STRUCT_COR_ALG_TCAM_DEV
EZlog_SUB_COMP_CP_STRUCT_COR_MMNG
EZlog_SUB_COMP_CP_STRUCT_COR_PRT
EZlog_SUB_COMP_CP_STRUCT_COR_HASH
EZlog_SUB_COMP_CP_STRUCT_COR_TABLE
EZlog_SUB_COMP_CP_STRUCT_COR_ULTRA_IP
EZlog_SUB_COMP_CP_STRUCT_PRM_SRCH_LOG

/* CP_TCAM sub-component defines */
EZlog_SUB_COMP_CP_TCAM_API
EZlog_SUB_COMP_CP_TCAM_COR
EZlog_SUB_COMP_CP_TCAM_PRM

/* CP_TM sub-component defines */
EZlog_SUB_COMP_CP_TM_API
EZlog_SUB_COMP_CP_TM_COR
EZlog_SUB_COMP_CP_TM_PRM
EZlog_SUB_COMP_CP_TM_PRM_SHAPING
EZlog_SUB_COMP_CP_TM_PRM_WFQ
EZlog_SUB_COMP_CP_TM_PRM_WRED

/* CP_GEN sub-component defines */
EZlog_SUB_COMP_CP_GEN_COR
EZlog_SUB_COMP_CP_GEN_MTX_SW
EZlog_SUB_COMP_CP_GEN_MTX_HW
EZlog_SUB_COMP_CP_GEN_PCI
EZlog_SUB_COMP_CP_GEN_TRN
EZlog_SUB_COMP_CP_GEN_MEM
EZlog_SUB_COMP_CP_GEN_MEM_DB
EZlog_SUB_COMP_CP_GEN_REG
```

```
EZlog_SUB_COMP_CP_GEN_SIM_IF
EZlog_SUB_COMP_CP_GEN_LOG
EZlog_SUB_COMP_CP_GEN_DEV
EZlog_SUB_COMP_CP_GEN_HMM
EZlog_SUB_COMP_CP_GEN_UTL

/* TBS sub-component defines */
EZlog_SUB_COMP_TBS_TBD

/* VPCI sub-component defines */
EZlog_SUB_COMP_VPCI_API
EZlog_SUB_COMP_VPCI_CORE
EZlog_SUB_COMP_VPCI_SOCKET

/* ENV sub-component defines */
EZlog_SUB_COMP_ENV_MSG
EZlog_SUB_COMP_ENV_OS
EZlog_SUB_COMP_ENV_RDEV
EZlog_SUB_COMP_ENV_DELAY

/* AGT sub-component defines */
EZlog_SUB_COMP_AGT_API
EZlog_SUB_COMP_AGT_CORE
EZlog_SUB_COMP_AGT_AGT

/* USER sub-component defines */
EZlog_SUB_COMP_USER_1
EZlog_SUB_COMP_USER_2
EZlog_SUB_COMP_USER_3
EZlog_SUB_COMP_USER_4
EZlog_SUB_COMP_USER_5

/* DEV sub-component defines */
EZlog_SUB_COMP_DEV_COMMON
EZlog_SUB_COMP_DEV_CHANNEL
EZlog_SUB_COMP_DEV_ISR

/* NL sub-component defines */
EZlog_SUB_COMP_NL_API

/* SPY sub-component defines */
EZlog_SUB_COMP_SPY_FUNC
EZlog_SUB_COMP_SPY_CPTR
EZlog_SUB_COMP_SPY_FPTR
EZlog_SUB_COMP_SPY_PTRQDB
EZlog_SUB_COMP_SPY_FCU
EZlog_SUB_COMP_SPY_OQ
EZlog_SUB_COMP_SPY_POL
EZlog_SUB_COMP_SPY_QC
EZlog_SUB_COMP_SPY_SC
EZlog_SUB_COMP_SPY_TMDMA
EZlog_SUB_COMP_SPY_IF
EZlog_SUB_COMP_SPY_MIN
EZlog_SUB_COMP_SPY_IOCFD
EZlog_SUB_COMP_SPY_RFD
EZlog_SUB_COMP_SPY_TOP
EZlog_SUB_COMP_SPY_SPTR
EZlog_SUB_COMP_SPY_INDEXQ
EZlog_SUB_COMP_SPY_MSGQ
EZlog_SUB_COMP_SPY_DEBUG
EZlog_SUB_COMP_SPY_RND
EZlog_SUB_COMP_SPY_TND
EZlog_SUB_COMP_SPY_PMU
EZlog_SUB_COMP_SPY_BMU
EZlog_SUB_COMP_SPY_STS
EZlog_SUB_COMP_SPY_CB
EZlog_SUB_COMP_SPY_MEM
EZlog_SUB_COMP_SPY_CLUSTER
```

EZlog_SUB_COMP_SPY_COMMON

/* General sub-component defines */
EZlog_SUB_COMP_LOG_ALL

EZui32 uiLevel

Level of logging:

EZlog_LEVEL_NONE - None level.

EZlog_LEVEL_FATAL - Fatal error.

EZlog_LEVEL_ERROR - Recoverable error.

EZlog_LEVEL_WARNING - Not error but suspicious operation.

EZlog_LEVEL_TRACE - Interface between components, flow of code, progress & key points in time.

EZlog_LEVEL_INFO - Logging info (detailed, but high-level - reasonable quantity).

EZlog_LEVEL_DEBUG - Highly detailed info (high quantity, low-level - API, pointer allocation/free).

)

2.3.2 EZlog_GetLog( )

Description

Get current logging.

Synopsis

```
EZstatus    EZlog_GetLog (
  EZui32                                    uiOutputMask,
    One of output streams to get configuration.
  EZui32                                    uiCompMask,
    Component to get configuration
  EZui32                                    uiSubCompMask,
    Subcomponent to get configuration.
  EZui32*                                   puiLevel
    Level of logging.
)
```

Precondition

Returns

Notes

See also

2.3.3 EZlog_IsLogEnabled( )

Description

Check if current component log level for output streams is greater than or equal to that given.

Synopsis

```
EZextern    EZlog_IsLogEnabled (
  EZui32                                uiComponent,
    Component
  EZui32                                uiLevel
    Logging level for check.
)
```

Precondition

Returns

Notes

See also

2.3.4 EZlog_SetFilePtr( )

Description

Set pointer for printing to file.

Synopsis

EZstatus EZlog_SetFilePtr (

EZfile fFile

Pointer to file.
Cannot be stdout or stderr.
If fFile is EZosIO_INVALID_FILE no output to file is permitted.

)

Precondition

Returns

Notes

See also

2.3.5 EZlog_GetFilePtr( )

Description

Get pointer for printing to file.

Synopsis

EZstatus EZlog_GetFilePtr (

EZfile* pfFile

Pointer to file.

)

Precondition

Returns

Notes

See also

2.3.6 EZlog_SetFileName( )

Description

Set file name for printing to file.

Synopsis

EZstatus EZlog_SetFileName (
EZc8* pcFileName
File name for log.
)

Precondition

Returns

Notes

See also

2.3.7 EZlog_GetFileName( )

Description

Get file name for printing to file.

Synopsis

