Avionics Databus Solutions

ARINC664 / EDE

Interface Module

Reference Manual

V2.1.x Rev. D

March 2018

www.aim-online.com

ARINC664 / EDE

Software Library
Reference Manual

Reference

Manual

V2.1.x Rev. D

March 2018

AIM NO.
60-15901-36-2.1.X

AIM – Gesellschaft für angewandte Informatik und Mikroelektronik mbH

AIM GmbH
Sasbacher Str. 2
D-79111 Freiburg / Germany
Phone +49 (0)761 4 52 29-0
Fax +49 (0)761 4 52 29-33
sales@aim-online.com

AIM UK Office
Cressex Enterprise Centre, Lincoln Rd.
High Wycombe, Bucks. HP12 3RB / UK
Phone +44 (0)1494-446844
Fax +44 (0)1494-449324
salesuk@aim-online.com

AIM GmbH – Munich Sales Office
Terofalstr. 23a
D-80689 München / Germany
Phone +49 (0)89 70 92 92-92
Fax +49 (0)89 70 92 92-94
salesgermany@aim-online.com

AIM USA LLC
Seven Neshaminy Interplex
Suite 211 Trevose, PA 19053
Phone 267-982-2600
Fax 215-645-1580
salesusa@aim-online.com

Notice: The information that is provided in this document is believed to be accurate. No responsibility is assumed by AIM GmbH for its use. No license or rights are granted by implication in connection therewith. Specifications are subject to change without notice.

DOCUMENT HISTORY

The following table defines the history of this document. Section 6 provides a more comprehensive list of changes made with each version.

Version	Cover Date	Created by	Description
1.00 Rev A	31.03.2006	T. Troshynski	See Section 5 for details
1.00 Rev B	07.07.2006	T. Troshynski	See Section 5 for details
1.00 Rev C	07.21.2006	T. Troshynski	See Section 5 for details
1.00 Rev D	10.09.2006	T. Troshynski	See Section 5 for details
2.00 Rev A	15.11.2006	T. Troshynski	See Section 5 for details
2.00 Rev B	29.01.2007	T. Troshynski	See Section 5 for details
2.00 Rev C	21.03.2007	T. Troshynski	See Section 5 for details
2.00 Rev D	01.10.2007	R. Heitzmann	See Section 5 for details
2.0.1 Rev E	19.09.2017	R. Heitzmann	New Front Cover
2.1.x Rev C	05.12.2017	R. Heitzmann	See Section 5 for details
2.1.x Rev D [1]	08.03.2018	R. Heitzmann	See Section 5 for details

[1] Rev. A and Rev. B were preliminarry distributet without updated history details.

Table of Contents

List of Tables

List of Figures

1. INTRODUCTION

1.1 General

The AyI / AMC -FDX High Level Application Interface Library provides a comprehensive set of 'C' functions for interfacing application programs to the AIM AFDX Interface Modules listed below. The ‘y’ in the ‘AyI’ is an AIM standard placeholder for encoding the module’s platform (where ‘y’ can be replaced with either C, V, or P as shown below).

C:	***ACI-FDX-2/4*** Compact PCI (cPCI) 6U
P:	***API-FDX-2*** PCI
AMC:	***AMC-FDX-2*** PMC-Module

This document defines the extensions to the FDX High Level Application Interface Library supporting the Boeing 787 program specific Error Detection Encoding (EDE) features. These extension functions may only be used in conjunction with the following board types:

- API-FDX-2B
- AMC-FDX-2B

1.2 Applicable Documents

The following documents shall be considered to be part of this document to the extent that they are referenced herein.

[1] API/ACI/AMC-FDX Reference Manual, V17.1x Rev. C March 2014

[2] AIM Reference Manual “VME Generic Interface Library”

[3] AMC/ACI-FDX Programmer’s Guide for VxWorks Applications

[4] API/ACI/AMC-FDX Programmer’s Guide for Windows Applications

[5] Interoperability Specification for the 787 End System,
D616Z004-01 Rev E December 7, 2015

2. APPLICATION INTERFACING

2.1 General

To interface the user's application program to the target hardware, the application program is required to call the basic functions of the FDX Application Interface Library as described in section 2 of [1].

2.2 Error Reporting

Each function of the B787 Extension to the FDX Interface Library has defined return values. For a successful function call the function returns a zero. For an unsuccessful function call the function returns a negative return value. These return values may be classified in prioritized groups of e.g. errors, warnings and information.

In addition to the return value, a defined Error Handler for special error reporting will be invoked by the Library. The error handler is an encapsulated function inside the Library with a defined interface.

2.3 Necessary Files and Defines

For all platforms two C-syntax header files, **AI_CDEF.H** and **AIFDX_DEF.H,** are provided which contain all information concerning constants, data types and function prototypes for the standard AyI/AMC-FDX High Level Application Interface Library. The application program only has to include **AIFDX_EDE_DEF.H** which itself includes **AI_CDEF.H** and **AIFDX_DEF.H** in addition to defining all of the constants, data types and function prototypes of the B787 specific functions.

For VME platforms an additional C syntax header files is required: **AIVME_DEF.H**.

The application program must enter the following preprocessor definition (e.g. usually /D or -D option of the C-compiler):

_AIM_FDX and _AIM_WINDOWS

For using the 32-Bit DLL (Win98/ME/2000/XP applications) the **aim_fdx.lib** import library must be linked to the application.

The calling convention for 32-Bit AIM_FDX Application Interface DLLs is ***stdcall*** (Windows 98, Windows ME, Windows 2000 and Windows XP).

For the operating systems mentioned above the FDX import library is available in 2 versions. One import library is compatible with BORLAND C/C++ compiler series (5.02 or higher) and another import library is compatible with the MICROSOFT Visual C/C++ compiler series (6.0 for 32-Bit applications). LabWindows/CVI supports compatibility modes for both import libraries. So it is able to work with both libraries.

_AIM_FDX and _AIM_VME

For using the Library in embedded VME environment, running under any operating system.

__LYNX_OS__	Additional define, for using the Library under LynxOS (supported version: 3.1.0) .
__VXWORKS__	Additional define, for using the Library under VxWorks.
HOST_ENDIAN_BIG or	
HOST_ENDIAN_LITTLE	For switching endian order to support Little Endian and Big Endian Systems.
_BSP_PC486_CP604	Support for ACI-FDX-4 with Pep Modular Computers CP604 cCPI, PentiumIII (VxWorks).

When using the Library in embedded VME environment, the source code is provided.

2.4 Files of the FDX High Level Interface

2.4.1 Include Files

The following B787 extensions to the FDX Application Interface Library 'C' syntax include file is valid for all platforms:

aifdx_ede_def.h

It defines all Function Prototypes, Data types and Constants.

2.4.2 Libraries and Files

Table 2.4-1: Necessary Application Interface Level Files

Operating System Platform	*Files*	*Comment*
WindowsXP. Windows 7/8.x	aim_fdx.dll	32-bit DLL
WindowsXP. Windows 7/8.x	aim_fdx.lib	Corresponding 32-bit import library

2.4.3 System Level Driver Files

Table 2.4-2: Necessary System Level Driver Files

Operating System Platform	*Files*	*Comment*
WindowsXP. Windows 7/8.x	Aim_Fdx.sys	WDM Kernel Mode Device Driver for ACI-FDX
	Aim_Fdx.sys	WDM Kernel Mode Device Driver for ACI-FDX (Win 98)

3. Function Reference

This chapter contains a reference for all FDX High Level Library 'C' functions. Special data type definitions are described with the corresponding 'C' function which is using the data type.

The first parameter of each function is called "***ul_Handle***" and determines the FDX destination resource. This handle is returned by the login function at login time as a unique handle to that resource.

This parameter is mentioned but not described for each function since the parameter must be given for all functions with the exception of the system related functions.

All Functions with parameter "***ul_Handle***" can additionally return the following error codes:

FDX_CLIENTHANDLE_INVALID

FDX_RESOURCEID_INVALID

FDX_RESOURCETYPE_INVALID

3.1 EDE Global Functions

The Handle input parameter to the following functions must be a port related one.

Table 3.1-1:Global Functions

Function	*Description*
Global EDE Functions	
FdxCmdReadEDECounter	Reads the current value of the Hardware EDE counter

3.1.1 FdxCmdReadEDECounter

Prototype:

AiReturn FdxCmdReadEDECounter (AiUInt32 ul_Handle,
TY_FDX_READ_EDE_CNTR_OUT* px_ReadEDECounterOut);

Driver Command:

FDX_READ_EDE_COUNTER (0x0000806C)

Purpose:

This function is used to read the current value of the EDE reference counter which is used to derive the EDE transmit timestamps. This function is also used to read the IRIG time at which the EDE reference counter was reset to 0.

Input:

None

Output:

TY_FDX_READ_EDE_CNTR_OUT*
px_ReadEDECounterOut

Pointer to the output data.

```
typedef struct {
   AiUInt32         ul_EDECounterHigh;
   AiUInt32         ul_EDECounterLow;
   TY_FDX_IRIG_TIME  x_EDEZeroTime;
}TY_FDX_READ_EDE_CNTR_OUT;
```

AiUInt32 ul_EDECounterHigh

31……………………………………………………16	*15……………………………………………………0*
Reserved (Coded 0)	EDE reference counter (bit 32 – 47)

AiUInt32 ul_EDECounterLow

31………0
EDE reference counter (bit 0 – 31)

The parameters ul_EDECounterHigh and ul_EDECounterLow combine to form the current value of the 48-bit EDE reference counter.

TY_FDX_IRIG_TIME x_EDEZeroTime

The IRIG time at which the EDE reference counter was last reset to 0.

Return Value

Returns FDX_OK on success or a negative error code on error.
Error Codes: FDX_ERR

3.2 EDE Subscriber Functions

This section describes the EDE subscriber functionality of the FDX-2/4B Board. The following EDE subscriber functions are available.

The Handle input parameter to the following functions must be a port related one.

Table 3.2-1: EDE Subscriber Functions

Function	*Description*
FdxCmdEDESubCreate	Creates a new EDE subscriber
FdxCmdEDESubCreateEx1	Creates a new EDE subscriber
FdxCmdEDESubClkControl	Controls EDE subscriber clock operation
FdxCmdEDESubControl	Controls EDE subscriber operation
FdxCmdEDESubControlEx	Controls EDE subscriber operation with extended options
FdxCmdEDESubControlEx2	Controls EDE subscriber operation with extended options
FdxCmdEDESubTMDef	Defines a connection to a Time Manager
FdxCmdEDESubDestroy	Destroys a configured EDE subscriber

3.2.1 FdxCmdEDESubCreate

Prototype:

AiReturn FdxCmdEDESubCreate (AiUInt32 ul_Handle,
const TY_FDX_EDE_SUB_CREATE_IN* px_EDESubCreateIn,
TY_FDX_EDE_SUB_CREATE_OUT* px_EDESubCreateOut);

Driver Command:

FDX_EDE_SUB_CREATE (0x00008040)

Purpose:

This function is used to create and configure the communication parameters needed to simulate an EDE Subscriber.

Input:

TY_FDX_EDE_SUB_CREATE_IN px_EDESubCreateIn*

Pointer to the input EDE Subscriber create structure.

```
typedef struct {
     AiUINt32                           ul_OwnSubscriberIndex;
} TY_FDX_EDE_SUB_CREATE_IN;
```

AiUInt32 ul_OwnSubscriberIndex

The subscribers index within the Time offset and time stamp list table distributed by the EDE Time managers.

Output

TY_FDX_EDE_SUB_CREATE_OUT px_EDESubCreateOut*

Pointer to the EDE subscriber create output structure.

```
typedef struct {
     AiUInt32              ul_EDESubscriberHandle;
} TY_FDX_EDE_SUB_CREATE_OUT;
```

AiUInt32 ul_EDESubscriberHandle

The handle to the EDE Subscriber.

Return Value

Returns FDX_OK on success or a negative error code on error.

Error Codes: FDX_ERR

3.2.2 FdxCmdEDESubCreateEx1

Prototype:

AiReturn FdxCmdEDESubCreateEx1 (AiUInt32 ul_Handle,
const TY_FDX_EDE_SUB_CREATE_EX1_IN* px_EDESubCreateIn,
TY_FDX_EDE_SUB_CREATE_OUT* px_EDESubCreateOut);

Driver Command:

FDX_EDE_SUB_CREATE_EX1

Purpose:

This function is used to create and configure the communication parameters needed to simulate an EDE Subscriber.

Input:

TY_FDX_EDE_SUB_CREATE_EX1_IN px_EDESubCreateIn*

Pointer to the input EDE Subscriber create structure.

```
typedef struct {
     AiUInt32    ul_OwnSubscriberIndex;
     AiUInt32    ul_ES_ConfigurationId;
     AiUInt32    ul_TimeResponseMessageTypeLength;
     AiUInt32    ul_Reserved2;
} TY_FDX_EDE_SUB_CREATE_EX1_IN;
```

AiUInt32 ul_OwnSubscriberIndex

The subscribers index within the Time offset and time stamp list table distributed by the EDE Time managers.

AiUInt32 ul_ES_ConfigurationId

The End System shall set the ES Configuration ID field to the Configuration Version Number parameter value configured for the E/S.

AiUInt32 ul_ TimeResponseMessageTypeLength

The Time Manager Response has changed in lengt with Boeing “Interoperability Specification for the 787 and 777X End System” Revision E. This parameter is to select the type and length of the Time Manager Response Message.

Constant	*Description*
FDX_EDE_TM_RESP_A	Set Time Manager Response Message to original Length (16 Byte)
FDX_EDE_TM_RESP_B	Set Time Manager Response to extended length as described in document Rev. E (48 Byte
FDX_EDE_TM_RESP_C	Set Time Manager Response to customer Request length (32 Bytes)
FDX_EDE_TM_RESP_X	Set Time Manager Response to an individual length. Please specify the length with the Parameter ul_Reserved2 in Byte

AiUInt32 ul_Reserved2

See ul_TimeResponseMessageTypeLength.

Output

TY_FDX_EDE_SUB_CREATE_OUT px_EDESubCreateOut*

Pointer to the EDE subscriber create output structure.

```
typedef struct {
    AiUInt32            ul_EDESubscriberHandle;
} TY_FDX_EDE_SUB_CREATE_OUT;
```

AiUInt32 ul_EDESubscriberHandle

The handle to the EDE Subscriber.

Return Value

Returns FDX_OK on success or a negative error code on error.

Error Codes: FDX_ERR

3.2.3 FdxCmdEDESubClkControl

Prototype:

AiReturn FdxCmdEDESubClkControl (AiUInt32 ul_Handle,
const AiUInt32 ul_EDESubscriberHandle,
const TY_FDX_EDE_SUB_CLK_CNTL_IN* px_EDESubClkCntlIn,
TY_FDX_EDE_SUB_CLK_CNTL_OUT* px_EDESubClkCntlOut);

Driver Command:

FDX_EDE_SUB_CLK_CONTROL (0x00008043)

Purpose:

This function is used to control the EDE subscriber Clock.

Input:

AiUInt32 ul_EDESubscriberHandle

The handle to the EDE subscriber

TY_FDX_EDE_SUB_CLK_CNTL_IN px_EDESubClkCntlIn*

Pointer to the subscriber clock input structure.

```
typedef struct {
   AiUInt32             ul_ControlMode;
   AiUInt32             ul_ClkHigh;
   AiUInt32             ul_ClkLow;
} TY_FDX_EDE_SUB_CLK_CNTL_IN;
```

AiUInt32 ul_ControlMode

Specifies how the clock will be modified.

Constant	*Description*
FDX_EDE_CLK_ABSOLUTE	Set the absolute value of the subscriber clock using ul_ClkHigh and ul_ClkLow
FDX_EDE_CLK_OFFSET	Offset the current subscriber clock by the relative time specified in x_Offset
FDX_EDE_CLK_READ	The current value of the subscriber clock is read and returned in the output data structure.

AiUint32 ul_ClkHigh

The lower 16-bits of this parameter specify the high order 16 bits of the 48 bit subscriber clock.

AiUInt32 ul_ClkLow

Specifies the lower order 32 bits of the 48 bit subscriber clock.

Output

TY_FDX_EDE_SUB_CLK_CNTL_OUT* px_EDESubClkCntlOut

Pointer to the subscriber clock output structure. This structure contains the current value of the EDE subscriber clock.

```
typedef struct {
   AiUInt32 ul_ClkHigh;
   AiUInt32 ul_ClkLow;
}TY_FDX_EDE_SUB_CLK_CNTL_OUT;
```

AiUint32 ul_ClkHigh

The lower 16-bits of this parameter specify the high order 16 bits of the 48 bit subscriber clock.