```
EZstatus        EZlog_GetFileName (
  EZc8*                                   pcFileName,
    File name for log.
  EZui32                                  uiMaxSize
    Size of pcFileName in bytes (maximum file name length).
)
```

Precondition

Returns

Notes

See also

2.3.8 EZlog_OpenLogFile( )

Description

Open file with previously set file name.

Synopsis

EZstatus EZlog_OpenLogFile (void)

Precondition

Returns

Notes

See also

2.3.9 EZlog_CloseLogFile( )

Description

Close file with previously set file name.

Synopsis

EZstatus EZlog_CloseLogFile (void)

Precondition

Returns

Notes

See also

2.3.10 EZlog_OpenAdditionalLogFile( )

Description

Open an additional file for a specific index.

Synopsis

```
EZextern        EZlog_OpenAdditionalLogFile (
  EZc8                                                    *pcFileName
    File name.
  EZui32                                                  uiIndex
    Index of file. Must be bigger than 0 and less than
    EZlog_NUMBERS_OF_ADDITIONAL_LOG_FILES
)
```

Precondition

Returns

Notes

See also

2.3.11 EZlog_CloseAdditionalLogFile( )

Description

Close previously opened additional file for a specific index.

Synopsis

EZextern EZlog_CloseAdditionalLogFile (

EZui32 uiIndex

Index of file. Must be bigger than 0 and less than EZlog_NUMBERS_OF_ADDITIONAL_LOG_FILES

)

Precondition

Returns

Notes

See also

2.3.12 EZlog_SetSubComponentLogFile( )

Description

Set file by index to specific subcomponent for component. Both component and subcomponent can be masked.

Synopsis

```
EZextern    EZlog_SetSubComponentLogFile (
  EZui32                                          uiIndex
    Index of file. Must be less than
    EZlog_NUMBERS_OF_ADDITIONAL_LOG_FILES
  EZui32                                          uiComponentMask
    Component mask
  EZui32                                          uiSubComponentMask
    Sub-component mask
)
```

Precondition

Returns

Notes

See also

2.3.13 EZlog_OpenLogMemory( )

Description

Open memory log stream.

Synopsis

EZstatus EZlog_OpenLogMemory (
EZui32 uiSize
Size of log in kilobytes.
)

Precondition

Returns

Notes

See also

2.3.14 EZlog_CloseLogMemory( )

Description

Close log memory stream.

Synopsis

EZstatus EZlog_CloseLogMemory (void)

Precondition

Returns

Notes

See also

2.3.15 EZlog_FlushLogMemory( )

Description

Flush log memory stream to streams.

Synopsis

EZstatus EZlog_FlushLogMemory (
EZui32 uiOutputMask
Output mask of streams.
)

Precondition

Returns

Notes

See also

2.3.16 EZlog_SetPrintTaskId( )

Description

Enable printing task ID.

Synopsis

EZstatus EZlog_SetPrintTaskId (
EZbool bEnable
Enable or disable.
)

Precondition

Returns

Notes

See also

2.3.17 EZlog_SetPrintErrorSource( )

Description

Enable printing source file and line.

Synopsis

EZstatus EZlog_SetPrintErrorSource (
EZbool bEnable
Enable or disable.
)

Precondition

Returns

Notes

See also

2.3.18 EZlog_SetPrintCompName( )

Description

Enable printing component name.

Synopsis

EZstatus EZlog_SetPrintCompName (
EZbool bEnable
Enable or disable.
)

Precondition

Returns

Notes

See also

2.3.19 EZlog_SetPrintSubCompName( )

Description

Enable printing sub-component name.

Synopsis

EZstatus EZlog_SetPrintSubCompName (
EZbool bEnable
Enable or disable.
)

Precondition

Returns

Notes

See also

2.3.20 EZlog_SetPrintTime( )

Description

Enable printing time.

Synopsis

EZstatus EZlog_SetPrintTime (

EZbool bEnable

Enable or disable.

)

Precondition

Returns

Notes

See also

2.3.21 EZlog_SetForceFlush( )

Description

Enable forcing of flush for file stream.

Synopsis

EZstatus EZlog_SetForceFlush (
EZbool bEnable
Enable auto flush.
)

Precondition

Returns

Notes

See also

2.3.22 EZlog_SetMaximalLogSize( )

Description

Set maximum size for log file.

Synopsis

EZstatus EZlog_SetMaximalLogSize (
EZui32 uiSizeInMBytes
Size of the log in Mbytes or EZlog_FILE_SIZE_INFINITE.
)

Precondition

Returns

Notes

See also

2.3.23 EZlog_EnableTaskFiltering( )

Description

Enable task filtering.

Synopsis

EZstatus EZlog_EnableTaskFiltering (

EZbool bEnable

Enable or disable log for specific task.

EZtask tTask

Task ID for enable/disable.
If *tTask* is EZosTask_INVALID_TASK and enable is TRUE, the log is enabled for all tasks.
If *tTask* is EZosTask_INVALID_TASK and enable is FALSE, the log is disabled for all tasks.

)

Precondition

Returns

Notes

See also

2.3.24 EZlog_EnableTaskForLog( )

Description

Enable log filter per task.

Synopsis

EZstatus EZlog_EnableTaskForLog (
EZbool bEnable
Enable or disable filtering log for tasks.
)

Precondition

Returns

Notes

See also

2.3.25 EZlog_SetECPULog( )

Description

Set ECPU logging.

Synopsis

EZstatus EZlog_SetECPULog (

EZui32 uiConfigMask

Mask of ECPU logging configuration:
EZlog_ECPU_LOG_MODE_NONE
EZlog_ECPU_LOG_MODE_ENABLE
EZlog_ECPU_LOG_MODE_FLUSH
EZlog_ECPU_LOG_ONLY_MIRROR

)

Precondition

Returns

Notes

Bit 0 - Enable logging.

See also

2.3.26 EZlog_Print( )

Description

Print log line.

Synopsis

```
EZui32          EZlog_Print (
  EZui32                                    uiComp,
    Component
  EZui32                                    uiSubComp,
    Subcomponent.
  EZui32                                    uiLevel,
    Level.
  const EZc8*                               pcFmt,
    Line to format.
                                            …
    List of arguments.
)
```

Precondition

Returns

Notes

See also

2.3.27 EZlog_VPrint( )

Description

Print log line.

Synopsis