AiUInt32 ul_ClkLow

Specifies the lower order 32 bits of the 48 bit subscriber clock.

Return Value

Returns FDX_OK on success or a negative error code on error.

Error Codes: FDX_ERR

3.2.4 FdxCmdEDESubControl

Prototype:

AiReturn FdxCmdEDESubControl (AiUInt32 ul_Handle,
const AiUInt32 ul_ EDESubscriberHandle,
const TY_FDX_EDE_SUB_CNTL_IN* px_EDESubCntlIn);

Driver Command:

FDX_EDE_SUB_CONTROL (0x00008041)

Purpose:

This function is used to control the EDE subscriber.

Input:

AiUInt32 ul_EDESubscriberHandle

The handle to the EDE Subscriber.

TY_FDX_EDE_SUB_CNTL_IN px_EDESubCntlIn*

Pointer to the EDE Subscriber Control input structure.

```
typedef struct {
   AiUInt32    ul_Enable;
   AiUInt32    ul_ControlMode;
   AiUInt32    ul_TSListTO;
   AiUInt32    ul_TimeOffsetTO;
   AiUInt32    ul_ManualControl;
}TY_FDX_EDE_SUB_CNTL_IN;
```

AiUInt32 ul_Enable

Specifies the Enable State of the EDE Subscriber. This defines whether or not the EDE Subscriber responds to Time Request messages from the EDE Time Manager(s).

Constant	*Description*
FDX_EDE_SUB_ENA	Enabled
FDX_EDE_SUB_DIS	Disabled

AiUInt32 ul_ControlMode

Specifies how the EDE subscriber is controlled.

Constant	*Description*
FDX_EDE_SUB_AUTO	The EDE subscriber is automatically controlled. The Offset tables and Inhibit state of the subscriber are automatically controlled by the Time Manager interaction as specified in the ES Interop specification.

AiUInt32 ul_TSListTO

The TSList timeout. This parameter is only valid when ul_ControlMode = FDX_SUB_AUTO. This defines the amount of time, in milliseconds, that the EDE subscriber waits to receive a TS_List pair before a TS_List timeout occurs.

AiUInt32 ul_TimeOffsetTO

The Time Offset timeout. This parameter is only valid when ul_ControlMode = FDX_SUB_AUTO. This defines the amount of time, in milliseconds, that the EDE subscriber waits to receive a Time Offset Table pair before a TimeOffset timeout occurs and the subscriber sets all Offsets to “Offset Unknown”.

AiUInt32 ul_ManualControl

Reserved

Output:

None.

Return Value

Returns FDX_OK on success or a negative error code on error.

Error Codes: FDX_ERR

3.2.5 FdxCmdEDESubControlEx

Prototype:

AiReturn FdxCmdEDESubControlEx (AiUInt32 ul_Handle,
const AiUInt32 ul_ EDESubscriberHandle,
const TY_FDX_EDE_SUB_CNTL_IN px_EDESubCntlIn,*
const TY_FDX_EDE_SUB_CNTL_EX_IN px_EDESubCntlExIn);*

Driver Command:

FDX_EDE_SUB_CONTROL_EX (0x00008045)

Purpose:

This function is used to control the EDE subscriber. It provides extended functionality which allows the setting of the EDE Validation parameters.

Input:

AiUInt32 ul_EDESubscriberHandle

The handle to the EDE Subscriber.

TY_FDX_EDE_SUB_CNTL_IN px_EDESubCntlIn*

Pointer to the EDE Subscriber Control input structure.

```
typedef struct {
   AiUInt32    ul_Enable;
   AiUInt32    ul_ControlMode;
   AiUInt32    ul_TSListTO;
   AiUInt32    ul_TimeOffsetTO;
   AiUInt32    ul_ManualControl;
}TY_FDX_EDE_SUB_CNTL_IN;
```

AiUInt32 ul_Enable

Specifies the Enable State of the EDE Subscriber. This defines whether or not the EDE Subscriber responds to Time Request messages from the EDE Time Manager(s).

Constant	*Description*
FDX_EDE_SUB_ENA	Enabled
FDX_EDE_SUB_DIS	Disabled

AiUInt32 ul_ControlMode

Specifies how the EDE subscriber is controlled.

Constant	*Description*
FDX_EDE_SUB_AUTO	The EDE subscriber is automatically controlled. The Offset tables and Inhibit state of the subscriber are automatically controlled by the Time Manager interaction as specified in the ES Interop specification.

AiUInt32 ul_TSListTO

The TSList timeout. This parameter is only valid when ul_ControlMode = FDX_SUB_AUTO. This defines the amount of time, in milliseconds, that the EDE subscriber waits to receive a TS_List pair before a TS_List timeout occurs.

AiUInt32 ul_TimeOffsetTO

The Time Offset timeout. This parameter is only valid when ul_ControlMode = FDX_SUB_AUTO. This defines the amount of time, in milliseconds, that the EDE subscriber waits to receive a Time Offset Table pair before a TimeOffset timeout occurs and the subscriber sets all Offsets to “Offset Unknown”.

AiUInt32 ul_ManualControl

Reserved

*TY_FDX_EDE_SUB_CNTL_EX_IN *px_EDESubCntlExIn*

Pointer to the Extended subscriber control input structure.

```
typedef struct {
    AiUInt32     ul_TOTIterationRate;
    AiUInt32     ul_ReqNumValidWindow;
    AiUInt32     ul_TREQMaxAge;
    AiUInt32     ul_TOTMaxAge;
    AiUInt32     ul_MyOS1Max;
    AiUInt32     ul_OtherOS1Max;
    AiUInt32     ul_RxClockError;
     AiUInt32     ul_EDEOffsetUnkHigh;
     AiUInt32     ul_EDEOffsetUnkLow;
}TY_FDX_EDE_SUB_CNTL_EX_IN;
```

AiUInt32 ul_TOTIterationRate

Defines the rate, in milliseconds, at which the EDE subscriber checks the Time Offset Table ports of the Time Manager connections for a valid time offset table.

When the EDE Subscriber is created, a default value of 1 second is used to initialize this parameter.

(See [5]: EIO-7440)

AiUInt32 ul_ReqNumValidWindow

Defines the Time Request number valid window to be used to validate the request number of the Time Offset Tables. The following must be true for the Time Offset Table message to be accepted:

Last TREQ Req. Num. >= Req. Num of TOT >= Last TREQ Req. Num. – ul_ReqNumValidWindow.

When the EDE Subscriber is created, a default value of 2 is used to initialize this parameter.

(See [5]: EIO-5897)

AiUInt32 ul_TREQMaxAge

Defines maximum amount of time, in microseconds, since the last received Time Request message on a Time Manager connection in order for the Time Offset Table from the connection to be used by the Subscriber.

When the EDE Subscriber is created, a default value of 2.5 Sec is used to initialize this parameter.

(See [5]: EIO-7467)

AiUInt32 ul_TOTMaxAge

Defines the maximum amount of time, in microseconds, since the reception of the Time Offset message on the Time Manager connection. This is the maximum amount of time allowed since the last receipt of a Time Offset Table message in order for the message to be used.

When the EDE Subscriber is created, a default value of 2.5 Sec is used to initialize this parameter.

(See [5]: EIO-7468)

AiUInt32 ul_MyOS1Max

Defines the valid value range, in microseconds, for the Offset time (Rx TS – Tx TS) of the Time Offset Table message containing ‘myROS’.

For the Time Offset table message (containing ‘my ROS’) to be accepted, the following must be true:

myROS – ul_MyOS1Max <= Rx TS – Tx TS <= myROS + ul_MyOS1Max

where Rx TS and Tx TS are the Rx and Tx EDE timestamps of the Time Offset Table message.

When the EDE Subscriber is created, a default value of 100 mSec is used to initialize this parameter.

(See [5]: EIO-7037)

AiUInt32 ul_OtherOS1Max

Defines the maximum time difference, in microseconds, between the EDE Tx Timestamp of the Time Offset message containing 'myROS' and the EDE Tx Timestamp of the Time Offset message NOT containing 'myROS'.

When the EDE Subscriber is created, a default value of 1500 mSec is used to initialize this parameter.

(See [5]: EIO-7037)

AiUInt32 ul_RxClockError

The 'Rx Clock Error', in microseconds, used during the calculation of the EDE Offsets for each remote subscriber in the Time Offset Tables. This defines the difference between the Rx TS of received messages and the actual value of the ES EDE clock at the time of the receipt of the message.

When the EDE Subscriber is created, a default value of 0 is used to initialize this parameter.

(See [5]: EIO-5899)

AiUInt32 ul_EDEOffsetUnkHigh

Bits 0-15 of this parameter define the high order2 bytes of the EDE Offset Unknown value to be used by the EDE subscriber

When the EDE Subscriber is created, a default value of 0x7FFFFFFFFFFF is used to initialize the offset unknown parameter for the EDE subscriber.

AiUInt32 ul_EDEOffsetUnkLow

Defines the low order 4 bytes of the EDE Offset Unknown value to be used by the EDE subscriber.

When the EDE Subscriber is created, a default value of 0x7FFFFFFFFFFF is used to initialize the offset unknown parameter for the EDE subscriber.

Output:

None.

Return Value

Returns FDX_OK on success or a negative error code on error.

Error Codes: FDX_ERR

3.2.6 FdxCmdEDESubControlEx2

Prototype:

AiReturn FdxCmdEDESubControlEx2 (AiUInt32 ul_Handle,
const AiUInt32 ul_ EDESubscriberHandle,
const TY_FDX_EDE_SUB_CNTL_IN* px_EDESubCntlIn,
const TY_FDX_EDE_SUB_CNTL_EX_IN* px_EDESubCntlEx1In,
const TY_FDX_EDE_SUB_CNTL_EX2_IN* px_EDESubCntlEx2In);

Driver Command:

FDX_EDE_SUB_CONTROL_EX2

Purpose:

This function is used to control the EDE subscriber. It provides extended functionality which allows the setting of the EDE Validation parameters.

Input:

AiUInt32 ul_EDESubscriberHandle

The handle to the EDE Subscriber.

TY_FDX_EDE_SUB_CNTL_IN px_EDESubCntlIn*

Pointer to the EDE Subscriber Control input structure.

```
typedef struct {
   AiUInt32    ul_Enable;
   AiUInt32    ul_ControlMode;
   AiUInt32    ul_TSListTO;
   AiUInt32    ul_TimeOffsetTO;
   AiUInt32    ul_ManualControl;
}TY_FDX_EDE_SUB_CNTL_IN;
```

AiUInt32 ul_Enable

Specifies the Enable State of the EDE Subscriber. This defines whether or not the EDE Subscriber responds to Time Request messages from the EDE Time Manager(s).

Constant	*Description*
FDX_EDE_SUB_ENA	Enabled
FDX_EDE_SUB_DIS	Disabled

AiUInt32 ul_ControlMode

Specifies how the EDE subscriber is controlled.

Constant	*Description*
FDX_EDE_SUB_AUTO	The EDE subscriber is automatically controlled. The Offset tables and Inhibit state of the subscriber are automatically controlled by the Time Manager interaction as specified in the ES Interop specification.

AiUInt32 ul_TSListTO

The TSList timeout. This parameter is only valid when ul_ControlMode = FDX_SUB_AUTO. This defines the amount of time, in milliseconds, that the EDE subscriber waits to receive a TS_List pair before a TS_List timeout occurs.

AiUInt32 ul_TimeOffsetTO

The Time Offset timeout. This parameter is only valid when ul_ControlMode = FDX_SUB_AUTO. This defines the amount of time, in milliseconds, that the EDE subscriber waits to receive a Time Offset Table pair before a TimeOffset timeout occurs and the subscriber sets all Offsets to “Offset Unknown”.

AiUInt32 ul_ManualControl

Reserved

*TY_FDX_EDE_SUB_CNTL_EX_IN *px_EDESubCntlEx1In*

Pointer to the Extended subscriber control input structure.

```
typedef struct {
    AiUInt32    ul_TOTIterationRate;
    AiUInt32    ul_ReqNumValidWindow;
    AiUInt32    ul_TREQMaxAge;
    AiUInt32    ul_TOTMaxAge;
    AiUInt32    ul_MyOS1Max;
    AiUInt32    ul_OtherOS1Max;
    AiUInt32    ul_RxClockError;
     AiUInt32    ul_EDEOffsetUnkHigh;
     AiUInt32    ul_EDEOffsetUnkLow;
}TY_FDX_EDE_SUB_CNTL_EX_IN;
```

AiUInt32 ul_TOTIterationRate

Defines the rate, in milliseconds, at which the EDE subscriber checks the Time Offset Table ports of the Time Manager connections for a valid time offset table.

When the EDE Subscriber is created, a default value of 1 second is used to initialize this parameter.

(See [5]: EIO-7440)

AiUInt32 ul_ReqNumValidWindow

Defines the Time Request number valid window to be used to validate the request number of the Time Offset Tables. The following must be true for the Time Offset Table message to be accepted:

Last TREQ Req. Num. >= Req. Num of TOT >= Last TREQ Req. Num. – ul_ReqNumValidWindow.

When the EDE Subscriber is created, a default value of 2 is used to initialize this parameter.

(See [5]: EIO-5897)

AiUInt32 ul_TREQMaxAge

Defines maximum amount of time, in microseconds, since the last received Time Request message on a Time Manager connection in order for the Time Offset Table from the connection to be used by the Subscriber.

When the EDE Subscriber is created, a default value of 2.5 Sec is used to initialize this parameter.

(See [5]: EIO-7467)

AiUInt32 ul_TOTMaxAge

Defines the maximum amount of time, in microseconds, since the reception of the Time Offset message on the Time Manager connection. This is the maximum amount of time allowed since the last receipt of a Time Offset Table message in order for the message to be used.

When the EDE Subscriber is created, a default value of 2.5 Sec is used to initialize this parameter.

(See [5]: EIO-7468)

AiUInt32 ul_MyOS1Max

Defines the valid value range, in microseconds, for the Offset time (Rx TS – Tx TS) of the Time Offset Table message containing ‘myROS’.

For the Time Offset table message (containing ‘my ROS’) to be accepted, the following must be true:

myROS – ul_MyOS1Max <= Rx TS – Tx TS <= myROS + ul_MyOS1Max where Rx TS and Tx TS are the Rx and Tx EDE timestamps of the Time Offset Table message.

When the EDE Subscriber is created, a default value of 100 mSec is used to initialize this parameter.

(See [5]: EIO-7037)

AiUInt32 ul_OtherOS1Max

Defines the maximum time difference, in microseconds, between the EDE Tx Timestamp of the Time Offset message containing ‘myROS’ and the EDE Tx Timestamp of the Time Offset message NOT containing ‘myROS’.

When the EDE Subscriber is created, a default value of 1500 mSec is used to initialize this parameter.

(See [5]: EIO-7037)

AiUInt32 ul_RxClockError

The ‘Rx Clock Error’, in microseconds, used during the calculation of the EDE Offsets for each remote subscriber in the Time Offset Tables. This defines the difference between the Rx TS of received messages and the actual value of the ES EDE clock at the time of the receipt of the message.

When the EDE Subscriber is created, a default value of 0 is used to initialize this parameter.

(See [5]: EIO-5899)

AiUInt32 ul_EDEOffsetUnkHigh

Bits 0-15 of this parameter define the high order2 bytes of the EDE Offset Unknown value to be used by the EDE subscriber

When the EDE Subscriber is created, a default value of 0x7FFFFFFFFFFF is used to initialize the offset unknown parameter for the EDE subscriber.

AiUInt32 ul_EDEOffsetUnkLow

Defines the low order 4 bytes of the EDE Offset Unknown value to be used by the EDE subscriber.

When the EDE Subscriber is created, a default value of 0x7FFFFFFFFFFF is used to initialize the offset unknown parameter for the EDE subscriber.

*TY_FDX_EDE_SUB_CNTL_EX2_IN *px_EDESubCntlEx1In*

Pointer to the Extended subscriber control input structure.

```
typedef struct {
    AiUInt32 ul_ChA_LinkQuality;
    AiUInt32 ul_ChB_LinkQuality;
    AiUInt32 ul_ChA_EDEAgeFailures;
    AiUInt32 ul_ChB_EDEAgeFailures;
}TY_FDX_EDE_SUB_CNTL_EX2_IN;
```

AiUInt32 ul_ChA_LinkQuality

Link Quality of Channel A. Only one Byte of this value is used.