```
EZui32        EZlog_VPrint (
  EZui32                                  uiComp,
    Component
  EZui32                                  uiSubComp,
    Subcomponent.
  EZui32                                  uiLevel,
    Level.
  const EZc8*                             pcFmt,
    Line to format.
  EZ_VaList                               vaPtr
    Pointer to list of arguments.
)
```

Precondition

Returns

Notes

See also

2.3.28 EZlog_PrintError( )

Description

Print error message (including error number and source line where the error was detected).

Synopsis

```
EZui32          EZlog_PrintError (
  EZui32                                    uiComp,
    Component
  EZui32                                    uiSubComp,
    Subcomponent.
  EZui32                                    uiError,
    Error number.
  const EZc8*                               pcFmt
    Line to format.
                                            ...
    List of arguments.
)
```

Precondition

Returns

Notes

See also

2.3.29 EZlog_PrintFatal( )

Description

Print fatal error message (including error number and source line where the error was detected).

Synopsis

```
EZui32          EZlog_PrintFatal (
  EZui32                                    uiComp,
    Component
  EZui32                                    uiSubComp,
    Subcomponent.
  EZui32                                    uiError,
    Error number.
  const EZc8*                               pcFmt
    Line to format.
                                            ...
    List of arguments.
)
```

Precondition

Returns

Notes

See also

2.3.30 EZlog_PrintSource( )

Description

rint message (including source line).

Synopsis

```
EZui32          EZlog_PrintSource (
  EZui32                                    uiComp,
    Component
  EZui32                                    uiSubComp,
    Subcomponent.
  EZui32                                    uiLevel,
    Level.
  const EZc8*                               pcFmt,
    Line to format.
                                            …
    List of arguments.
)
```

Precondition

Returns

Notes

See also

2.3.31 EZlog_PrintData8( )

Description

Print stream of 8 bit data.

Synopsis

```
EZui32          EZlog_PrintData8 (
  EZui32                                    uiComp,
    Component
  EZui32                                    uiSubComp,
    Subcomponent.
  EZui32                                    uiLevel,
    Level.
  const EZc8*                               pucData,
    Stream of data to print.
  EZui32                                    uiSize,
    Size of data (in 8 bit segments).
  EZui32                                    uiAlign
    Size to align (in 8 bit segments).
)
```

Precondition

Returns

Notes

See also

2.3.32 EZlog_PrintData16( )

Description

Print stream of 16 bit data.

Synopsis

```
EZui32          EZlog_PrintData16 (
  EZui32                                    uiComp,
    Component
  EZui32                                    uiSubComp,
    Subcomponent.
  EZui32                                    uiLevel,
    Level.
  const EZc16*                              pusData,
    Stream of data to print.
  EZui32                                    uiSize,
    Size of data (in 16 bit segments).
  EZui32                                    uiAlign
    Size to align (in 16 bit segments).
)
```

Precondition

Returns

Notes

See also

2.3.33 EZlog_PrintData32( )

Description

Print stream of 32 bit data.

Synopsis

```
EZui32          EZlog_PrintData32 (
  EZui32                                    uiComp,
    Component
  EZui32                                    uiSubComp,
    Subcomponent.
  EZui32                                    uiLevel,
    Level.
  const EZc32*                              puiData,
    Stream of data to print.
  EZui32                                    uiSize,
    Size of data (in 32 bit segments).
  EZui32                                    uiAlign
    Size to align (in 32 bit segments).
)
```

Precondition

Returns

Notes

See also

2.3.34 EZlog_PrintNoPrefix( )

Description

Print log line without prefixes.

Synopsis

```
EZui32          EZlog_PrintNoPrefix (
  EZui32                                    uiComp,
    Component
  EZui32                                    uiSubComp,
    Subcomponent.
  EZui32                                    uiLevel,
    Level.
  const EZc8*                               pcFmt,
    Line to format.
                                            …
    List of arguments.
)
```

Precondition

Returns

Notes

See also

2.3.35 EZlog_VPrintNoPrefix( )

Description

Print log line without prefixes.

Synopsis

```
EZui32        EZlog_VPrintNoPrefix (
  EZui32                                  uiComp,
    Component
  EZui32                                  uiSubComp,
    Subcomponent.
  EZui32                                  uiLevel,
    Level.
  const EZc8*                             pcFmt,
    Line to format.
  EZ_VaList                               vaPtr
    Pointer to list of arguments.
)
```

Precondition

Returns

Notes

See also

2.3.36 EZlog_PrintHeader( )

Description

Print log line header style.

Synopsis

```
EZui32          EZlog_PrintHeader (
  EZui32                                    uiComp,
    Component
  EZui32                                    uiSubComp,
    Subcomponent.
  EZui32                                    uiLevel,
    Level.
  const EZc8*                               pcFmt,
    Line to format.
                                            …
    List of arguments.
)
```

Precondition

Returns

Notes

See also

2.3.37 EZlog_VPrintHeader( )

Description

Print log line header style.

Synopsis

```
EZui32        EZlog_VPrintHeader (
  EZui32                                  uiComp,
    Component
  EZui32                                  uiSubComp,
    Subcomponent.
  EZui32                                  uiLevel,
    Level.
  const EZc8*                             pcFmt,
    Line to format.
  EZ_VaList                               vaPtr
    Pointer to list of arguments.
)
```

Precondition

Returns

Notes

See also

2.3.38 EZlog_PrintPrefix( )

Description

Print prefix only.

Synopsis

```
EZui32        EZlog_PrintPrefix (
  EZui32                                  uiComp,
    Component
  EZui32                                  uiSubComp,
    Subcomponent.
  EZui32                                  uiLevel,
    Level.
)
```

Precondition

Returns

Notes

See also

2.3.39 EZlog_SetPrefix( )

Description

Set logging prefix.

Synopsis

EZui32 EZlog_SetPrefix (

EZui32 uiPrefix

Value of prefix.

)

Precondition

Returns

Notes

See also

2.3.40 EZlog_ShiftPrefix( )

Description

Shift logging prefix.

Synopsis

EZui32 EZlog_ShiftPrefix (

EZi32 iShiftPrefix

Value of the shift.

)

Precondition

Returns

Notes

If value of prefix is negative it set to 0.

See also

2.4 Messaging Routines (EZmsg.h)

Header file for utilities message manager.

2.4.1 EZmsg_Start( )

Description

Start the message queue module.
This function starts working with remote tasks through TCP.

Synopsis

EZstatus EZmsg_Start (void)

Precondition

Returns

Notes

See also

2.4.2 EZmsg_Stop( )

Description

Stop the message queue module.
This function stops working with remote tasks.

Synopsis

EZstatus EZmsg_Stop (void)

Precondition

Returns

Notes

See also

2.4.3 EZmsg_Create( )

Description

Create a message queue.

Synopsis