AiUInt32 ul__ChB_LinkQuality

Link Quality of Channel B. Only one Byte of this value is used.

AiUInt32 ul_ ChA_EDEAgeFailures

EDE Age Failures of Channel A. Only two Byte of this value are used.

AiUInt32 ul_ ChB_EDEAgeFailures

EDE Age Failures of Channel B. Only two Byte of this value are used.

Output:

None.

Return Value

Returns FDX_OK on success or a negative error code on error.

Error Codes: FDX_ERR

3.2.7 FdxCmdEDESubTMDef

Prototype:

AiReturn FdxCmdEDESubTMDef (AiUInt32 ul_Handle,
const AiUInt32 ul_ EDESubscriberHandle,
const TY_FDX_EDE_SUB_TM_DEF_IN* px_EDESubTMDefIn);

Driver Command:

FDX_EDE_SUB_TM_DEF (0x00008044)

Purpose:

This function is used to define a Subscriber connection to an EDE Time Manager.

Input:

AiUInt32 ul_EDESubscriberHandle

The handle to the EDE Subscriber.

TY_FDX_EDE_SUB_TM_DEF_IN *px_EDESubTMDefIn

Pointer to the EDE Subscriber TM define input structure.

```
typedef struct {
   AiUInt32          ul_TMIndex;
   AiUInt32          ul_TMType;
   AiUInt32          ul_TRESPA;
   AiUInt32          ul_TRESPB;
   AiUInt32          ul_TREQA;
   AiUInt32          ul_TREQB;
   AiUInt32          ul_TOT1A;
   AiUInt32          ul_TOT2B;
   AiUInt32          ul_TOT2A;
   AiUInt32          ul_TOT2B;
   AiUInt32          ul_TSLISTA;
   AiUInt32          ul_TSLISTB;
} TY_FDX_EDE_SUB_TM_DEF_IN;
```

AiUInt32 ul_TMIndex

Index of the Time manger within the EDE Subscriber. Possible values are 0 – 3;

AiUInt32 ul_TMType

Defines the type of Time Manager

Constant	Description
FDX_EDE_TM_INTERNAL	An internally simulated Time Manager
FDX_EDE_TM_EXTERNAL	A real external Time Manager

The parameters ul_TRESPA – ul_TSLISTB are only used in the case the ul_TMType = FDX_EDE_TM_EXTERNAL. They are not used for FDX_EDE_TM_INTERNAL.

AiUInt32 ul_TRESPA

UDP port handle to the port used by the subscriber to transmit Time Response A messages to the TM.

AiUInt32 ul_TRESPB

UDP port handle to the port used by the subscriber to transmit Time Response ABmessages to the Time Manager.

AiUInt32 ul_TREQA

UDP port handle to the port used by the subscriber to receive network A Time Request messages from the TM.

AiUInt32 ul_TREQB

UDP port handle to the port used by the subscriber to receive network B Time Request messages from the TM.

AiUInt32 ul_TOT1A

UDP port handle to the port used by the subscriber to receive network A Time Offset Table 1 messages from the TM.

AiUInt32 ul_TOT2B

UDP port handle to the port used by the subscriber to receive network B Time Offset Table 1 messages from the TM.

AiUInt32 ul_TOT2A

UDP port handle to the port used by the subscriber to receive network A Time Offset Table 2 messages from the TM.

AiUInt32 ul_TOT2B

UDP port handle to the port used by the subscriber to receive network B Time Offset Table 2 messages from the TM.

AiUInt32 ul_TSLISTA

UDP port handle to the port used by the subscriber to receive network A Time Stamp List messages from the TM.

AiUInt32 ul_TSLISTB

UDP port handle to the port used by the subscriber to receive network B Time Stamp List messages from the TM.

3.2.8 FdxCmdEDESubDestroy

Prototype:

```
AiReturn FdxCmdEDESubscriberDestroy(AiUInt32 ul_PortHandle,
                                    const AiUInt32 ul_EDESubscriberHandle);
```

Driver Command:

FDX_EDE_SUB_DESTROY (0x00008042)

Purpose:

This function is used to remove and release all resources associated with an EDE Subscriber.

Input:

AiUInt32 ul_EDESubscriberHandle

Handle to the EDE subscriber to be destroyed.

NOTE: An input value of 0 will instruct the library to Destroy all previously created EDE subscribers on the port.

Output:

None

Return Value

Returns FDX_OK on success or a negative error code on error.

Error Codes: FDX_ERR

3.3 EDE Transmit Functions

Table 3.3-1: EDE Transmit Functions

Function	*Description*
Generic or Replay Transmitter Functions	
FdxCmdTxQueueWriteEx	Writes AFDX Frames to the Queue (supports EDE Frames)
UDP Port-Oriented Transmitter Functions	
FdxCmdTxEDECreatePort	Creates a new EDE enabled AFDX Communication port
FdxCmdTxEDEWrite	Writes a message to an EDE enabled AFDX Comm. port
FdxCmdTxUDPWrite	Writes a message to either and EDE on non-EDE AFDX Comm. port

3.3.1 Generic Transmitter Functions

3.3.1.1 FdxCmdTxQueueWriteEx

Prototype:

```
AiReturn FdxCmdTxQueueWriteEx (        AiUInt32 ul_Handle,
                                    AiUInt32 ul_HeaderType,
                                    AiUInt32 ul_EntryCount,
                                    AiUInt32 ul_WriteBytes,
                                    const void *pv_WriteBuffer);
```

Driver Command:

FDX_TX_QUEUE_WRITE_EX (0x0000806D)

Purpose:

This function is used to write Entries to a Transmit Queue from a provided buffer. For this write function the number of Entries and the number of bytes to write needs to be specified. The entries will always be queued at the end of the transmit queue.

Input:

AiUInt32 ul_HeaderType

This parameter defines, the type of the frame header structure.

Value:	*Description:*
FDX_TX_FRAME_HEADER_GENERIC	Standard Generic Tx Frame, only applicable for Generic Transmit mode. Layout of frame header follows the TY_FDX_TX_FRAME_HEADER structure
FDX_TX_FRAME_HEADER_REPLAY	Replay Tx Frame, only applicable in Replay Transmit mode. Layout of frame header follows the TY_FDX_FRAME_BUFFER_HEADER Structure, described at the **FdxCmdMonQueueRead** command.

AiUInt32 ul_EntryCount

Number of Entries to write. Not applicable for Replay Transmit mode.

AiUInt32 ul_WriteBytes

Number of bytes that shall be written to the queue.

*void *pv_WriteBuffer*

Pointer to the data buffer providing the Entries to write. The size of this buffer should correspond to *ul_WriteBytes*.

One Entry specifies one Frame + Header Information. This means one complete MAC frame plus a fixed sized Header. The Header contains information about the manner in which the frame should be sent on the network.

Layout of one Queue Entry:

	Entry Layout
Fixed Header	Fixed Frame Header Layout dependent on *ul_HeaderType* and *uc_FrameType* parameter (see following description)
AFDX Frame	AFDX- FRAME data to transmit (dependent on the Payload Buffer and Payload Generation mode, see description below) (802.3 defines: 64 to 1518 bytes)

For Header Type ***FDX_TX_FRAME_HEADER_REPLAY*** refer to the frame buffer layout described in function **FdxCmdMonQueueRead**.

Note: The replay mode does not reproduce any recorded physical error conditions, but is tolerant to protocol errors as well as size violations. A packet will be discarded by the firmware if any of the following error conditions is detected: PHY, PRE, TRI, CRC, IFG, SFD. The following error types are tolerated and will be replayed: IPE, MAE, LNG (up to frame length of 2000 bytes), SHR (from frame length of 40 bytes), VLS, SNE, TNS.

For Header Type ***FDX_TX_FRAME_HEADER_GENERIC*** see following description.

TY_FDX_TX_FRAME_HEADER_EX x_TxFrameHeaderEx

```
typedef struct {
   AiUInt8   uc_FrameType;
   TY_FDX_TX_FRAME_ATTRIB_EX x_FrameAttribEx;
   TY_FDX_TX_INSTR_ATTRIB x_InstrAttrib;
} TY_FDX_TX_FRAME_HEADER_EX;
```

Note: ***The FdxInitTxFrameHeaderEx function supports a default initialization of this structure (see this function in the chapter 'Target Independent Administration Functions'***

AiUInt8 uc_FrameType

The Type of the frame:

Value:	*Description:*
FDX_TX_FRAME_STD	Standard Generic Tx Frame
FDX_TX_FRAME_INSTR	Instruction Type

TY_FDX_TX_FRAME_ATTRIB_EX
x_FrameAttribEx

This structure describes the Frame Attributes in case of ***FDX_TX_FRAME_STD uc_FrameType.***

```
typedef struct {
   AiUInt16  uw_FrameSize;
   AiUInt32  ul_InterFrameGap;
   AiUInt32  ul_PacketGroupWaitTime;
   AiUInt8   uc_PayloadBufferMode;
   AiUInt8   uc_PayloadGenerationMode;
   AiUInt32  ul_BufferQueueHandle;
   AiUInt8   uc_ExternalStrobe;
   AiUInt8   uc_PreambleCount;
   AiUInt32  ul_Skew;
   AiUInt8    uc_NetSelect;
   AiUInt8    uc_FrameStartMode;
   AiUInt32  ul_PhysErrorInjection;
   AiUInt16   uw_SequenceNumberInit;
   AiUInt16   uw_SequenceNumberOffset;
   AiUInt32   ul_EDEFlags;
   AiUInt32  ul_EDESourceId;
   AiUInt32  ul_EDEMessageSize;
   AiUInt32   ul_EDETsOffsetHigh;
   AiUInt32  ul_EDETsOffsetLow;
   AiUInt32  ul_EDESNOffset;
} TY_FDX_TX_FRAME_ATTRIB_EX;
```

AiUInt16 uw_FrameSize;

Total size of the associated frame in Bytes (incl. CRC). Short and Long Frame Error Conditions are possible by setting the corresponding values. AFDX compliant values are 64…1518. For Frame length less than 60 no proper frame transmission is guaranteed.

AiUInt32 ul_InterFrameGap

This value defines the interframe gap between the preceding frame and the current frame with a resolution of 40ns, measured from the end of the last bit fo the preceding frame to the first preamble bit of the actual frame.

To implement a physical gap between the frames, a minimum interframe gap of 120 ns (value = 3) shall be initialized. The maximum provided interframe gap will be up to approx. 655us (14 Bits are used for encoding). If the Packet group Wait Time is used, this field shall be initialized with zero. This Gap is only used if ***uc_FrameStartMode*** is set to FDX_TX_START_FRAME_IFG.

See also the notes for ul_Skew parameter in redundant mode.

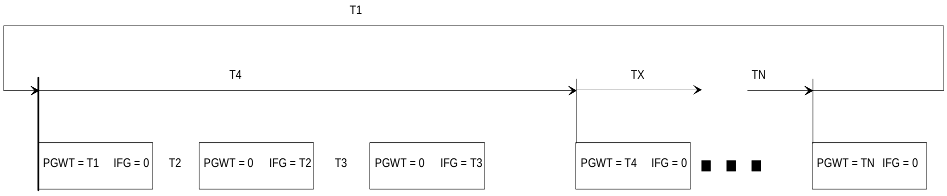
AiUInt32 ul_PacketGroupWaitTime

The Packet Group Wait Time (PGWT) provides the capability to implement a sequencing control for one or a group of frames. Each time, if a frame is processed where the PGWT value is not zero, the BIU- Processor handles the corresponding timing. The PGWT value (20bit) specifies the time from the transmission start point of the last frame where the PGWT value is processed to the start point of the current frame with a resolution of 1us.

At the processing of the first frame after the operation is enabled or after the execution of a 'Wait for Trigger' instruction, the PGWT value is ignored and the frame is transmitted immediately.

This Gap is only used if ***uc_FrameStartMode*** is set to FDX_TX_START_FRAME_PGWT.

AiUInt8 uc_PayloadBufferMode



Reserved.

AiUInt8 uc_PayloadGenerationMode

The Payload Generation Mode (PGM) defines, which contents of the frame data are automatically generated by the MAC- Hardware out of the static Tx data registers (see Command ***FdxCmdTxStaticRegsCtrl***). The payload generation mode is based on the UDP- protocol within the frame. Therefore, if one of the data reduction modes is selected, the transmit frame must include an UDP- protocol type.

NOTE: Only payload generation mode FDX_TX_FRAME_PGM_USER is valid in the case that the frame is EDE enabled (i.e. ul_EDEFlags = FDX_EDE_TX_FRAME_ENA.

Value:	***Description:***
FDX_TX_FRAME_PGM_USER	Default, no payload generation. All AFDX- frame data for transmission will be provided from this frame entry or partly from the payload buffer queue. In this mode, all MAC- frame data, except the FCS- field, has to be fed into the MAC by the BIU- Processor. Complete frame data must be provided for this frame.
FDX_TX_FRAME_PGM_IP_PART	The MAC- Destination Address Bytes 2..5, the MAC- Source Address Bytes 3…5, the MAC- Length Type Field Byte 0..1, the IP- Version field, the IP- Header Length field, the IP- protocol field and the UDP- checksum field as well as the hole UDP- payload will be generated by the MAC- Hardware by using the Static Transmit Registers. Only the MAC header and IP Header and UDP Header needs to be provided as frame data for this frame.
FDX_TX_FRAME_PGM_IP_FULL	The MAC- Destination Address Bytes 2…5, the MAC- Source Address Bytes 3…5, the MAC- Length Type Field Byte 0...1, the IP- Version field, the hole IP- Header (20 bytes) and the hole UDP- Header as well as the hole UDP- payload will be generated by the MAC- Hardware by using the Static Transmit Registers. Only the MAC header needs to be provided as frame data for this frame.
FDX_TX_FRAME_PGM_IP_PART_TT	Same as FDX_TX_FRAME_PGM_IP_PART, plus UDP- payload data will be filled with the start Timetag, which is repeated every eight bytes.
FDX_TX_FRAME_PGM_IP_FULL_TT	Same as FDX_TX_FRAME_PGM_IP_FULL, plus UDP- payload data will be filled with the start Timetag, which is repeated every eight bytes.

Note: ***This Static Transmit Registers must be setup properly if any frame uses a Payload Generation Mode, different from FDX_TX_FRAME_PGM_USER ! Otherwise the frame data may be invalid.***

The following Table shows the necessary size of one Queue Entry dependent on the Payload Buffer and Payload Generation Modes.

Payload Generation Mode	*Size of Queue Entry*
FDX_TX_FRAME_PGM_USER	sizeof (TY_FDX_TX_FRAME_HEADER) + uw_FrameSize
FDX_TX_FRAME_PGM_IP_PART FDX_TX_FRAME_PGM_IP_PART_TT	sizeof (TY_FDX_TX_FRAME_HEADER) + MAC-Header (14 Bytes) + IP-Header (20 Bytes) + UDP-Header (8 Bytes)
FDX_TX_FRAME_PGM_IP_FULL FDX_TX_FRAME_PGM_IP_FULL_TT	sizeof (TY_FDX_TX_FRAME_HEADER) + MAC-Header (14 Bytes)

Note: ***For Timetag-Payload-Generation modes the Timetag Format in Payload is***
started at Byte 44 (2 bytes after UDP-Checksum)
BYTE 44...47 – Timetag HI

Bit 5... 0	***:***	***Seconds of minute***
Bit 11... 6	***:***	***Minutes of hour***
Bit 16...12	***:***	***Hours of day***
Bit 25...17	***:***	***Days of year***
Bit 31...26	***:***	***reserved (0)***

BYTE 48...51 – Timetag LO

Bit 19... 0	***:***	***Microseconds of second***
Bit 25...20	***:***	***Seconds of minute***
Bit 31...26	***:***	***Minutes of hour***

AiUInt32 ul_BufferQueueHandle

Reserved.

AiUInt8 uc_ExternalStrobe

Control assertion of Trigger Strobe if this frame is transmitted. See the ***FdxCmdTxTrgLineCtrl*** for further information about the Trigger Lines.

Value:	*Description:*
FDX_DIS	Disable Trigger Strobe
FDX_ENA	Assert external Trigger Strobe on transmission of this frame

AiUInt8 uc_PreambleCount

This value defines the number of preamble Bytes sent for this frame

Value:	*Description:*
FDX_TX_FRAME_PRE_DEF	Send default preamble count of 7 Bytes
All other values from n=1..15	Send n preamble Bytes

AiUInt32 ul_Skew

This parameter defines the transmission skew between the redundant frames on the AFDX- ports. The skew can be programmed with a resolution of 1us. Range is 0...65535. This parameter is only used if uc_NetSelect is FDX_TX_FRAME_DLY_A or FDX_TX_FRAME_DLY_B.

Note: ***This function is only provided in redundant port operation mode.***

Note: ***If the ul_Skew parameter is set and one redundant frame is delayed this time may be added to ul_InterFrameGap and may exceed maximum value of ul_InterFrameGap in the receiver.This means it can result in a higher Interframe Gap Time because the IFG counter for transmit is sterted synchroniously for both networks after both redundant frames are sent..***

AiUInt8 uc_NetSelect

This parameter defines the physical Interface_ID of the MAC, which shall send it's current frame delayed (with ***ul_Skew*** µs) to the alternate port.

Value:	***Description:***
FDX_TX_FRAME_DLY_A	Packet on Network A is delayed by the Skew value, related to Network B
FDX_TX_FRAME_DLY_B	Packet on Network B is delayed by the Skew value, related to Network A
FDX_TX_FRAME_BOTH	Packet transmitted on both Networks (Skew=0)
FDX_TX_FRAME_ONLY_A	Packet only transmitted on Network A
FDX_TX_FRAME_ONLY_B	Packet only transmitted on Network B

Note: ***This function is only provided in redundant port operation mode.***

AiUInt8 uc_FrameStartMode

This parameter defines the Frame Start mode for the transmission of the current frame.

Value:	***Description:***
FDX_TX_FRAME_START_IFG	Start transmission of this frame if Interframe GAP time has expired (see *ul_InterFrameGap* parameter)
FDX_TX_FRAME_START_PGWT	Start transmission of this frame if Packet Group Wait Time (PGWT) has expired (see *ul_PacketGroupWaitTIme* parameter)
FDX_TX_FRAME_START_TRG	Start transmission of this frame on external Trigger Strobe. This means, frame transmission is stopped with this frame, until the external Trigger Strobe is given to continue transmission with this frame.

AiUint32 ul_PhysErrorInjection

This parameter defines physical error injection types. The error injection information can be a combination of the following error types:

Value:	***Description:***
FDX_TX_FRAME_ERR_OFF	No Error Injection enabled
FDX_TX_FRAME_ERR_CRC	CRC Error transmitted with this frame
FDX_TX_FRAME_ERR_ALI	Wrong Byte alignment in transmit frame, which means that an odd number of nibbles will be transmitted. Therefore, this error will also cause a CRC error condition
FDX_TX_FRAME_ERR_PRE	Wrong Preamble Sequence transmitted. If this type is selected., the Encoder device substitutes the first nibble of the Start Frame Delimiter with the value ‘1000’ instead of ‘1001’
FDX_TX_FRAME_ERR_PHY	Physical Symbol Error. During Frame Transmission, the MAC-Encoder device asserts the Tx-Error signal, which forces the physical transceiver to transmit ‘HALT’ symbols.

AiUint16 uw_SequenceNumberInit

This parameter defines Initial Sequence Number for this frame. First frame transmission starts with this Sequence Number and adds then the Sequence Number Offset ***uc_SequenceNumberOffset*** , as described at the following parameter.

Value:	***Description:***
FDX_TX_FRAME_SEQ_INIT_AUTO	Sequence Number Init value is set by the Driver automatically.
0…255	Set Sequence Number Offset for this frame (e.g. Error Injection Purposes if invalid values used !)

AiUint16 uw_SequenceNumberOffset

This parameter defines the Sequence Number Offset. This field provides the Sequence Number offset, which is added to the Sequence Number after the Frame has been transmitted. The Sequence number will be incremented until the value of 255 and then it wraps around to 1. Thus, the user can initialize a transmission sequence, which implements more than one frame with the same VL. If the transmission sequence implements N packets with the same VL, this field shall be initialized with N to implement proper sequence numbering for each transmitted VL- frame.

Value:	***Description:***
FDX_TX_FRAME_SEQ_OFFS_AUTO	Sequence Number Offset is set by the Driver automatically.
0…255	Set Sequence Number Offset for this frame (e.g. Error Injection Purposes if invalid values used !)

AiUInt32 ul_EDEFlags

This parameter defines whether or not the frame is an EDE frame. If EDE is enabled, it can also be used to specify the EDE error injection and Time stamping modesfor the frame.

Possible values are FDX_TX_FRAME_EDE_DIS, or any combination of the other possible values listed below.

Value:	***Description:***
FDX_EDE_TX_FRAME_DIS	No EDE handling is used for this frame
FDX_EDE_TX_FRAME_ENA	EDE handling (Firmware timestamping and CRC calculations) are used for this frame
FDX_EDE_TX_FRAME_FIRST	This frame is the first fragment of a EDE message (Must be set for single frame messages)
FDX_EDE_TX_FRAME_LAST	This frame is the last fragment of a EDE message (Must be set for single frame messages)
FDX_EDE_TX_FRAME_ERR_TS	The board firmware is instructed not to insert an EDE TS for this frame
FDX_EDE_TX_FRAME_ERR_CRCX	An EDE CRC-X error is transmitted
FDX_EDE_TX_FRAME_ERR_CRCY	An EDE CRC-Y error is transmitted
FDX_EDE_TX_FRAME_ERR_NO_SN	Instructs the Firmware that an EDE Sequence number shall not be inserted into the frame. This is only valid for frames with the flag FDX_TX_FRAME_EDE_FIRST.

AiUInt32 ul_EDESourceId

The EDE Source ID used for calculating the EDE CRCs for the frame.

AiUInt32 ul_EDEMessageSize

The remaining EDE message size for the complete EDE message. This includes the EDE Sequence Number, Timestamp, payload, and both CRCs.

AiUInt32 ul_EDETsOffsetHigh

31…………………………………………………..16	***15……………………………………………..0***
Reserved (Coded 0)	EDE Time Stamp Offset (bit 32 – 47)

AiUInt32 ul_EDETsOffsetLow

31…………………………………………………………………………………………………….0

EDE Time Stamp Offset (bit 0 – 31)

The parameters ul_EDETsOffseHigh and ul_EDETsOffsetLow combine to form the 48-bit 2’s compliment offset that is added by the firmware to the 48-bit EDE counter before applying to the frame as the EDE transmit timestamp.

AiUInt32 ul_EDESNOffset

The EDE SN offset which is added to the EDE SN for the current EDE port (specified by ul_EDEPortIndex) after the frame is transmitted. Possible values are 0..7.

TY_FDX_TX_INSTR_ATTRIB x_TxInstrAttrib

This structure describes the Instruction Attributes in case of ***FDX_TX_FRAME_INSTR uc_FrameType.***

```
typedef struct {
   AiUInt8 uc_Code;
   AiUInt8 uc_Interrupt ;
} TY_FDX_TX_INSTR_ATTRIB;
```

AiUInt8 uc_Code

Following Instruction Codes are supported :

Value:	***Description:***
FDX_TX_FRAME_INSTR_NOP	No Operation
FDX_TX_FRAME_INSTR_STOP	Stop Transmission Transmission is stopped if BIU processor runs on this Instruction
FDX_TX_FRAME_INSTR_SYNC	Synchronize BIU Processor waits until Transmit Burst Buffer (between BIU and MAC) is empty

AiUInt8 uc_Interrupt

Enable/Disable Interrupt on execution of Instruction

Output:

None

Return Value

Returns FDX_OK on success or a negative error code on error.
Error Codes: FDX_ERR

3.3.1.2 **FdxCmdTxQueueWriteEde**

Prototype:

AiReturn FdxCmdTxQueueWriteEde (const AiUInt32 ul_Handle,
const AiUInt32 ul_HeaderType,
const AiUInt32 ul_EntryCount,
const AiUInt32 ul_WriteBytes,
const void *pv_WriteBuffer);

Driver Command:

FDX_TX_QUEUE_WRITE

Purpose:

This function is used to write Entries to a Transmit Queue from a provided buffer. For this write function the number of Entries and the number of bytes to write needs to be specified. The entries will always be queued at the end of the transmit queue.

Input:

AiUInt32 ul_HeaderType

This parameter defines, the type of the frame header structure.

Value:	*Description:*
FDX_TX_FRAME_HEADER_GENERIC	Standard Generic Tx Frame, only applicable for Generic Transmit mode. Layout of frame header follows the TY_FDX_TX_FRAME_HEADER structure
FDX_TX_FRAME_HEADER_REPLAY	Replay Tx Frame, only applicable in Replay Transmit mode. Layout of frame header follows the TY_FDX_FRAME_BUFFER_HEADER Structure, described at the **FdxCmdMonQueueRead** command.

AiUInt32 ul_EntryCount

Number of Entries to write. Not applicable for Replay Transmit mode.

At the moment only a count of 1 is suported

AiUInt32 ul_WriteBytes

Number of bytes that shall be written to the queue.

*void *pv_WriteBuffer*

Pointer to the data buffer providing the Entries to write. The size of this buffer should correspond to *ul_WriteBytes*.

One Entry specifies one Frame + Header Information. This means one complete MAC frame plus a fixed sized Header. The Header contains information about the manner in which the frame should be sent on the network.

Layout of one Queue Entry:

	Entry Layout
Fixed Header	Fixed Frame Header Layout dependent on *ul_HeaderType* and *uc_FrameType* parameter (see following description)
AFDX Frame	AFDX- FRAME data to transmit (dependent on the Payload Buffer and Payload Generation mode, see description below) (802.3 defines: 64 to 1518 bytes)

For Header Type ***FDX_TX_FRAME_HEADER_REPLAY*** refer to the frame buffer layout described in function **FdxCmdMonQueueRead**.

Note: The replay mode does not reproduce any recorded physical error conditions, but is tolerant to protocol errors as well as size violations. A packet will be discarded by the firmware if any of the following error conditions is detected: PHY, PRE, TRI, CRC, IFG, SFD. The following error types are tolerated and will be replayed: IPE, MAE, LNG (up to frame length of 2000 bytes), SHR (from frame length of 40 bytes), VLS, SNE, TNS.

For Header Type ***FDX_TX_FRAME_HEADER_GENERIC*** see following description.

TY_FDX_TX_FRAME_HEADER x_TxFrameHeader

```
typedef struct {
   AiUInt8    uc_FrameType;
   TY_FDX_TX_FRAME_ATTRIB x_FrameAttrib;
   TY_FDX_TX_INSTR_ATTRIB x_InstrAttrib;
   TY_FDX_TX_FRAME_ATTRIB_EDE x_FrameAttribEde;
} TY_FDX_TX_FRAME_HEADER_EDE;
```

Note: The FdxInitTxFrameHeader function supports a default initialization of this structure (see this function in the chapter 'Target Independent Administration Functions'

AiUInt8 uc_FrameType

The Type of the frame:

Value:	*Description:*
FDX_TX_FRAME_STD	Standard Generic Tx Frame
FDX_TX_FRAME_INSTR	Instruction Type

TY_FDX_TX_FRAME_ATTRIB
x_FrameAttrib

This structure describes the Frame Attributes in case of ***FDX_TX_FRAME_STD uc_FrameType.***

```
typedef struct {
   AiUInt16 uw_FrameSize;
   AiUInt32 ul_InterFrameGap;
   AiUInt32  ul_PacketGroupWaitTime;
   AiUInt8   uc_PayloadBufferMode;
   AiUInt8   uc_PayloadGenerationMode;
   AiUInt32  ul_BufferQueueHandle;
   AiUInt8   uc_ExternalStrobe;
   AiUInt8   uc_PreambleCount;
   AiUInt32  ul_Skew;
   AiUInt8   uc_NetSelect;
   AiUInt8   uc_FrameStartMode;
   AiUInt32  ul_PhysErrorInjection;
   AiUInt16  uw_SequenceNumberInit;
   AiUInt16  uw_SequenceNumberOffset;
   AiUInt8   uc_TxIntEnable
   AiUInt32  ul_IntIdent
} TY_FDX_TX_FRAME_ATTRIB;
```

AiUInt16 uw_FrameSize;

Total size of the associated frame in Bytes (incl. CRC). Short and Long Frame Error Conditions are possible by setting the corresponding values. AFDX compliant values are 64…1518. For Frame length less than 60 no proper frame transmission is guaranteed.

AiUInt32 ul_InterFrameGap

This value defines the interframe gap between the preceding frame and the current frame with a resolution of 40ns, measured from the end of the last bit fo the preceding frame to the first preamble bit of the actual frame.

To implement a physical gap between the frames, a minimum interframe gap of 120 ns (value = 3) shall be initialized. The maximum provided interframe gap will be up to approx. 655us (14 Bits are used for encoding). If the Packet group Wait Time is used, this field shall be initialized with zero. This Gap is only used if ***uc_FrameStartMode*** is set to FDX_TX_START_FRAME_IFG.

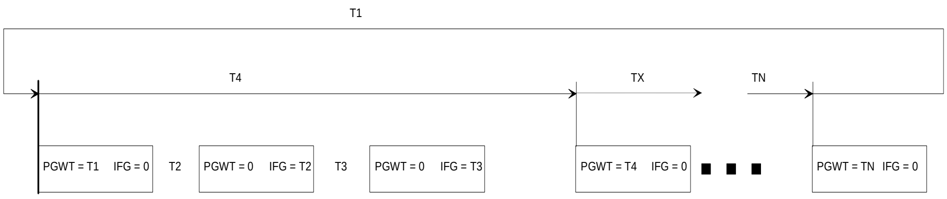
See also the notes for ul_Skew parameter in redundant mode.

AiUInt32 ul_PacketGroupWaitTime

The Packet Group Wait Time (PGWT) provides the capability to implement a sequencing control for one or a group of frames. Each time, if a frame is processed where the PGWT value is not zero, the BIU- Processor handles the corresponding timing. The PGWT value (20bit) specifies the time from the transmission start point of the last frame where the PGWT value is processed to the start point of the current frame with a resolution of 1us.

At the processing of the first frame after the operation is enabled or after the execution of a 'Wait for Trigger' instruction, the PGWT value is ignored and the frame is transmitted immediately.

This Gap is only used if ***uc_FrameStartMode*** is set to FDX_TX_START_FRAME_PGWT.



AiUInt8 uc_PayloadBufferMode

The Payload Buffer Modes (PBMs) can be used to implement dynamic payload for either MAC or UDP- frame for this transmit frame, by using the separate buffer queue. The separate buffer queue will only be used, if the Buffer Queue Header Pointer is appropriate initialized (not zero) and the Payload Generation is disabled. The Buffer Queue itself implements a queue with one or multiple payload buffers, which each provides it's individual payload for the transmit frame. Due to the Buffer Queue function capabilities, different payload buffers can be used for the transmission of the frame. If a Payload Buffer Mode different from **FDX_TX_FRAME_PBM_STD** is used, the data for the frame must be provided in a separate Buffer, allocated via the ***FdxCmdTxBufferQueueAlloc*** function, dependent on the Payload Buffer Mode.

Value:	***Description:***
FDX_TX_FRAME_PBM_STD	Complete frame data is taken from the entry *ul_BufferQueueHandle* must be set to NULL
FDX_TX_FRAME_PBM_MAC Note: This mode cannot be selected if a Payload Generation Mode different from FDX_TX_FRAME_PGM_USER has been selected for this frame.	MAC- Payload is provided in the separate Buffer Queue. That means, the BIU- Processor fetches the Frame Header Words and the first 16 bytes of the frame data out from this entry and switches then to the separate buffer queue. Thus, the complete MAC- Header and the two static bytes of the IP- Header are used from this entry and the rest of the frame payload is used from the separate buffer queue. *ul_BufferQueueHandle* must contain a valid Buffer Queue handle, previously allocated via the function *FdxCmdTxBufferQueueAlloc*.
FDX_TX_FRAME_PBM_UDP Note: This mode cannot be selected if a Payload Generation Mode different from FDX_TX_FRAME_PGM_USER has been selected for this frame.	UDP- Payload is provided in the separate Buffer Queue. That means, the BIU- Processor fetches the Frame Header Words and the first 40 bytes of the frame data out from this entry and switches then to the separate buffer queue. Thus, the complete MAC- Header, the IP- Header and 6 bytes of the UDP- Header are used from this entry and the remainder of the frame payload is used from the separate buffer queue. That means, the 2 bytes of the UDP-Checksum (always zero) and the UDP- payload resides in the separate buffer. *ul_BufferQueueHandle* must contain a valid Buffer Queue handle, previously allocated via the function *FdxCmdTxBufferQueueAlloc*.
FDX_TX_FRAME_PBM_FULL Note: This mode cannot be selected if a Payload Generation Mode different from FDX_TX_FRAME_PGM_USER has been selected for this frame.	The full MAC-Frame is provided in the separate Buffer Queue. That means, the BIU- Processor fetches the Frame Header Words out from this entry and switches then to the separate buffer queue. *ul_BufferQueueHandle* must contain a valid Buffer Queue handle, previously allocated via the function *FdxCmdTxBufferQueueAlloc*.