```
EZstatus    EZmsg_Create (
  EZui32                                    uiMsgQId,
    ID of message queue.
  EZui32                                    uiMaxMsgs
    Maximum number of messages. (Cannot be greater than 64).
)
```

Precondition

Returns

Notes

See also

2.4.4 EZmsg_Delete( )

Description

Delete a message queue.

Synopsis

EZstatus EZmsg_Delete (

EZui32 uiMsgQId,

ID of message queue.

)

Precondition

Returns

Notes

See also

2.4.5 EZmsg_Send( )

Description

Send message to task.

Synopsis

```
EZstatus        EZmsg_Send (
  EZmsg_Header*                                   psHeader,
    Header of the message.
  EZptr                                           pData,
    Buffer with message.
  EZui32                                          uiTimeout
    Time-out to send.
)
```

Precondition

Returns

Notes

See also

2.4.6 EZmsg_Receive( )

Description

Receive message.

Synopsis

```
EZstatus    EZmsg_Receive (
  EZui32                          uiMsgQId,
    ID of message queue.
  EZmsg_Header*                   psHeader,
    Header of the message.
  EZptr*                          ppData,
    Buffer with message.
  EZui32                          uiTimeout
    Time-out to receive.
)
```

Precondition

Returns

Notes

See also

2.4.7 EZmsg_ChangeConnectionPort( )

Description

This function sets a TCP port for working with remote tasks.

Synopsis

EZstatus EZmsg_ChangeConnectionPort (
EZui32 uiPort
New port.
)

Precondition

Must be called before EZmsg_Start( ).

Returns

Notes

See also

2.4.8 EZmsg_Header

```
struct EZmsg_Header {
  EZui32                                    uiDstId;
    Destination message queue ID.
  EZui32                                    uiSize;
    Size of the data + sizeof ( EZmsg_Header ).
  EZui32                                    uiSrcId;
    Source message queue ID.
  EZstatus                                  uiStatus;
    Status of the message.
  EZui32                                    uiCmd;
    Command ID.
  EZui32                                    uiUserData;
    General user data - can be used for 4 byte messages.
}
```

2.5 OS Routines (EZos.h)

Header file for OS independent library.

2.5.1 EZos_Create( )

Description

Creates an OS library.

Synopsis

EZstatus EZos_Create (void)
)

Precondition

Returns

Notes

See also

2.5.2 EZos_Delete( )

Description

Deletes an OS library.

Synopsis

EZstatus EZos_Delete (void)
)

Precondition

Returns

Notes

See also

2.6 OS File Manager Routines (EZosIO.h)

Header file for OS independent file manager.

2.6.1 EZosIO_CreateModule( )

Description

Creates an IO module.

Synopsis

EZstatus EZosIO_CreateModule (void)
)

Precondition

Returns

Notes

See also

2.6.2 EZosIO_DeleteModule( )

Description

Deletes an IO module.

Synopsis

EZstatus EZosIO_DeleteModule((void))

Precondition

Returns

Notes

See also

2.6.3 EZosIO_ChangeDevicePtrs( )

Description

Change default pointers for devices.

Synopsis

```
EZui32          EZosIO_ChangeDevicePtrs (
  EZosIO_dopen_FuncPtr                          pfDopen,
    New pointer for device open.
  EZosIO_dclose_FuncPtr                         pfDclose,
    New pointer for device close.
  EZosIO_dread_FuncPtr                          pfDread,
    New pointer for device read.
  EZosIO_dwrite_FuncPtr                         pfDwrite,
    New pointer for device offset write.
  EZosIO_dfread_FuncPtr                         pfDfread,
    New pointer for device stream read.
  EZosIO_dfwrite_FuncPtr                        pfDfwrite,
    New pointer for device stream write.
  EZosIO_dioctl_FuncPtr                         pfDioctl
    New pointer for device ioctl.
)
```

Precondition

Returns

Status

Notes

See also

2.6.4 EZosIO_GetNativeDevicePtrs( )

Description

Get native pointers for devices.

Synopsis

```
EZui32          EZosIO_GetNativeDevicePtrs (
  EZosIO_dopen_FuncPtr                          *ppfDopen,
    Pointer to native pointer for device open
  EZosIO_dclose_FuncPtr                         *ppfDclose,
    Pointer to native pointer for device close.
  EZosIO_dread_FuncPtr                          *ppfDread,
    Pointer to native pointer for device read.
  EZosIO_dwrite_FuncPtr                         *ppfDwrite,
    Pointer to native pointer for deviceoffset write.
  EZosIO_dfread_FuncPtr                         *ppfDfread,
    Pointer to native pointer for device stream read.
  EZosIO_dfwrite_FuncPtr                        *ppfDfwrite,
    Pointer to native pointer for device stream write.
  EZosIO_dioctl_FuncPtr                         *ppfDioctl
    Pointer to native pointer for device ioctl.
)
```

Precondition

Returns

Status

Notes

See also

2.6.5 EZosIO_fopen( )

Description

Open a file. Return value of pDest.

Synopsis

```
EZfile          EZosIO_fopen (
  const EZc8*                                   pcFileName,
    File name.
  const EZc8*                                   pcFileMode
    Type of access permitted.
)
```

Precondition

Returns

File handler, or EZosIO_INVALID_FILE in case of error.

Notes

See also

2.6.6 EZosIO_fprintf( )

Description

Write formatted output using a pointer to a list of arguments.

Synopsis

```
EZui32          EZosIO_fprintf (
  EZfile                                    fileStream,
    File stream.
  const EZc8*                               pcFormat,
    Line to format.
                                            …
    List of arguments.
)
```

Precondition

Returns

Returns the number of characters written, not including the terminating null character, or a negative value if an output error occurs.

Notes

See also

2.6.7 EZosIO_vfprintf( )

Description

Write formatted output using a pointer to a list of arguments.

Synopsis

```
EZui32          EZosIO_vfprintf (
  EZfile                                    fileStream,
    File stream.
  const EZc8*                               pcFormat,
    Line to format.
  EZ_VaList                                 vaList
    Pointer to list of arguments.
)
```

Precondition

Returns

Return the number of characters written, not including the terminating null character, or a negative value if an output error occurs.

Notes

See also

2.6.8 EZosIO_fclose( )

Description

Close file pointer.

Synopsis

EZui32 EZosIO_fclose (
EZfile fileStream
File stream.
)

Precondition

Returns

Notes

See also

2.6.9 EZosIO_fflush( )

Description

Copies characters between buffers. If some regions of the source area and the destination overlap, memmove ensures that the original source bytes in the overlapping region are copied before being overwritten.

Synopsis

EZui32 EZosIO_fflush (

EZfile fileStream

File stream.

)

Precondition

Returns

Notes

See also

2.6.10 EZosIO_fread( )

Description

Reads data from a stream. Return number of bytes read.

Synopsis