AiUInt8 uc_PayloadGenerationMode

The Payload Generation Mode (PGM) defines, which contents of the frame data are automatically generated by the MAC- Hardware out of the static Tx data registers (see Command ***FdxCmdTxStaticRegsControl***). The payload generation mode is based on the UDP- protocol within the frame. Therefore, if one of the data reduction modes is selected, the transmit frame must include an UDP- protocol type.

Value:	***Description:***
FDX_TX_FRAME_PGM_USER	Default, no payload generation. All AFDX- frame data for transmission will be provided from this frame entry or partly from the payload buffer queue. In this mode, all MAC- frame data, except the FCS- field, has to be fed into the MAC by the BIU- Processor. Complete frame data must be provided for this frame.
FDX_TX_FRAME_PGM_IP_PART	The MAC- Destination Address Bytes 2..5, the MAC- Source Address Bytes 3…5, the MAC- Length Type Field Byte 0..1, the IP- Version field, the IP- Header Length field, the IP- protocol field and the UDP- checksum field as well as the hole UDP- payload will be generated by the MAC- Hardware by using the Static Transmit Registers. Only the MAC header and IP Header and UDP Header needs to be provided as frame data for this frame.
FDX_TX_FRAME_PGM_IP_FULL	The MAC- Destination Address Bytes 2…5, the MAC- Source Address Bytes 3…5, the MAC- Length Type Field Byte 0...1, the IP- Version field, the hole IP- Header (20 bytes) and the hole UDP- Header as well as the hole UDP- payload will be generated by the MAC- Hardware by using the Static Transmit Registers. Only the MAC header needs to be provided as frame data for this frame.
FDX_TX_FRAME_PGM_IP_PART_TT	Same as FDX_TX_FRAME_PGM_IP_PART, plus UDP- payload data will be filled with the start Timetag, which is repeated every eight bytes.
FDX_TX_FRAME_PGM_IP_FULL_TT	Same as FDX_TX_FRAME_PGM_IP_FULL, plus UDP- payload data will be filled with the start Timetag, which is repeated every eight bytes.

Note: ***This Static Transmit Registers must be setup properly if any frame uses a Payload Generation Mode, different from FDX_TX_FRAME_PGM_USER ! Otherwise the frame data may be invalid.***

The following Table shows the necessary size of one Queue Entry dependent on the Payload Buffer and Payload Generation Modes.

Payload Generation Mode	***Size of Queue Entry***
FDX_TX_FRAME_PGM_USER	sizeof (TY_FDX_TX_FRAME_HEADER) + uw_FrameSize
FDX_TX_FRAME_PGM_IP_PART FDX_TX_FRAME_PGM_IP_PART_TT	sizeof (TY_FDX_TX_FRAME_HEADER) + MAC-Header (14 Bytes) + IP-Header (20 Bytes) + UDP-Header (8 Bytes)
FDX_TX_FRAME_PGM_IP_FULL FDX_TX_FRAME_PGM_IP_FULL_TT	sizeof (TY_FDX_TX_FRAME_HEADER) + MAC-Header (14 Bytes)

The following Table shows the necessary size of one Queue Entry dependent on the Payload Buffer and Payload Generation Modes.

Payload Buffer Mode	*Payload Generation Mode*	*Size of Queue Entry*
FDX_TX_FRAME_PBM_STD	FDX_TX_FRAME_PGM_USER	sizeof (TY_FDX_TX_FRAME_HEADER) + uw_FrameSize
FDX_TX_FRAME_PBM_STD	FDX_TX_FRAME_PGM_IP_PART FDX_TX_FRAME_PGM_IP_PART_TT	sizeof (TY_FDX_TX_FRAME_HEADER) + MAC-Header (14 Bytes) + IP-Header (20 Bytes) + UDP-Header (8 Bytes)
FDX_TX_FRAME_PBM_STD	FDX_TX_FRAME_PGM_IP_FULL FDX_TX_FRAME_PGM_IP_FULL_TT	sizeof (TY_FDX_TX_FRAME_HEADER) + MAC-Header (14 Bytes)
FDX_TX_FRAME_PBM_FULL	FDX_TX_FRAME_PGM_USER	sizeof (TY_FDX_TX_FRAME_HEADER) Note: The full MAC Frame must be provided in separate Buffer.
FDX_TX_FRAME_PBM_MAC	FDX_TX_FRAME_PGM_USER	sizeof (TY_FDX_TX_FRAME_HEADER) + MAC-Header (14 Bytes) + IP-Version (1Byte) + IP-Type of Service (1Byte) Note: Remaining Data must be provided in separate Buffer.
FDX_TX_FRAME_PBM_UDP	FDX_TX_FRAME_PGM_USER	sizeof (TY_FDX_TX_FRAME_HEADER) + MAC-Header (14 Bytes) + IP-Header (20 Bytes) + UDP Source Port (2Bytes) + UDP Destination Port (2Bytes) + UDP Length (2Bytes) Note: Remaining Data must be provided in separate Buffer.

Note: ***For Timetag-Payload-Generation modes the Timetag Format in Payload is***
started at Byte 44 (2 bytes after UDP-Checksum)
BYTE 44...47 – Timetag HI
Bit 5... 0 : Seconds of minute
Bit 11... 6 : Minutes of hour
Bit 16...12 : Hours of day
Bit 25...17 : Days of year
Bit 31...26 : reserved (0)
BYTE 48...51 – Timetag LO
Bit 19... 0 : Microseconds of second
Bit 25...20 : Seconds of minute
Bit 31...26 : Minutes of hour

AiUInt32 ul_BufferQueueHandle

In Payload Buffer Mode FDX_TX_FRAME_PBM_FULL, FDX_TX_FRAME_PBM_MAC or FDX_TX_FRAME_PBM_UDP is used for this frame, a valid Buffer Queue Handle must be set. This Buffer Handle is obtained via the function ***FdxCmdTxBufferQueueAlloc***. If Payload Buffer Mode is not used, it should be initialized with NULL. Dependent on the Payload Buffer Mode, the allocated Buffer must contain the corresponding data beginning with MAC payload or UDP payload.

Using Payload Buffer Mode FDX_TX_FRAME_PBM_UDP or FDX_TX_FRAME_PBM_MAC would allow the user to change data associated with this frame during run-time by the ***FdxCmdTxBufferQueueWrite*** and ***FdxCmdTxBufferQueueCtrl*** functions.

AiUInt8 uc_ExternalStrobe

Control assertion of Trigger Strobe if this frame is transmitted. See the ***FdxCmdTxTrgLineCtrl*** for further information about the Trigger Lines.

Value:	***Description:***
FDX_DIS	Disable Trigger Strobe
FDX_ENA	Assert external Trigger Strobe on transmission of this frame

AiUInt8 uc_PreambleCount

This value defines the number of preamble Bytes sent for this frame

Value:	***Description:***
FDX_TX_FRAME_PRE_DEF	Send default preamble count of 7 Bytes
All other values from n=1..15	Send n preamble Bytes

AiUInt32 ul_Skew

This parameter defines the transmission skew between the redundant frames on the AFDX- ports. The skew can be programmed with a resolution of 1us. Range is 0...65535. This parameter is only used if uc_NetSelect is FDX_TX_FRAME_DLY_A or FDX_TX_FRAME_DLY_B.

Note: ***This function is only provided in redundant port operation mode.***

Note: ***If the ul_Skew parameter is set and one redundant frame is delayed this time may be added to ul_InterFrameGap and may exceed maximum value of ul_InterFrameGap in the receiver.This means it can result in a higher Interframe Gap Time because the IFG counter for transmit is sterted synchroniously for both networks after both redundant frames are sent..***

AiUInt8 uc_NetSelect

This parameter defines the physical Interface_ID of the MAC, which shall send it's current frame delayed (with ***ul_Skew*** µs) to the alternate port.

Value:	***Description:***
FDX_TX_FRAME_DLY_A	Packet on Network A is delayed by the Skew value, related to Network B
FDX_TX_FRAME_DLY_B	Packet on Network B is delayed by the Skew value, related to Network A
FDX_TX_FRAME_BOTH	Packet transmitted on both Networks (Skew=0)
FDX_TX_FRAME_ONLY_A	Packet only transmitted on Network A
FDX_TX_FRAME_ONLY_B	Packet only transmitted on Network B

Note: ***This function is only provided in redundant port operation mode.***

AiUInt8 uc_FrameStartMode

This parameter defines the Frame Start mode for the transmission of the current frame.

Value:	***Description:***
FDX_TX_FRAME_START_IFG	Start transmission of this frame if Interframe GAP time has expired (see *ul_InterFrameGap* parameter)
FDX_TX_FRAME_START_PGWT	Start transmission of this frame if Packet Group Wait Time (PGWT) has expired (see *ul_PacketGroupWaitTIme* parameter)
FDX_TX_FRAME_START_TRG	Start transmission of this frame on external Trigger Strobe. This means, frame transmission is stopped with this frame, until the external Trigger Strobe is given to continue transmission with this frame.

AiUint32 ul_PhysErrorInjection

This parameter defines physical error injection types. The error injection information can be a combination of the following error types:

Value:	***Description:***
FDX_TX_FRAME_ERR_OFF	No Error Injection enabled
FDX_TX_FRAME_ERR_CRC	CRC Error transmitted with this frame
FDX_TX_FRAME_ERR_ALI	Wrong Byte alignment in transmit frame, which means that an odd number of nibbles will be transmitted. Therefore, this error will also cause a CRC error condition
FDX_TX_FRAME_ERR_PRE	Wrong Preamble Sequence transmitted. If this type is selected., the Encoder device substitutes the first nibble of the Start Frame Delimiter with the value ‘1000’ instead of ‘1001’
FDX_TX_FRAME_ERR_PHY	Physical Symbol Error. During Frame Transmission, the MAC-Encoder device asserts the Tx-Error signal, which forces the physical transceiver to transmit ‘HALT’ symbols.

AiUint16 uw_SequenceNumberInit

This parameter defines Initial Sequence Number for this frame. First frame transmission starts with this Sequence Number and adds then the Sequence Number Offset ***uc_SequenceNumberOffset*** , as described at the following parameter.

Value:	***Description:***
FDX_TX_FRAME_SEQ_INIT_AUTO	Sequence Number Init value is set by the Driver automatically.
0…255	Set Sequence Number Offset for this frame (e.g. Error Injection Purposes if invalid values used !)
FDX_TX_FRAME_SEQ_OFF	Sequence Number handling for this transfer is switched off and used as normal data byte.

AiUint16 uw_SequenceNumberOffset

This parameter defines the Sequence Number Offset. This field provides the Sequence Number offset, which is added to the Sequence Number after the Frame has been transmitted. The Sequence number will be incremented until the value of 255 and then it wraps around to 1. Thus, the user can initialize a transmission sequence, which implements more than one frame with the same VL. If the transmission sequence implements N packets with the same VL, this field shall be initialized with N to implement proper sequence numbering for each transmitted VL- frame.

If uw_SequenceNumberInit is switched to FDX_TX_FRAME_SEQ_OFF the value for uw_SequenceNumberOffset is obsolete and don't care.

Value:	***Description:***
FDX_TX_FRAME_SEQ_OFFS_AUTO	Sequence Number Offset is set by the Driver automatically.
0…255	Set Sequence Number Offset for this frame (e.g. Error Injection Purposes if invalid values used !)

AiUInt8   uc_TxIntEnable

This parameter enables signaling by interrupt when this Transfer is transmitted. The interrupt will be report information in the Interrupt loglist

AiUInt32 ul_IntIdent

With this parameter it is possible to define a unique interrupt identifier for this transfer. This identifier will be reported in the Interrupt Loglist

TY_FDX_TX_INSTR_ATTRIB x_TxInstrAttrib

This structure describes the Instruction Attributes in case of ***FDX_TX_FRAME_INSTR uc_FrameType.***

```
typedef struct {
   AiUInt8 uc_Code;
   AiUInt8 uc_Interrupt ;
   AiUInt8 uc_NumOfSubQueues
   AiUInt8 uc_ActivSubQueue
   AiUInt32 aul_SubQueueHandle[FDX_MAX_TX_SUB_QUEUES];
} TY_FDX_TX_INSTR_ATTRIB;
```

AiUInt8 uc_Code

Following Instruction Codes are supported :

Value:	***Description:***
FDX_TX_FRAME_INSTR_NOP	No Operation
FDX_TX_FRAME_INSTR_STOP	Stop Transmission Transmission is stopped if BIU processor runs on this Instruction
FDX_TX_FRAME_INSTR_SYNC	Synchronize BIU Processor waits until Transmit Burst Buffer (between BIU and MAC) is empty
FDX_TX_FRAME_INSTR_CALL	Call a Transmit Sub Queue.
FDX_TX_FRAME_INSTR_ACYC_MARK	Insert a Marker point for execution of Acyclic Instruction. If an Acyclic instruction is defined, it will be scheduled for transmission by reaching this instruction. This marker can be inserted several times in a transfer queue or SubQueue.

AiUInt8 uc_Interrupt

Enable/Disable Interrupt on execution of Instruction

AiUInt8 uc_ActivSubQueue

This parameter is only valid for command FDX_TX_FRAME_INSTR_CALL of other commands i is obsolete.

This parameter defines, which Transmit SubQueue of all referenced SubQueues shall be active first with this call.

The paramete must be in a range of 0 up to uc_NumOfSubQueues-1.

AiUInt8 aul_SubQueueHandle[FDX_MAX_TX_SUB_QUEUES]

This parameter is only valid for command FDX_TX_FRAME_INSTR_CALL of other commands i is obsolete.

This is a array of Transmit Sub Queue handles returened form the command ‘FdxCmdTxSubQueueCreate’

The size of this array can be up to FDX_MAX_TX_SUB_QUEUES entries.

FDX_MAX_TX_SUB_QUEUES is defined to 8.

Note: ***For the usage of Call instructions to sub Queues in an AFDX conform environment it is recommended to use only one Sub Queue. The reason for this limitation is the problem of Sequence numbering. Only the Sub Queue initialized before start of transmitter is taken in account for correct AFDX Sequence numbering.***

TY_FDX_TX_FRAME_ATTRIB_EDE
x_FrameAttribEde

This structure describes the EDE Parameters in case of ***FDX_TX_FRAME_INSTR uc_FrameType.***

```
typedef struct {
   AiUInt32  ul_EDEFlags;
   AiUInt32  ul_EDESourceId;
   AiUInt32  ul_EDEMessageSize;
   AiUInt32  ul_EDETsOffsetHigh;
   AiUInt32  ul_EDETsOffsetLow;
   AiUInt32  ul_EDESNOffset;
} TY_FDX_TX_FRAME_ATTRIB_EDE;
```

AiUInt32 ul_EDEFlags

This parameter defines whether or not the frame is an EDE frame. If EDE is enabled, it can also be used to specify the EDE error injection and Time stamping modesfor the frame.

Possible values are FDX_TX_FRAME_EDE_DIS, or any combination of the other possible values listed below.

Value:	*Description:*
FDX_EDE_TX_FRAME_DIS	No EDE handling is used for this frame
FDX_EDE_TX_FRAME_ENA	EDE handling (Firmware timestamping and CRC calculations) are used for this frame
FDX_EDE_TX_FRAME_FIRST	This frame is the first fragment of a EDE message (Must be set for single frame messages)
FDX_EDE_TX_FRAME_LAST	This frame is the last fragment of a EDE message (Must be set for single frame messages)
FDX_EDE_TX_FRAME_ERR_TS	The board firmware is instructed not to insert an EDE TS for this frame
FDX_EDE_TX_FRAME_ERR_CRCX	An EDE CRC-X error is transmitted
FDX_EDE_TX_FRAME_ERR_CRCY	An EDE CRC-Y error is transmitted
FDX_EDE_TX_FRAME_ERR_NO_SN	Instructs the Firmware that an EDE Sequence number shall not be inserted into the frame. This is only valid for frames with the flag FDX_TX_FRAME_EDE_FIRST.