```
EZui32      EZosIO_fread (
  EZptr                                   pBuffer,
    Data
  EZui32                                  uiSize,
    Item size in bytes.
  EZui32                                  uiCount,
    Number of characters to copy.
  EZfile                                  fileStream
    File stream.
)
```

Precondition

Returns

Returns number of bytes read.

Notes

See also

2.6.11 EZosIO_fwrite( )

Description

Write data to stream. Return number of bytes written.

Synopsis

```
EZui32        EZosIO_fwrite (
  EZptr                                   pBuffer,
    Data
  EZui32                                  uiSize,
    Item size in bytes.
  EZui32                                  uiCount,
    Number of characters to copy.
  EZfile                                  fileStream
    File stream.
)
```

Precondition

Returns

Returns number of bytes written.

Notes

See also

2.6.12 EZosIO_fseek( )

Description

Moves the file pointer to a specified location.

Synopsis

```
EZui32          EZosIO_fseek (
  EZfile                                    fileStream,
    File stream.
  EZi32                                     iOffset,
    Number of bytes from uqOrigin.
  EZui32                                    uqOrigin
    Initial position.
)
```

Precondition

Returns

Returns current pointer position.

Notes

See also

2.6.13 EZosIO_printf( )

Description

Write formatted output using a pointer to a list of arguments.

Synopsis

```
EZui32          EZosIO_printf (
  const EZc8*                                   pcFormat,
    Line to format.
                                                ...
    List of arguments.
)
```

Precondition

Returns

Returns the number of characters written, not including the terminating null character.

Notes

See also

2.6.14 EZosIO_sprintf( )

Description

Write formatted output with arguments.

Synopsis

```
EZui32          EZosIO_sprintf (
  EZc8*                                   pcOutputString,
    Storage location for output.
  const EZc8*                             pcFormat,
    Line to format.
                                          …
    List of arguments.
)
```

Precondition

Returns

Return the number of characters written, not including the terminating null character, or a negative value if an output error occurs.

Notes

See also

2.6.15 EZosIO_vsprintf( )

Description

Write formatted output using a pointer to a list of arguments.

Synopsis

```
EZui32          EZosIO_vsprintf (
  EZc8*                                   pcOutputString,
    Storage location for output.
  const EZc8*                             pcFormat,
    Line to format.
  EZ_VaList                               vaList
    Pointer to list of arguments.
)
```

Precondition

Returns

Returns the number of characters written, not including the terminating null character, or a negative value if an output error occurs.

Notes

See also

2.6.16 EZosIO_dopen( )

Description

Open a device file. Return value of pDest.

Synopsis

```
EZfile        EZosIO_dopen (
  const EZc8*                                   pcFileName,
    File name.
  const EZc8*                                   pcFileMode
    Type of access permitted.
)
```

Precondition

Returns

File handler, or EZosIO_INVALID_FILE in case of error.

Notes

See also

2.6.17 EZosIO_dclose( )

Description

Close device file pointer.

Synopsis

EZui32 EZosIO_dclose (
EZfile fileStream
File stream.
)

Precondition

Returns

Notes

See also

2.6.18 EZosIO_dread( )

Description

Reads up to uiSize bytes from device file fileStream at offset uiOffs (from the start of the file) into the buffer starting at pBuf. The file offset is not changed.

Synopsis

```
EZui32          EZosIO_dread (
  EZfile                                    fileStream,
    File descriptor.
  EZptr                                     pBuf,
    Bufffer for write.
  EZui32                                    uiSize,
    Size to read.
  EZui32                                    uiOffs
    Offset of file to read.
)
```

Precondition

Returns

Notes

See also

2.6.19 EZosIO_dwrite( )

Description

Writes up to uiSize bytes to device file fileStream at offset uiOffs (from the start of the file) from the buffer starting at pBuf. The file offset is not changed.

Synopsis

```
EZui32          EZosIO_dwrite (
  EZfile                                    fileStream,
    File descriptor.
  EZptr                                     pBuf,
    Source buffer.
  EZui32                                    uiSize,
    Size to write.
  EZui32                                    uiOffs
     Offset of file to write.
)
```

Precondition

Returns

Notes

See also

2.6.20 EZosIO_dfread( )

Description

Read data from device. Return number of bytes read or -1 if failed.

Synopsis

```
EZui32        EZosIO_dfread (
  EZptr                                   pBuffer,
    Data.
  EZui32                                  uiSize,
    Item size in bytes.
  EZui32                                  uiCount,
    Number of characters to copy.
  EZfile                                  fileStream
    File descriptor.
)
```

Precondition

Returns

Notes

See also

2.6.21 EZosIO_dfwrite( )

Description

Write data to device. Return number of bytes written.

Synopsis

```
EZui32      EZosIO_dfwrite (
  EZptr                                   pBuffer,
    Data.
  EZui32                                  uiSize,
    Item size in bytes.
  EZui32                                  uiCount,
    Number of characters to copy
  EZfile                                  fileStream
    File descriptor.
)
```

Precondition

Returns

Notes

See also

2.6.22 EZosIO_dioctl( )

Description

Manipulates the underlying device parameters of device file fileStream.

Synopsis

```
EZui32        EZosIO_dioctl (
  EZfile                                  fileStream,
    File descriptor.
  EZui32                                  uiRequest,
    Device-dependent request code.
  EZptr                                   pCommand,
    Command.
  EZui32                                  uiSize
    Size of command.
)
```

Precondition

Returns

Notes

See also

2.7 Memory Handling Routines (EZosMem.h)

2.7.1 EZosMem_memcpy( )

Description

Memory copy.

Synopsis

```
EZptr        EZosMem_memcpy (
  EZptr                          pDest,
  const EZptr                    pSrc,
  EZui32                         uiSize
)
```

Precondition

Returns

Notes

See also

2.7.2 EZosMem_memmove( )

Description

Memory move.

Synopsis

```
EZptr        EZosMem_memmove (
  EZptr                          pDest,

  const EZptr                    pSrc,

  EZui32                         uiSize

)
```

Precondition

Returns

Notes

See also

2.7.3 EZosMem_memset( )

Description

Memory set.

Synopsis

```
EZptr        EZosMem_memset (
  EZptr                          pDest,

  EZchar                         iChar,

  EZui32                         uiSize

)
```

Precondition

Returns

Notes

See also

2.7.4 EZosMem_memcmp( )

Description

Compare memory buffers.

Synopsis

```
EZi32        EZosMem_memcmp (
  const EZptr                    pDest,

  const EZptr                    pSrc,

  EZui32                         uqSize

)
```

Precondition

Returns

Notes

See also

2.7.5 EZosMem_strcpy( )

Description

Copy string.

Synopsis