AiUInt32 ul_EDESourceId

The EDE Source ID used for calculating the EDE CRCs for the frame.

AiUInt32 ul_EDEMessageSize

The remaining EDE message size for the complete EDE message. This includes the EDE Sequence Number, Timestamp, payload, and both CRCs.

AiUInt32 ul_EDETsOffsetHigh

31………………………………………………….16	*15…………………………………………..0*
Reserved (Coded 0)	EDE Time Stamp Offset (bit 32 – 47)

AiUInt32 ul_EDETsOffsetLow

31……………………………………………………………………………………………0
EDE Time Stamp Offset (bit 0 – 31)

The parameters ul_EDETsOffseHigh and ul_EDETsOffsetLow combine to form the 48-bit 2's compliment offset that is added by the firmware to the 48-bit EDE counter before applying to the frame as the EDE transmit timestamp.

AiUInt32 ul_EDESNOffset

The EDE SN offset which is added to the EDE SN for the current EDE port (specified by ul_EDEPortIndex) after the frame is transmitted. Possible values are 0..7.

Output:

None

Return Value

Returns FDX_OK on success or a negative error code on error.

Error Codes: FDX_ERR

3.3.2 Individual (UDP Port Oriented) Transmitter Functions

3.3.2.1 FdxCmdTxEDECreatePort

Prototype:

AiReturn FdxCmdTxEDECreatePort(AiUInt32 ul_Handle,
const TY_FDX_UDP_DESCRIPTION *px_UdpDescription,
const TY_FDX_EDE_DESCRIPTION* px_EDETxDescription,
AiUInt32 *pul_EDEHandle);

Driver Command:

FDX_EDE_TX_CREATE_PORT (0x0000806E)

Purpose:

This function is used to create and configure a new EDE enabled UDP port. Default settings after creation of the EDE port are:

- UDP port enabled
- Error injection: OFF
- Skew (redundant mode only): 0 usec.

To change the settings of the UDP port the function ***FdxCmdTxUDPControl*** can be used. This function can be used only if transmitter is not running.

Input:

TY_FDX_UDP_DESCRIPTION
****px_UdpDescription***

Pointer to a structure, which describes the UDP connection.

```
typedef struct {
   AiUInt32 ul_PortType;
   TY_FDX_QUINTUPLET x_Quint;
   AiUInt32 ul_SubVlId;
   AiUInt32 ul_UdpNumBufMessages;
   AiUInt32 ul_UdpMaxMessageSize;
   AiUInt32 ul_UdpSamplingRate;
}TY_FDX_UDP_DESCRIPTION

typedef struct _quintuplet {
   AiUInt32 ul_UdpSrc;
   AiUInt32 ul_IpSrc
   AiUInt32 ul_VlId;
   AiUInt32 ul_IpDst;
   AiUInt32 ul_UdpDst;
} TY_FDX_QUINTUPLET;
```

AiUInt32 ul_PortType

Type of the port connection

Value	*Description*
FDX_UDP_SAMPLING	Port is a sampling port. Each Message is represented by one MAC Frame. The size of the messages is fix.
FDX_UDP_QUEUING	Port is a queuing port: Each Message can be represented by one or more MAC Frame. Reassemble will be done in the IP layer. A Message can have a size up to 8kByte.

struct TY_FDX_QUINTUPLET

This structure describes a full identification of the communication.

AiUInt32 x_Quint.ul_UdpSrc

UDP port-number of the source UDP port.

AiUInt32 x_Quint.ul_IpSrc

IP address of the source partition.

AiUInt32 x_Quint.ul_VlId

Virtual Link Identifier

AiUInt32 x_Quint.ul_IpDst

Destination IP address

AiUInt32 x_Quint.ul_UdpDst

Destination UDP port

AiUInt32 ul_SubVlId;

Sub Virtual Link Identifier (Sub VLs are only relevant in Tx Mode). This value must be in a range from 1 to 4. If Sub VLs are not used, the Sub VL Id equals to 1.

Note: ***Must be consistent with parameter ul_SubVls of function FdxCmdTxCreateVL or FdxCmdTxCreateHiResVL.***

AiUInt32 ul_UdpNumBufMessages

Number of messages which can be stored by the onboard Target software.

Size of the local Buffer which should be created by the onboard Target software to store data of the created UDP port can be calculated by ul_UdpNumBufMessages * ul_UdpMaxMessageSize.

For sampling ports the number of messages has to be set to 1.

For queuing ports an adequate buffer depth have to be provided.

If this value is set to zero, the onboard target software will set to a default value.

AiUInt32 ul_UdpMaxMessageSize;

Maximum size of a message to send.

For a sampling port, this is the fix size of the sampling message, which means the size without the header overhead (MAC, IP and UDP).

For a queuing port this is the maximum size of the variable length message.

This is the maximum size of the UDP payload which includes the EDE Header (SN and TS) and the EDE CRCs.

Port Type	*Value*
Sampling	Must be equal or smaller than ul_MaxFrameLength – (MAC, IP and UDP) defined in function ***FdxCmdTxCreateVL*** or ***FdxCmdTxCreateHiResVL.***
Queuing	Must be equal or smaller than 8Kbytes (max. UDP message size)

AiUInt32 ul_UdpSamplingRate;

Specifies the sampling rate for this frame on the AFDX bus in milliseconds. This means a repetition rate the data shall be sent cyclic. Specify value starting with 1msec in multiples of 1msec. Refer also to parameter ul_Bag of function ***FdxCmdTxCreateVL*** or ***FdxCmdTxCreateHiResVL*** which restrict usage of sampling rate.

Note: ***This parameter is only applicable for a sampling port.***

TY_FDX_EDE_DESCRIPTION px_EDETxDescription*

Pointer to the EDE Create port input data structure.

```
typedef struct {
   AiUInt32 ul_EDESourceId;
   AiUInt32 ul_EDESubscriberHandle;
} TY_FDX_EDE_TX_DESCRIPTION;
```

AiUInt32 ul_EDESourceId

The 32-bit EDE source Id assigned to this port and used for CRC calculations of all transmitted messages.

AiUInt32 ul_EDESubscriberHandle

The handle to the EDE Subscriber to which this port is associated.

Output:

*AiUInt32 *pul_EDEHandle*

Handle to the UDP port.

Return Value

Returns FDX_OK on success or a negative error code on error.

Error Codes: FDX_ERR

3.3.2.2 **FdxCmdTxEDEWrite**

Prototype:

AiReturn FdxCmdTxEDEWrite(AiUInt32 ul_Handle, const AiUInt32 ul_EDEHandle,
const TY_FDX_TX_EDE_WRITE_IN* px_TxEDEWriteIn,
TY_FDX_TX_EDE_WRITE_OUT* px_TxEDEWriteOut);

Driver Command:

FDX_EDE_TX_WRITE (0x0000806F)

Purpose:

This function is used to write a pure message to an EDE port. This port can be a sampling or a queuing port. If the data size is not applicable for the data size associated to this port, this function will return an error.

This function can be used to inject EDE errors into the written message.

For sampling ports this function initializes / modifies data contents.

For queuing ports a transmission is initiated when data is written to a UDP port.

This function can be used if the transmitter is running or not running.

For sampling ports this function should be called before the port is started to initialize the data contents of the sampling port.

Input:

AiUInt32 ul_EDEPortHandle

The handle to the UDP port

TY_FDX_TX_EDE_WRITE_IN px_TxEDEWriteIn*

Pointer to the EDE UDP port intput data structure.

```
typedef struct {
   AiUInt32 ul_ByteCount;
   TY_FDX_EDE_PARAM x_EdeParam;
   void     *pv_Data;
} TY_FDX_TX_UDP_EDE_WRITE_IN;
```

AiUInt32 ul_ByteCount

The size of the message, EDE payload in bytes, to be written to the UDP port.

TY_FDX_EDE_PARAM x_EdeParam

The EDE parameter control structure for the message.

```
typedef struct {
   AiUInt32 ul_EDEFlags;
   AiUInt32 ul_EDETsOffsetHigh;
   AiUInt32 ul_EDETsOffsetLow;
   AiUInt32 ul_EDESnOffset;
} TY_FDX_EDE_PARAM;
```

AiUInt32 ul_EDEFlags

The EDE flags which specify the error injection control for the message. This may be set to either FDX_EDE_TX_ERR_DIS, or a logical combination of all other flags.

Constant	***Description***
FDX_EDE_TX_ERR_DIS	EDE Error Injection is disabled
FDX_EDE_TX_ERR_SN	The EDE SN field is filled with the value given in ul_EDESnOffset
FDX_EDE_TX_ERR_TS	The EDE TS field is filled with the value given in ul_EDETsOffsetHigh and ul_EDETsOffsetLow
FDX_EDE_TX_ERR_CRCX	Erroneous CRCX values are inserted into the message
FDX_EDE_TX_ERR_CRCY	Erroneous CRCY values are inserted into the message.

AiUInt32 ul_EDESnOffset

The offset added to the last EDE SN before it is inserted into the message. For normal (non- error) cases a value of 1 shall be used. If the flag FDX_EDE_TX_ERR_SN is set, then this specifies the absolute value of the EDE SN to be inserted into the message.

AiUInt32 ul_EDETsOffsetHigh

The high order 16-bits of the 48-bit offset to be added to the associated EDE subscriber clock before inertion of the EDE timestamp into the message. If the flag FDX_EDE_TX_ERR_TS is set, then this specifies the absolute value of the EDE TS to be inserted into the message.

AiUInt32 ul_EDETsOffsetLow

The low order 32-bits of the 48-bit offset to be added to the associated EDE subscriber clock before inertion of the EDE timestamp into the message. If the flag FDX_EDE_TX_ERR_TS is set, then this specifies the absolute value of the EDE TS to be inserted into the message.

void *pv_Data

A pointer to the memory location holding the EDE payload data to be written. This is a pointer to ul_ByteCount bytes of data.

Output:

TY_FDX_TX_EDE_WRITE_OUT px_TxEDEWriteOut*

Pointer to the EDE UDP port output data structure.

```
typedef struct {
   AiUInt32 ul_BytesWritten;
} TY_FDX_TX_UDP_EDE_WRITE_OUT;
```

AiUInt32 ul_BytesWritten

The number of payload bytes written to the port.

Return Value

Returns FDX_OK on success or a negative error code on error.

Error Codes: FDX_ERR

3.3.2.3 **FdxCmdTxUDPWrite**

Prototype:

```
AiReturn FdxCmdTxUDPWrite (  AiUInt32 ul_Handle,
                             const AiUInt32 ul_UdpHandle
                             const AiUInt32 ul_ByteCount
                             const void *pv_Data
                             AiUInt32 *pul_BytesWritten);
```

Driver Command:

FDX_TX_UDP_WRITE ***(0x0000805D)***

Purpose:

This function is used to write a pure message to a UDP or EDE port. This port can be a sampling or a queuing port. If the data size is not applicable for the data size associated to this port, this function will return an error.

For sampling ports this function initializes / modifies data contents.

For queuing ports a transmission is initiated when data is written to a UDP port.

This function can be used if the transmitter is running or not running.

For sampling ports this function should be called before the port is started to initialize the data contents of the sampling port.

Input:

AiUInt32 ul_UdpHandle

See description of *FdxCmdTxUDPCreatePort* or *FdxCmdTxEDECreatePort*.

AiUInt32 ul_ByteCount

For UDP Ports:

Number of bytes to write to this UDP port. The value must be equal to or smaller than ul_MaxMessageSize defined with ***FdxCmdTxUDPCreatePort()***.

For EDE Ports:

Number of bytes to write to this EDE port. The value must be equal to or smaller thatn ul_MaxMessageSize – 12, where ul_MaxMessageSize is the value defined in **FdxCmdTxEDECreatePort()**.

Port Type	*Comment*
Sampling	The value does not influence transmitted frame size. When the number is smaller than ul_MaxMessageSize only the first part of the UDP buffer is updated.
Queuing	The value is equivalent to transmitted UDP message size.

void *pv_Data

For UDP Ports, pointer to a buffer containing the UDP payload data to write. For EDE Ports, pointer to a buffer containing the EDE payload data to write.

Output:

AiUInt32 *pul_BytesWritten

Number of bytes actually written. Might be smaller than ul_ByteCount if UDP buffer is full. (Queuing ports ul_NumBufMessages defined with FdxCmdTxUDPCreatePort)

Return Value

Returns FDX_OK on success or a negative error code on error.

Error Codes: FDX_ERR

3.4 EDE Receive Functions

Table 3.4-1: Receiver Functions

Function	Description
Global Receiver Functions	
FdxCmdRxEDEVlControl	Controls the operation of an EDE enabled Rx VL
VL-Oriented Receiver Functions	
FdxCmdRxEDECreatePort	Creates a new Rx EDE enabled AFDX Communication port
FdxCmdRxEDEControl	Controls the operation of the Rx EDE port.
FdxCmdRxUDPRead	Reads a message from an EDE enabled AFDX Comm. port

3.4.1 Global Receiver Functions

3.4.1.1 FdxCmdRxEDEVlControl

Prototype:

AiReturn FdxCmdRxEDEVlControl (AiUInt32 ul_Handle,
const TY_FDX_RX_VL_CTRL *px_VLControl,
const TY_FDX_RX_VL_DESCRIPTION *px_VlDescription,
const TY_FDX_RX_VL_EDE_DESCRIPTION
****px_VlEDEDescription);***

Driver Command:

FDX_EDE_RX_EDE_VL_CONTROL (0x00008079)

Purpose:

This function is used to control the operation of an EDE enabled receive VL. This function should only be used for VL Oriented mode operations (not Chronological monitor or recording receive modes).

Input:

TY_FDX_RX_VL_CTRL *px_VLControl

A pointer to a structure which describes the Virtual Link related parameters for the receiver.

```
typedef struct {
   AiUInt32 ul_VLId;
   AiUInt32 ul_VLRange;
   AiUInt32 ul_EnableMode;
   AiUInt32 ul_PayloadMode;
   AiUInt32 ul_TCBIndex;
} TY_FDX_RX_VL_CTRL;
```

AiUInt32 ul_VLId

Virtual Link Identifier. A value in a range from 0 to 65535. This value is part of the MAC destination address.

A range of VLs to which the TY_FDX_RX_VL_CTRL settings are applied can be given by setting the Start VL Identifier into the ***ul_VLId***, and the number of VLs (starting at the VL Id in the ***ul_VLId*** parameter) into the ***ul_VLRange*** parameter. This is not allowed together with ***ul_EnableMode*** = FDX_RX_VL_ENA_EXT of the TY_FDX_RX_VL_CTRL settings !

AiUInt32 ul_VLRange

Number of VLs which are affected by the TY_FDX_RX_VL_CTRL settings, beginning with the VL set in the ***ul_VLId*** parameter.

AiUInt32 ul_EnableMode

Mode which is used for the given VL.

Value	***Comment***
FDX_RX_VL_ENA_EXT	Virtual Link is enabled for extended operation. Frames are stored, dependent on the receive mode set via the *FdxCmdModeControl* command. If the receiver is running in VL oriented reception mode, frames of the given VL are stored in an individual buffer, with the size given by the *ul_VLBufSize* parameter. When the receiver is in chronological mode, the frames of the given VL are stored in a Global Monitor buffer. In this mode, the verification mode and the extended filtering can be applied to the given VL if necessary.

AiUInt32 ul_PayloadMode

Defines the Payload Mode, for data flow reduction. So you can specify up to which level the data shall be stored in the data buffer. With this function you can define this level for each VL.

Using this mode you can see the traffic on the bus without monitoring and transferring the full data load.

Value	***Description***
FDX_PAYLOAD_FULL	Frames will be stored with full payload in the Monitor buffer. This means the full Ethernet (MAC) frame is available for the application.