```
EZc8 *          EZosMem_strcpy (
  EZc8 *                                  pDest,

  const EZc8 *                            pSrc,

)
```

Precondition

Returns

Notes

See also

2.7.6 EZosMem_strlen( )

Description

Copy length.

Synopsis

```
EZui32          EZosMem_strlen (
  const EZc8 *                                    pDest,

)
```

Precondition

Returns

Notes

See also

2.7.7 EZosMem_strcat( )

Description

Append a string.

Synopsis

```
EZui32          EZosMem_strcat (
  const EZc8 *                                    pDest,

  const EZc8 *                                    pSrc,

)
```

Precondition

Returns

Notes

See also

2.7.8 EZosMem_strncpy( )

Description

Copy characters of one string to another.

Synopsis

```
EZui32          EZosMem_strncpy (
  const EZc8 *                          pDest,

  const EZc8 *                          pSrc,

  EZui32                                uiSize

)
```

Precondition

Returns

Notes

See also

2.7.9 EZosMem_free( )

Description

Memory free.

Synopsis

EZstatus EZosMem_free (
EZptr pMem

)

Precondition

Returns

Notes

See also

2.7.10 EZosMem_malloc( )

Description

Memory allocation.

Synopsis

```
EZptr        EZosMem_malloc (
  EZui32                                    uiSize

)
```

Precondition

Returns

Notes

See also

2.7.11 EZosMem_stricmp( )

Description

Lexicographically compares lowercase versions of pcDest and pcSource and returns a value indicating their relationship.

Synopsis

```
EZi32          EZosMem_stricmp (
  const EZc8*                          pcDest,

  const EZc8*                          pcSource

)
```

Precondition

Returns

Lexicographically compares lowercase versions of pcDest and pcSource and returns a value indicating their relationship.

Notes

See also

2.8 Miscellaneous Routines (EZosMisc.h)

Header file for OS independent miscellaneous functions.

2.8.1 EZosMisc_GetErrorNumber( )

Description

Get system error number

Synopsis

EZi32 EZosMisc_GetErrorNumber (void)

Precondition

Returns

Notes

See also

2.8.2 EZosMisc_PrintErrorNumber( )

Description

Print system error number

Synopsis

void **EZosMisc_PrintErrorNumber (**

EZi32 iErrNo

Error number.

)

Precondition

Returns

Notes

See also

2.8.3 EZosMisc_GetClock( )

Description

Get system clock.

Synopsis

EZui32 EZosMisc_GetClock(void)

Precondition

Returns

Notes

See also

2.8.4 EZosMisc_Srand( )

Description

Initialize random number generator.

Synopsis

void **EZosMisc_EZosMisc_Srand**

EZui32 uiSeed

Seed for random number generator.

)

Precondition

Returns

Notes

See also

2.8.5 EZosMisc_Rand( )

Description

Generate a random number.

Synopsis

EZui32 EZosMisc_Rand(void)

Precondition

Returns

Random number.

Notes

See also

2.9 Memory Queue Routines (EZosMsgQ.h)

Header file for OS independent message manager.

2.9.1 EZosMsgQ_CreateModule( )

Description

Create a message queue module.

Synopsis

EZstatus EZosMsgQ_CreateModule(void)

Precondition

Must be called before any other message queue function.

Returns

Notes

See also

2.9.2 EZosMsgQ_DeleteModule( )

Description

Delete a message queue module.

Synopsis

EZstatus EZosMsgQ_DeleteModule(void)

Precondition

Returns

Notes

See also

2.9.3 EZosMsgQ_Create( )

Description

Create

Synopsis

```
EZmsgq        EZosMsgQ_Create (
  EZui32                                    uiMaxMsgs,
    Maximum number of messages
  EZui32                                    uiMaxMsgLength
    Maximum length of the message.
)
```

Precondition

Returns

Notes

See also

2.9.4 EZosMsgQ_Delete( )

Description

Delete message queue.

Synopsis

EZstatus EZosMsgQ_Delete (
EZmsgq msgqId
Message queue ID.
)

Precondition

Returns

Notes

See also

2.9.5 EZosMsgQ_Receive( )

Description

Synopsis

```
EZui32        EZosMsgQ_Receive (
  EZmsgq                                  msgqId,
    ID.
  EZptr                                   pBuffer,
    Buffer with message.
  EZui32                                  uiMaxNBytes,
    Size of buffer.
  EZui32                                  uiTimeout
    Time-out to receive.
)
```

Precondition

Returns

Notes

See also

2.9.6 EZosMsgQ_Send( )

Description

Synopsis

```
EZstatus    EZosMsgQ_Send (
  EZmsgq                                    msgqId,
    ID.
  EZptr                                     pBuffer,
    Buffer with message.
  EZui32                                    uiNBytes,
    Size of buffer.
  EZui32                                    uiTimeout
    Time-out to send.
)
```

Precondition

Returns

Notes

See also

2.10 Socket Handling Routines (EZosSocket.h)

2.10.1 EZosSocket_CreateModule( )

Description

Create a socket module.

Synopsis

EZstatus EZosSocket_CreateModule(void)

Precondition

Must be called before any other socket function.

Returns

Notes

See also

2.10.2 EZosSocket_DeleteModule( )

Description

Delete a socket module.

Synopsis

EZstatus EZosSocket_DeleteModule(void)

Precondition

Must be called after any other socket function.

Returns

Notes

See also

2.10.3 EZosSocket_Create( )

Description

Create EZsock.

Synopsis

EZsock EZosSocket_Create (void)

Precondition

Returns

Returns socket ID or EZosSocket_INVALID if error.

Notes

See also

2.10.4 EZosSocket_Close( )

Description

Close EZsock.

Synopsis

EZstatus EZosSocket_Close (void)

Precondition

Returns

Notes

See also

2.10.5 EZosSocket_Shutdown( )

Description

Shutdown socket.

Synopsis

```
EZstatus        EZosSocket_Shutdown (
  EZsock                                    sockId,
    Socket to shutdown.
  EZui32                                    uiHow
    How to shutdown.
)
```

Precondition

Returns

Notes

See also

2.10.6 EZosSocket_Accept( )

Description

Accept socket.

Synopsis

EZsock EZosSocket_Accept (
EZsock sockServerId
Server socket ID.
)

Precondition

Returns

Returns socket ID or EZosSocket_INVALID if error.

Notes

See also

2.10.7 EZosSocket_Listen( )

Description

Listen to current socket for given port address and specified number of connections

Synopsis

```
EZstatus        EZosSocket_Listen (
  EZsock                                        sockServerId
    Server socket ID.
  EZui32                                        uiPortAddr,
    port address to listen.
  EZui32                                        uiNumberOfConnections
    Number of connections to wait.
)
```

Precondition

Returns

Notes

See also

2.10.8 EZosSocket_Send( )

Description

Send iSize bytes from pucBuffer.

Synopsis

```
EZstatus    EZosSocket_Send (
  EZsock                                    sockId,
    Socket descriptor.
  EZptr                                     pBuffer,
    Buffer to send.
  EZui32                                    uiSize
    Number of bytes.
)
```

Precondition

Must be connected.

Returns

Notes

See also

2.10.9 EZosSocket_Recv( )

Description

Receive iSize bytes to pucBuffer.

Synopsis

```
EZstatus    EZosSocket_Recv (
  EZsock                                    sockId,
    Socket descriptor.
  EZptr                                     pBuffer,
    Buffer to send.
  EZui32                                    uiSize
    Number of bytes.
)
```

Precondition

Must be connected.

Returns

Notes

See also

2.10.10 EZosSocket_Connect( )

Description

Connect to server in remote computer.

Synopsis

```
EZstatus        EZosSocket_Connect (
  EZsock                                    sockId,
    Structure for init.
  EZc8*                                     pcCompName,
    Remote computer name.
  EZui32                                    uiPortAddr
    Port address.
)
```

Precondition

Returns

Notes

See also

2.10.11 EZosSocket_ConnectAddr( )

Description

Connect to server.

Synopsis

```
EZstatus    EZosSocket_ConnectAddr (
  EZsock                                    sockId,
    Structure for init.
  EZuc8*                                    pucAddr,
    4 bytes address to connect.
  EZui32                                    uiPortAddr
    Port address.
)
```

Precondition

Returns

Notes

See also

2.10.12 EZosSocket_Wait( )

Description

Wait for send from socket.

Synopsis

EZstatus EZosSocket_Wait (
EZsock sockId
Socket ID.
)

Precondition

Returns

Notes

See also

2.10.13 EZosSocket_WaitMulti( )

Description

Wait for receive from multiple sockets (select).

Synopsis

```
EZstatus        EZosSocket_WaitMulti (
  EZui32                                    uiNumberOfSockets,
    number of sockets in array
  EZsock*                                   asSockId,
    Array of sockets
  EZui32                                    uiTimeout
    Timeout in microseconds.
)
```

Precondition

Returns

-2 – if no active socket in given array
-1 – if error occurred in select
0 – if timeout expired
0> – number of sockets which received signal.

When the function returns, sockets in asSockId that have not received data are cleared to EZosSocket_INVALID.

Notes

See also

2.11 Arguments Manager Routines (EZosStdarg.h)

Header file for OS independent arguments manager.

2.11.1 EZosStdarg_START( )

Description

Initialization of argument manager.

Synopsis

EZosStdarg_START (

EZ_VaList	var
Pointer to list of arguments.	
-	arg
Last argument	

)

Precondition

Returns

Notes

See also

2.11.2 EZosStdarg_END( )

Description

Clear argument manager

Synopsis

EZosStdarg_END (

EZ_VaList var

Pointer to list of arguments

)

Precondition

Returns

Notes

See also

2.11.3 EZosStdarg_COPY( )

Description

Copy argument manager

Synopsis

EZosStdarg_COPY (

EZ_VaList dst

Pointer to list of arguments (destination)

EZ_VaList src

Pointer to list of arguments (source)

)

Precondition

Returns

Notes

See also

2.12 Task Manager Routines (EZosTask.h)

Header file for OS independent task manager.

2.12.1 EZosTask_CreateModule( )

Description

Create a task module.

Synopsis

EZstatus EZosTask_CreateModule(void)

Precondition

Must be called before any other task function.

Returns

Notes

See also

2.12.2 EZosTask_DeleteModule( )

Description

Delete a task module.

Synopsis

EZstatus EZosTask_DeleteModule(void)

Precondition

Returns

Notes

See also

2.12.3 EZosTask_SemaphoreCreate( )

Description

Create a counted semaphore.

Synopsis

EZstatus EZosTask_SemaphoreCreate (
EZui32 uiCount
Number of counts.
)

Precondition

Returns

Notes

See also

2.12.4 EZosTask_SemaphoreDestroy( )

Description

Destroy the semaphore.

Synopsis

EZstatus EZosTask_SemaphoreDestroy (
EZsem semId
Semaphore ID.
)

Precondition

Returns

Notes

See also

2.12.5 EZosTask_SemaphoreTake( )

Description

Wait for the semaphore uiTime milliseconds and take it.

Synopsis

```
EZstatus    EZosTask_SemaphoreTake (
  EZsem                                    semId
    Semaphore ID.
  EZui32                                   uiTime
    Time in milliseconds.
)
```

Precondition

Returns

Notes

See also

2.12.6 EZosTask_SemaphoreGive( )

Description

Release the semaphore.

Synopsis

EZstatus EZosTask_SemaphoreGive (
EZsem semId
Semaphore ID.
)

Precondition

Returns

Notes

See also

2.12.7 EZosTask_MutexCreate( )

Description

Create reentrant mutex.

Synopsis

```
EZmtx        EZosTask_MutexCreate (
  EZbool                                    bRecursive
    When TRUE, mutex is recursive.
)
```

Precondition

Returns

Notes

See also

2.12.8 EZosTask_MutexDestroy( )

Description

Destroy mutex.

Synopsis

EZstatus EZosTask_MutexDestroy (
EZmtx mtxId
Mutex ID.
)

Precondition

Returns

Notes

See also

2.12.9 EZosTask_MutexLock( )

Description

Lock mutex.

Synopsis

EZstatus EZosTask_MutexLock (
EZmtx mtxId
Mutex ID.
)

Precondition

Returns

Notes

See also

2.12.10 EZosTask_MutexUnlock( )

Description

Unlock previously locked mutex.

Synopsis

EZstatus EZosTask_MutexUnlock (
EZmtx mtxId
Mutex ID.
)

Precondition

Returns

Notes

See also

2.12.11 EZosTask_Delay( )

Description

Delay of task in given milliseconds.

Synopsis

EZstatus EZosTask_Delay (
EZui32 uiDelayInMilliseconds
Delay in milliseconds.
)

Precondition

Returns

Notes

See also

2.12.12 EZosTask_MicroDelay( )

Description

Delay of task in given microseconds.

Synopsis

EZstatus EZosTask_MicroDelay (
EZui32 uiDelayInMicroSeconds
Delay in microseconds.
)

Precondition

Returns

Notes

See also

2.12.13 EZosTask_Spawn( )

Description

Start new task with name, priority, stack size and argument.

Synopsis