AiUInt32 ul_TCBIndex

Reserved

*TY_FDX_RX_VL_DESCRIPTION *px_VlDescription*

```
typedef struct {
   AiUInt32 ul_VerificationMode;
   AiUInt32 ul_Bag;
   AiUInt32 ul_Jitter;
   AiUInt32 ul_MaxFrameLength;
   AiUInt32 ul_MaxSkew;
   AiUInt32 ul_VLBufSize;
   TY_FDX_RX_VL_EXT_FLT x_VLExtendedFilter;
   AiUInt32 ul_MinFrameLength;
} TY_FDX_RX_VL_DESCRIPTION;
```

Note: ***Structure TY_FDX_RX_VL_DESCRIPTION is used only if parameter ul_EnableMode is FDX_RX_VL_ENA_EXT, pointer can be set to NULL otherwise.***

AiUInt32 ul_VerificationMode

Control parameter to set up verification mode for the given VL. This parameter is only applicable if the VL is set to **FDX_RX_VL_ENA_EXT** mode.

Value	*Description*	*Comment*	*R*	*S*
FDX_RX_VL_CHECK_DISA	Verification disabled			
FDX_RX_VL_CHECK_ENA_DEFAULT	Default Setting	This is a combination of the following values, dependant on if the Port is **R**edundant or **S**ingle. See the columns 'R' and 'S' for the specification of this value.	default redundant	default single
FDX_RX_VL_CHECK_REDMAM	Redundancy Management	Enable Redundancy Management like described in AFDX End System Specification [3]	✓	
FDX_RX_VL_CHECK_TRAFIC	Traffic shaping Verification	Enable Traffic Shaping Verification like described in AFDX Switch Specification [4]		✓
FDX_RX_VL_CHECK_FRAMESIZE	Vl specific Framesize Check	Maximum frame size for the given VL is checked.	✓	✓
FDX_RX_VL_CHECK_SNINTEG	Sequence Number Integrity check	Sequence numbering of the incoming frames are checked	✓	
FDX_RX_VL_CHECK_INVPAC	Invalid Packet processing	All Packets, also the erroneous, will be passed through to the buffer		

Note: ***If a verification mode is enabled, the structure px_VLDescription must be properly setup.***

AiUInt32 ul_Bag

Bandwidth Allocation Gap (BAG) in milliseconds for the defined Virtual Link.

Possible Values for the BAG are 1 to 128ms defined as 2^k[in ms], where k can have a range from integer 0 to 7.(Source [5]). The specified value in ul_Bag will be set by the target SW to the next valid value.

E.g. a value of ul_Bag of 70 will be set to 64 (ms).

AiUInt32 ul_Jitter

Maximum allowed Jitter Value in µs, for the given Virtual Link.

Possible Range for the Jitter 1 to 65535 µs.

AiUInt32 ul_MaxFrameLength

Maximum Frame Length of a MAC Frame specified for this Virtual Link.

AiUInt32 ul_MinFrameLength

Minimum Frame Length of a MAC Frame specified for this Virtual Link.

AiUInt32 ul_MaxSkew

The maximum time difference between the arrival time of redundant frame with the same sequence number in µs. Possible Range for the MaxSkew 0 to 65535 µs.

AiUInt32 ul_VLBufSize

Size of the local Buffer which should be created by the onboard Target software to store data of the selected VL.

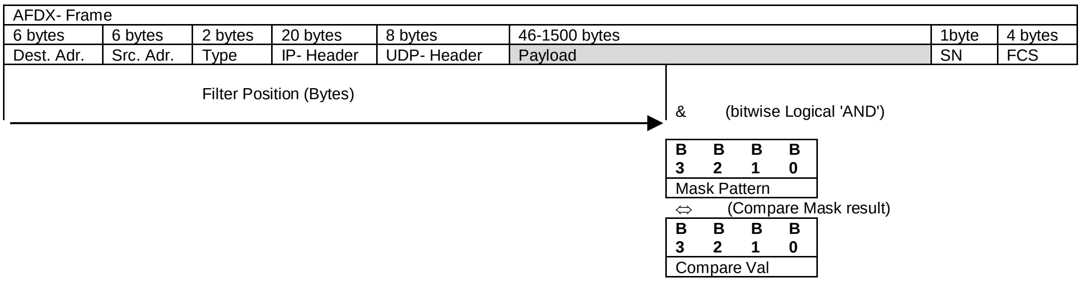
This parameter is only applicable if the receive port works in VL oriented mode (see ***FdxCmdModeControl*** command)

If this value is set to 0, the onboard target software will set to a default value.

TY_FDX_RX_VL_EXT_FLT x_VLExtendedFilter

By defining this structure, an extended, second level, frame Filter for each Virtual Link can be applied.

This extended filter is a generic filter to mask and compare four bytes of the data stream. These four bytes can be located on any position in the frame, specified by the filter position. The following figure shows the mechanism of this filter



AFDX- Frame							
6 bytes	6 bytes	2 bytes	20 bytes	8 bytes	46-1500 bytes	1byte	4 bytes
Dest. Adr.	Src. Adr.	Type	IP- Header	UDP- Header	Payload	SN	FCS

Figure 3.4.1-1: Mechanism of second level Filter

```
typedef struct {
   AiUInt32 ul_FilterMode
   AiUInt32 ul_FilterPosition;
   AiUInt32 ul_FilterMask;
   AiUInt32 ul_FilterData;
} TY_FDX_VL_EXT_FLT;
```

AiUInt32 ul_FilterMode

Filter Mode of the second level filter

Value	*Description*
FDX_DIS	Disable filtering
FDX_RX_VL_FLT_ENA	Enable filtering, frame is stored if the filter condition matches
FDX_RX_VL_FLT_ENA_INV	Enable filtering, frame is stored if the filter condition does not match

AiUInt32 ul_FilterPosition

Filter position offset to the start of the AFDX frame, where the value shall be compared.

AiUInt32 ul_FilterMask

Filter Mask to mask the bits of four consecutive bytes for comparing with the filter data. If bit is set (1) the according bit in filter data is relevant. If bit is not set (0) the according bit in filter data is don’t care.

AiUInt32 ul_FilterData

Filter Data to compare with the result of masking.

(Example: Checking for Udp-Destination = 10 decimal.

The following settings can be used:
ul_FilterPosition = 36 (Udp-Destination)
ul_FilterMask = FFFF0000hex (mask out UDP-length)
ul_FilterData = 000A0000hex (check for UDP-destination 10decimal))

*TY_FDX_RX_VL_EDE_DESCRIPTION *px_VlEDEDescription*

```
typedef struct {
   AiUInt32   ul_EDESourceIndex;
} TY_FDX_RX_VL_EDE_DESCRIPTION;
```

AiUInt32 ul_EDESourceIndex

The Time Offset Table index for the remote End System which is the source for all data frames received on this EDE enabled virtual link.

Output:

None

Return Value

Returns FDX_OK on success or a negative error code on error.

Error Codes: FDX_ERR

3.4.2 VL-Oriented Receiver Functions

3.4.2.1 **FdxCmdRxEDECreatePort**

Prototype:

AiReturn FdxCmdRxEDECreatePort(AiUInt32 ul_Handle,
const TY_FDX_UDP_DESCRIPTION* px_UdpDescription,
const TY_FDX_ EDE_DESCRIPTION* px_EDERxDescription,
AiUInt32 *pul_EDEHandle);

Driver Command:

FDX_EDE_RX_EDE_CREATE_PORT (0x00008086)

Purpose:

This function is used to create and configure a new EDE enabled UDP port.

Input:

TY_FDX_UDP_DESCRIPTION
****px_UdpDescription***

Pointer to a structure, which describes the UDP connection.

```
typedef struct {
   AiUInt32 ul_PortType;
   struct _quintuplet {
      AiUInt32 ul_UdpSrc;
      AiUInt32 ul_IpSrc
      AiUInt32 ul_VlId;
      AiUInt32 ul_IpDst;
      AiUInt32 ul_UdpDst;
   }x_Quint;
   AiUInt32 ul_SubVlId;
   AiUInt32 ul_UdpNumBufMessages;
   AiUInt32 ul_UdpMaxMessageSize;
   AiUInt32 ul_UdpSamplingRate;
}TY_FDX_UDP_DESCRIPTION;
```

AiUInt32 ul_PortType

Type of the port connection

Value	*Description*
FDX_UDP_SAMPLING	Port is a sampling port. Each Message is represented by one MAC Frame. The size of the messages is fixed.
FDX_UDP_QUEUING	Port is a queuing port: Each Message can be represented by one or more MAC Frame. Reassembling will be done in the IP layer. A Message can have a size up to 8kByte.

struct _quintuplet

This structure provides a full identification of the communication.

AiUInt32 x_Quint.ul_UdpSrc

UDP port-number of the source UDP port.

AiUInt32 x_Quint.ul_IpSrc

IP address of the source partition.

AiUInt32 x_Quint.ul_VlId

Virtual Link Identifier

AiUInt32 x_Quint.ul_IpDst

Destination IP address

AiUInt32 x_Quint.ul_UdpDst

Destination UDP port

AiUInt32 uw_SubVlId;

Not relevant in Rx Mode.

AiUInt32 ul_UdpNumBufMessages

Number of messages which should be stored by the onboard Target software.

Size of the local Buffer which should be created by the onboard Target software to store data of the created UDP port can be calculated by ul_UdpNumBufMessages * ul_UdpMaxMessageSize.

For sampling ports the number of messages has to be set to 1.

For queuing ports an adequate buffer depth has to be provided.

If this value is set to zero, the onboard target software will be set to a default value.

AiUInt32 ul_UdpMaxMessageSize;

Maximum size of a message to receive.

For a sampling port, this is the fixed size of the sampling message, which means the size without the header overhead (MAC, IP and UDP).

For a queuing port this is the maximum size of the variable length message.

(Queuing: 0..ize ≤ 8kBytes, Sampling: ul_UdpMaxMessageSize)

Port Type	***Value***
Sampling	0..1471 bytes
Queuing	0..8 kBytes

Note: ***If received message exceeds maximum size this message will be cut off and only ul_UdpMaxMessageSize bytes will be saved.***

AiUInt32 ul_UdpSamplingRate;

Not relevant in Rx Mode.

TY_FDX_EDE_DESCRIPTION px_EDERxDescription*

Pointer to the EDE Create port input data structure.

```
typedef struct {
   AiUInt32 ul_EDESourceId;
   AiUInt32 ul_EDESubscriberHandle;
} TY_FDX_EDE_TX_DESCRIPTION;
```

AiUInt32 ul_EDESourceId

The 32-bit EDE source Id assigned to this port and used for CRC calculations of all transmitted messages.

AiUInt32 ul_EDESubscriberHandle

The handle to the EDE Subscriber to which this port is associated.

Output:

*AiUInt32 *pul_EDEPortHandle*

The handle to the newly created Rx EDE port.

Return Value

Returns FDX_OK on success or a negative error code on error.

Error Codes: FDX_ERR

3.4.2.2 **FdxCmdRxEDEControl**

Prototype:

AiReturn FdxCmdRxEDEControl (AiUInt32 ul_Handle,
const AiUInt32 ul_UdpHandle,
const TY_FDX_RX_UDP_CONTROL *px_UdpControl,
const TY_FDX_RX_EDE_CONTROL *px_EdeControl);

Driver Command:

FDX_RX_EDE_CONTROL (0x00008087)

Purpose:

This function allows the user to configure several settings of a EDE Receive Port.

Input:

AiUInt32 ul_UdpHandle

The handle of the associated EDE UDP port.

*TY_FDX_RX_UDP_CONTROL *px_UdpControl*

Pointer to a UDP control structure.

```
typedef struct {
   AiUInt32 ul_NetSelect;
   AiUInt32 ul_InterruptControl;
} TY_FDX_RX_UDP_CONTROL;
```

AiUInt32 ul_NetSelect

Specifies the network to which the Rx UDP port is bound.

Value	***Description***
FDX_RX_UDP_BOTH	The port will receive messages from both networks
FDX_RX_UDP_ONLY_A	The port will receive messages from only network A
FDX_RX_UDP_ONLY_B	The port will receive messages from only network B

AiUInt32 ul_InterruptControl

This parameter provides for an interrupt on reception of each message on a UDP-Port.

Value	Description
FDX_RX_UDP_NOINT	No interrupt
FDX_RX_UDP_INT	Interrupt
FDX_RX_UDP_UDF	Not implemented

TY_FDX_RX_EDE_CONTROL *px_EdeControl

Pointer to the EDE Control structure.

```
typedef struct {
    AiUInt32 ul_VerificationMode;
    AiUInt32 ul_EnableMode;
    AiUInt32 ul_AgeMax;
    AiUInt32 ul_MaxSkew;
    AiUInt32 ul_EdeSnValidWindow;
     AiUInt32 ul_EdeRmInvalidSnWindow;
} TY_FDX_RX_EDE_CONTROL;
```

AiUInt32 ul_VerificationMode

Defines the verification mode of the EDE Rx Port.

Value	Description
FDX_RX_EDE_CHECK_DISA	All EDE Verifications are disabled
FDX_RX_EDE_CHECK_ENA_DEFAULT	The default EDE Verifications, FDX_RX_EDE_CHECK_CRCX and FDX_RX_EDE_CHECK_CRCY, are enabled
FDX_RX_EDE_CHECK_CRCX	Enables validation of EDE CRCX
FDX_RX_EDE_CHECK_CRCY	Enables validation of EDE CRCY
FDX_RX_EDE_CHECK_AGE	Enables EDE Age Validation
FDX_RX_EDE_CHECK_SNINTEG	Enables EDE Sequence number integrity checking
FDX_RX_EDE_CHECK_REDMAN	Enables EDE Redundancy Management

Any logical combination (‘or’) of the above flags can be used.

AiUInt32 ul_EnableMode

Defines how the UDP port will handle messages that fail any of the EDE verifications.

Value	Description
FDX_RX_EDE_INVDISCARD	All messages that fail verification are discarded
FDX_RX_EDE_INVCRCX	Messages that fail CRCX verification will be processed
FDX_RX_EDE_INVCRCY	Messages that fail CRCY verification will be processed
FDX_RX_EDE_INVAGE	Messages that fail AGE validation will be processed
FDX_RX_EDE_INVSN	Messages that fail SN validation will be processed
FDX_RX_EDE_INVDEFAULT	Messages that fail CRCX/Y, AGE, and SN validation will be processed. (logical combination of: FDX_RX_EDE_INVCRCX, FDX_RX_EDE_INVCRCY, FDX_RX_EDE_INVAGE, FDX_RX_EDE_INVSN)

Any logical combination (‘or’) of the above flags can be used.

AiUInt32 ul_AgeMax

Maximum allowed age, in micoseconds, for received messages. This value is used when Age validation is enabled (FDX_RX_EDE_CHECK_AGE).

When the EDE port is created, this parameter is initialized with a default value of 67 Seconds.

AiUInt32 ul_MaxSkew

Maximum skew, in microseconds, for redundant messages received on Network A and Network B. This value is used for EDE Redundancy management (FDX_RX_EDE_CHECK_REDMAN).

When the EDE port is created, this parameter is initialized with a default value of 200mSec.

AiUInt32 ul_EdeSnValidWindow

Defines the window of valid EDE Sequence numbers to be used when EDE Sequence Number integrity checking is enabled (FDX_RX_EDE_CHECK_SNINTEG).

When the EDE port is created, this parameter is initialized with a default value of 8.

(See [5]: EIO-7165, EIO-7166, EIO-7167)

AiUInt32 ul_EdeRmInvalidSnWindow

Defines the window of invalid EDE Sequence numbers to be used when EDE Redundancy Management is enabled (FDX_RX_EDE_REDMAN).

When the port is created, this parameter is initialized with a default value of 1,024.

See [5]: EIO-7060.

Output:

None

Return Value

Returns FDX_OK on success or a negative error code on error.

Error Codes: FDX_ERR

3.4.2.3 **FdxCmdRxUDPRead**

Prototype:

```
AiReturn FdxCmdRxUDPRead (  AiUInt32 ul_Handle,
                            const AiUInt32 ul_UdpHandle,
                            AiUInt32 ul_MsgCount,
                            AiUInt32 *pul_MsgRead,
                            void *pv_ReadBuffer);
```

Driver Command:

FDX_RX_UDP_READ   *(0x0000807F)*

Purpose:

This function reads data from a UDP or EDE connection oriented port. This is the same function that is defined in the FDX High Level Application Programmer's Interface. It is re-documented here to show that alternative message entry format that is provided as output in the case of reading a message from an EDE enabled UDP port.

Input:

AiUInt32 ul_UdpHandle

See description of *FdxCmdRxUDPCreatePort*.