```
EZtask          EZosTask_Spawn (
  const EZc8*                          pcName,
  EZui32                               uiPri,
  EZui32                               uiStackSize,
  EZosTask_Spawn_FuncPtr               pFun,
  EZptr                                pArg
)
```

Precondition

Returns

Notes

See also

2.12.14 EZosTask_GetId( )

Description

Get ID of the current task.

Synopsis

EZtask **EZosTask_GetId (void)**

Precondition

Returns

Notes

See also

2.12.15 EZosTask_Exit( )

Description

Exit from process with specific error code.

Synopsis

```
void        EZosTask_Exit (
  EZui32                                    uiExitCode

)
```

Precondition

Returns

Notes

See also

2.13 Time Measurement Routines (EZosTime.h)

Header file for OS time measurements.

2.13.1 EZosTime_CreateModule( )

Description

Create module and take base time-stamp.

Synopsis

EZstatus EZosTime_CreateModule (void)

Precondition

Must be called before any other time function.

Returns

Notes

See also

2.13.2 EZosTime_DestroyModule( )

Description

Delete a time module.

Synopsis

EZstatus EZosTime_DestroyModule (void)

Precondition

Returns

Notes

See also

2.13.3 EZosTime_GetCurrentTimeStamp( )

Description

Get current time.
The time is relative to the taken base time-stamp.

Synopsis

EZstatus EZosTime_GetCurrentTimeStamp (void)

Precondition

Returns

Notes

See also

2.13.4 EZosTime_TimeDifference( )

Description

Calculates difference between to time values.

Synopsis

```
EZstatus        EZosTime_TimeDifference (
  EZtime                                        *ptmStartTime
    Start time.
  EZtime                                        *ptmEndTime
    End time.
  EZui32                                        *puiSec
    Pointer to seconds value.
  EZui32                                        *puiNanoSec
    Pointer to nanoseconds value.
)
```

Precondition

Returns

puiSec - difference between points in the seconds. Can be NULL.
puiNanoSec - difference between points in the nanoseconds (fraction of seconds).

Notes

See also

2.13.5 EZosTime_InitTimeValue( )

Description

Init time value.

```
EZstatus        EZosTime_InitTimeValue (
  EZui32                                  *ptmTime
    Init time value.
)
```

Precondition

Returns

Notes

See also

2.13.6 EZosTime_SetTimeValue( )

Description

Set time value.

```
EZstatus        EZosTime_SetTimeValue (
  EZtime                                    *ptmTime
    Time value.
  EZui32                                    uiSec
    Seconds.
  EZui32                                    uiNanoSec
    Nanoseconds.
)
```

Precondition

Returns

Notes

See also

2.13.7 EZosTime_CopyTimeValue( )

Description

Copy time value.

```
EZstatus        EZosTime_CopyTimeValue (
  EZtime                                    *ptmDestTime
    Destination time value.
  EZtime                                    *ptmSrcTime
    Source time value.
)
```

Precondition

Returns

Notes

See also

2.13.8 EZosTime_AddTimeDifference( )

Description

Calculates difference between two time values and adds to ptmTime.

```
EZstatus        EZosTime_AddTimeDifference (
  EZtime                                        *ptmTime
    Time.
  EZtime                                        *ptmStartTime
    Start time.
  EZtime                                        *ptmEndTime
    End time.
)
```

Precondition

Returns

Notes

See also

2.13.9 EZosTime_PrintTime( )

Description

Print in specific format.

```
EZc8 *          EZosTime_PrintTime (
  EZc8                                          *pcString

  EZosTime_PRINT_MODE                           eFormat
    Print mode format.
  EZui32                                        uiSec
    Seconds value.
  EZui32                                        uiNanoSec
    Nanoseconds value.
)
```

Associated Structures

```
enum            EZosTime_PRINT_MODE {
  EZosTime_PRINT_MODE_SECONDS = 1,
    Print time in seconds.
    Only time.
    Example: 2
  EZosTime_PRINT_MODE_MILLISECONDS,
    Print time in milliseconds.
    Only time.
    Example: 2 , 2500
  EZosTime_PRINT_MODE_MICROSECONDS,
    Print time in microseconds.
    Only time.
    Example: 2 , 2500000
  EZosTime_PRINT_MODE_NANOSECONDS,
    Print time in nanoseconds.
    Only time.
    Example: 2 , 2500000000
  EZosTime_PRINT_MODE_AUTO   = 10
    Print only significant values of the time.
    Example: 2 sec. 510 milisec.
}
```

Precondition

Returns

Notes

See also

2.13.10 EZosTime_PrintTimeDifference( )

Description

Print difference between two time-points in a specific format.

```
EZc8 *          EZosTime_PrintTimeDifference (
  EZc8                                          *pcString

  EZosTime_PRINT_MODE                           eFormat
    Print mode format.
  EZtime                                        *ptmStartTime
    Start time.
  EZtime                                        *ptmEndTime
    End time.
)
```

Precondition

Returns

Notes

See also

2.13.11 EZosTime_PrintTimeValue( )

Description

Print time value in specific format.

```
EZc8 *          EZosTime_PrintTimeValue (
  EZc8                                          *pcString

  EZosTime_PRINT_MODE                           eFormat
    Print mode format.
  EZtime                                        *ptmTime
    Time.
)
```

Precondition

Returns

Notes

See also

3. Appendix A: Preprocessor Definitions

The following preprocessor definition may be used to control EZenv related functionality:

Operating system (EZdef.h):

- EZ_OS_WIN – Windows operating system (default)
- EZ_OS_VXWORKS – VxWorks operating system
- EZ_OS_LINUX_USER – Linux/Unix operating system, user space
- EZ_OS_LINUX_KERNEL – Linux/Unix operating system, kernel space

CPU Endianness (EZdef.h):

- EZdef_ENDIAN_LITTLE – Compile for little endian CPU (default)
- EZdef_ENDIAN_BIG – Compile for big endian CPU

PCI Swap (EZdef.h):

- EZdef_PCI_NO_SWAP – The OS or hardware passed PCI data to the NPS as is (default)
- EZdef_PCI_SWAP – The OS or hardware swaps each 4 bytes of PCI data

CPU alignment definitions (EZdef.h):

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

CPU address/pointer size definitions (EZdef.h):

- EZ_CPU_ADDRESS_32_BIT – Compile for 32-bit CPU (default)
- EZ_CPU_ADDRESS_64_BIT – Compile for 64-bit CPU

CPU type (external/embedded) definitions (EZdef.h):

- EZ_CPU_TYPE_EXTERNAL – Compile for external CPU (default).
- EZ_CPU_TYPE_EMBEDDED – Compile for embedded CPU.

Development level (EZdevL.h):

- EZdevL_USER_LEVEL – User level
- EZdevL_NOTE_LEVEL – Note level (default)
- EZdevL_MAINTENANCE_LEVEL – Maintenance level
- EZdevL_DEBUG_LEVEL – Debug level