AiUInt32 ul_MsgCount

Number of Messages to read. This means the newest *ul_MsgCount* Entries. For a sampling port there is a maximum of one message to read.

Note: ***For sampling ports the number of messages to read shall be always 1.***

Output:

*AiUInt32 *pul_MsgRead*

Number of Messages actually read.

*void *pv_ReadBuffer*

Pointer to the data buffer the Entries should be stored. Required size of buffer can be calculated: ul_UdpMaxMessageSize * ul_MsgCount.

The ul_UdpMaxMessageSize is defined with function ***FdxCmdRxUDPCreatePort***.

One Entry specifies one Message, which means one complete sampling or queuing message. For special system information and administration a Fixed sized Header is preceded.

The following figure shows the schematic of such an entry for data read from a UDP port.

	UDP Message Buffer Layout			
	31 24	23 16	15 8	7 0
UDP Header	Time Tag High			
	Time Tag Low			
	Message Size			
	Error Information			
UDP Frame	Received UDP- FRAME (sampling message: up to UDP payload (1 – 1471 bytes queuing message: up to 8Kbytes)			

Figure 3.4.2-1: RX UDP Message Buffer Layout

TY_FDX_UDP_HEADER

This is a structural description of the UDP header

```
typedef struct _fdx_udp_header {
   TY_FDX_FW_IRIG_TIME x_FwIrigTime;
   AiUInt32 ul_MsgSize;
   AiUInt32 ul_ErrorInfo;
} TY_FDX_UDP_HEADER;
```

TY_FDX_FW_IRIG_TIME x_FwIrigTime

The Firmware IRIG Time Tag information is from the last received message. For a queuing port where the messages can be fragmented, it is the Time Tag of the last received fragment.

```
typedef struct {
   AiUInt32 ul_TtHigh;
   AiUInt32 ul_TtLow;
} TY_FDX_FW_IRIG_TIME;
```

AiUInt32 ul_TtHigh;

Timetag word in firmware format. The higher part of the time tag, contains the minutes of hour, hours of day and day of year.

For further description see Firmware specification.

AiUInt32 ul_TtLow;

Timetag word in firmware format. The lower part of the time tag, contains the Microseconds of second, seconds of minutes and minutes of hour.

To get a ‘C’ structured information of the Time Tag you can use the functions **Fehler! Verweisquelle konnte nicht gefunden werden.** ().

AiUInt32 ul_MsgSize;

UDP payload size in bytes of received Frame.

AiUInt32 ul_ErrorInfo;

Bit 31-16	***Bit15-0***
Error information as available in FdxCmdMonQueueRead variable uw_ErrorField	Queuing ports additional error information: IP_REASS_ERROR_SYNC IP_REASS_ERROR_ORDER IP_REASS_ERROR_FRAG IP_REASS_ERROR_SIZE IP_REASS_ERROR_BUF

The following figure shows the schematic of such an entry for data read from an EDE port.

	UDP Message Buffer Layout			
	31 24	23 16	15 8	7 0
UDP Header	EDE Receive Time Stamp			
	EDE Receive Time Stamp			
	IRIG Time Stamp High			
	IRIG Time Stamp Low			
	Message Size			
	Error Information			
	EDE Transmit Time Stamp (High)		EDE Sequence Number	
	EDE Transmit Time Stamp (Low)			
UDP Frame	Received EDE Payload (sampling message: up to EDE payload (1 – 1459 bytes queuing message: up to 8180 bytes)			
	EDE CRCX (High)	EDE CRCX (Low)	EDE CRCY (High)	EDE CRCY (Low)

Figure 3.4.2-2: RX EDE Message Buffer Layout

TY_FDX_EDE_HEADER

```
typedef struct {
   AiUInt32               ul_EDERxTsHigh;
   AiUInt32               ul_EDERxTsLow;
   TY_FDX_FW_IRIG_TIME    x_FwIrigTime;
   AiUInt32               ul_MsgSize;
   AiUInt32               ul_ErrorInfo;
   AiUInt16               uw_EDESn;
   AiUInt16               uw_EDETxTsHigh;
   AiUInt32               ul_EDETxTsLow;
}TY_FDX_EDE_HEADER;
```

AiUInt32 ul_EDERxTsHigh

The high order 16-bits of the 48-bit EDE timestamp applied locally on reception of the message.

AiUInt32 ul_EDERxTsLow

The low order 32-bits of the 48-bit EDE timestamp applied locally on reception of the message.

TY_FDX_FW_IRIG_TIME x_FwIrigTime

See FdxCmdRxUDPRead for a description of this structure.

AiUInt32 ul_MsgSize

The size, in bytes of the received message. (The number of bytes in the EDE payload).

AiUInt32 ul_ErrorInfo

Specifies error information about the received frame.

Bit 31-16	*Bit 15-0*
Error information as available in FdxCmdMonQueueRead variable uw_ErrorField	For Queuing ports additional information: IP_REAS_ERROR_SYNC IP_REAS_ERROR_ORDER IP_REAS_ERROR_FRAQ IP_REAS_ERROR_SIZE IP_REAS_ERROR_BUF EDE_CRCX_ERROR EDE_CRCY_ERROR EDE_SN_ERROR EDE_SN_ERROR_UNK EDE_AGE_ERROR EDE_AGE_ERROR_UNK

AiUInt16 uw_EDESn

The EDE Sequence number of the received message.

AiUInt32 ul_EDETxTsHigh

The high order 16-bits of the EDE transmit timestamp received in the incoming message.

AiUInt32 ul_EDETxTsLow

The low order 32-bits of the EDE transmit timestamp received in the incoming message.

Return Value

Returns FDX_OK on success or a negative error code on error.
Error Codes: FDX_ERR

3.5 EDE Target Independent Functions

Table 3.5-1: Target Independent Administration Functions

Function	*Description*
FdxInitTxEDEFrameHeader	Supports a default initialization of a Transmit Header Structure for an EDE Frame.
FdxCalcEDECRCx	Calculates an EDE CRC X
FdxCalcEDECRCy	Calculates and EDE CRC Y

3.5.1 FdxInitTxFrameHeaderEx

Prototype:

AiReturn FdxInitTxFrameHeaderEx (TY_FDX_TX_FRAME_HEADER_EX
****px_TxFrameHeaderEx)***

Driver Command

none

Purpose:

This function initializes a Transmit Frame Header Structure for Standard Frame (No Instruction Type). This structure is used for defining a Generic Transmit Queue entry with the ***FdxCmdTxQueueWrite*** function.

Input:

TY_FDX_TX_FRAME_HEADER
**px_TxFrameHeader*

```
typedef struct {
   AiUInt8   uc_FrameType;
   TY_FDX_TX_FRAME_ATTRIB_EX x_FrameAttribEx;
   TY_FDX_TX_INSTR_ATTRIB x_InstrAttrib;
} TY_FDX_TX_FRAME_HEADER_EX;
```

```
typedef struct {
   AiUInt16  uw_FrameSize;
   AiUInt32  ul_InterFrameGap;
   AiUInt32  ul_PacketGroupWaitTime;
   AiUInt8   uc_PayloadBufferMode;
   AiUInt8   uc_TimetagInsertionMode;
   AiUInt32  ul_BufferQueueHandle;
   AiUInt8   uc_ExternalStrobe;
   AiUInt8   uc_PreambleCount;
   AiUInt32  ul_Skew;
   AiUInt8   uc_NetSelect;
   AiUInt8   uc_FrameStartMode;
   AiUInt32  ul_PhysErrorInjection;
   AiUInt16  uw_SequenceNumberInit;
   AiUInt16  uw_SequenceNumberOffset;
   AiUInt32  ul_EDEFlags;
   AiUInt32  ul_EDESourceId;
   AiUInt32  ul_EDEMessageSize;
   AiUInt32  ul_EDETsOffsetHigh;
   AiUInt32  ul_EDETsOffsetLow;
   AiUInt32  ul_EDESNOffset;
} TY_FDX_TX_FRAME_ATTRIB_EX;
```

Pointer to structure, which holds the Transmit Frame Header Information. See ***FdxCmdTxQueueWriteEx*** function. This structure is initialized as follows:

```
            x_FrameAttrib.uc_FrameType = FDX_TX_FRAME_STD;
            x_FrameAttrib.uc_NetSelect = FDX_TX_FRAME_BOTH;
            x_FrameAttrib.uc_ExternalStrobe = FDX_DIS;
            x_FrameAttrib.uc_FrameStartMode = FDX_TX_FRAME_START_IFG;
            x_FrameAttrib.uc_PayloadBufferMode = 0;
            x_FrameAttrib. uc_TimetagInsertionMode = FDX_TX_FRAME_TTI_DIS;
            x_FrameAttrib.uc_PreambleCount = FDX_TX_FRAME_PRE_DEF;
            x_FrameAttrib.ul_BufferQueueHandle = 0;
            x_FrameAttrib.ul_InterFrameGap = 25;           // (1us)
            x_FrameAttrib.ul_PacketGroupWaitTime = 1000;   // (1ms)
            x_FrameAttrib.ul_PhysErrorInjection = FDX_TX_FRAME_ERR_OFF;
            x_FrameAttrib.ul_Skew = 0;
            x_FrameAttrib.uw_FrameSize = 0;
            x_FrameAttrib.uw_SequenceNumberInit = FDX_TX_FRAME_SEQ_INIT_AUTO;
            x_FrameAttrib.uw_SequenceNumberOffset                            =
FDX_TX_FRAME_SEQ_OFFS_AUTO;
            x_FrameAttrib.ul_EDEFlags = FDX_TX_FRAME_EDE_ENA |
                                           FDX_TX_FRAME_EDE_FIRST |
                                           FDX_TX_FRAME_EDE_LAST;
            x_FrameAttrib.ul_EDESourceId = 0;
            x_FrameAttrib.ul_EDEMessageSize = 0;
            x_FrameAttrib.ul_EDETsOffsetHigh = 0;
            x_FrameAttrib.ul_EDETsOffsetLow = 0;
            x_FrameAttrib.ul_EDESNOffset = 1;
```

Output:

*TY_FDX_TX_FRAME_HEADER *px_TxFrameHeader*

Initialized structure (see above)

Return Value

Returns FDX_OK on success or a negative error code on error.

Error Codes: FDX_ERR

Output:

*TY_FDX_TX_FRAME_HEADER *px_TxFrameHeader*

Initialized structure (see above)

Return Value

Returns FDX_OK on success or a negative error code on error.

Error Codes: FDX_ERR

3.5.1 FdxCalcEDECRCx/FdxCalcEDECRCy

Prototype:

```
AiReturn FdxCalcEDECRCx( AiUInt32 ul_EDESourceId,
                         AiUInt32 ul_MessageLength,
                         void* pv_Message, AiUInt16* uw_CRC);


AiReturn FdxCalcEDECRCy( AiUInt32 ul_EDESourceId,
                         AiUInt32 ul_MessageLength,
                         void* pv_Message, AiUInt16* puw_CRC);
```

Driver Command

none

Purpose:

This function is used to calculate an EDE CRC given the EDE message (including EDE Sequence number and transmit timestamp) and the EDE source ID to be used for the calculation.

Input:

AiUInt32 ul_EDESourceId

The EDE source ID to be used as input for the calculation of the EDE CRC.

AiUInt32 ul_MessageLength

The length of the EDE message pointed to by **pv_Message.** This should include the EDE Sequence Number, transmit time stamp, and payload.

*void *pv_Message*

Pointer to the EDE message for which the CRC shall be calculated. The message should include the EDE Sequence Number, transmit time stamp, and payload.

Output:

AiUInt16 puw_CRC*

Pointer to the location in which the calculated EDE CRC shall be stored.

Return Value

Returns FDX_OK on success or a negative error code on error.

Error Codes: FDX_ERR

4. Notes

4.1 Abbreviations

EDE	Error Detection Encoding
CRC	Cyclic Redundancy Check
SN	Sequence Number
TS	Time Stamp

4.2 Definition of Terms

5. Document History

5.1 Modification Function List compared with previous versions

The following tables gives an overview of the compatibility rate between this and the previous versions of the Application Interface Library.

☺ No changes

😐 Changes (including prototype, DLL and/or documentation changes). If only documentation was changed a "**D**" is added

☹ Not supported any more / Removed

5.1.1 EDE Global Functions

Table 5.1-1: EDE Global Function Changes between several versions

Function	*➔V02.1.x*	*➔V02.0x*	*➔V01.0x*
FdxCmdReadEDECounter	☺	☺	!!NEW!!

5.1.2 EDE Subscriber Functions

Table 5.1-2: EDE Subscriber Functions Changes between several versions

Function	*➔V02.1.x*	*➔V02.0x*	*➔V01.0x*
FdxCmdEDESubCreate	☺	☺	!!NEW!!
FdxCmdEDESubCreateEx1	!!NEW!!		
FdxCmdEDESubClkControl	☺	☺	!!NEW!!
FdxCmdEDESubControl	☺	☺	!!NEW!!
FdxCmdEDESubControlEx	☺	!!NEW!!	
FdxCmdEDESubControlEx2	!!NEW!!		
FdxCmdEDESubTMDef	☺	☺	!!NEW!!
FdxCmdEDESubDestroy	☺	☺	!!NEW!!

5.1.3 EDE Transmitter Functions

Table 5.1-3: EDE Transmitter Functions Changes between several versions

Function	*➔V02.1.x*	*➔V02.0x*	*➔V01.0x*
FdxCmdTxQueueWriteEx	☺	☺	!!NEW!!
FdxCmdTxQueueWriteEde	☺	!!NEW!!	
FdxCmdTxEDECreatePort	☺	☺	!!NEW!!
FdxCmdTxEDEWrite	☺	☺	!!NEW!!

5.1.4 Receiver Functions

Table 5.1-4: Receiver Functions Changes between several versions

Function	*➔V02.1.x*	*➔V02.0x*	*➔V01.0x*
FdxCmdRxEDEVlControl	☺	☺	!!NEW!!
FdxCmdRxEDECreatePort	☺	☺	!!NEW!!
FdxCmdRxEDEControl	☺	!!NEW!!	
FdxCmdRxUDPRead	☺	😐	😐

5.1.5 Target Independent Administration Functions

Table 5.1-5: Target Independent Administration Functions Changes between several versions

Function	***➔V02.1.x***	***➔V02.0x***	***➔V01.0x***
FdxInitTxEDEFrameHeader	☺	☺	!!NEW!!
FdxCalcEDECRCx	☺	☺	!!NEW!!
FdxCalcEDECRCy	☺	☺	!!NEW!!

5.2 Document History Details

Table 5.2-1: Document History Details

Version	Date	Author	Description
01.0x Rev A	31.03.2006	T.Troshynski	- Creation of Document
01.0x Rev B	07.07.2006	T. Troshynski	- Clarifications made to descriptions of FdxCmdTxEDEWrite and FdxCmdRxUDPRead
01.0x Rev C	07.21.2006	T. Troshynski	- Added functions FdxCalcEDECRCx and FdxCalcEDECRCy - Added the possible error code EDE_CRCX_ERROR and EDE_CRCY_ERROR to FdxCmdRxUDPRead
01.0x Rev D	10.09.2006	T. Troshynski	- Updated ul_Skew and ul_InterframeGap descriptions in FdxCmdTxQueueWriteEx
02.0x Rev A	15.11.2006	T. Troshynski	- Updated the description FdxCmdRxUDPRead to add the new possible error codes for messages that fail EDE Age and SN verification - Added FdxCmdEDESubControlEx - Added FdxCmdRxEDEControl
02.0x Rev B	29.01.2007	T. Troshynski	- Added additional error codes to FdxCmdRxUDPRead for EDE messages.
02.0x Rev C	21.03.2007	T. Troshynski	- Updated FdxCmdEDESubControlEx to reflect different units used for the timing and timeout input parameters. - Updated FdxCmdRxEDControl to include new input parameter in support of a programmable RM_SN_Invld_Window.
02.0x Rev D	01.10.2007	R. Heitzmann	- Updated FdxCmdTxQueueWriteEx correct description of ul_EDEMessageSize. - New Function FdxCmdTxQueueWriteEde to have same function extentions like in standard function.
02.1.x Rev C [2]	05.12.2017	R. Heitzmann	- New Functions FdxCmdEDESubCreateEx1 and FdxCmdEDESubControlEx2 to cover new requirements of Boeing specification Rev. E [5] - New Cover sheet.
02.1.x Rev D [3]	08.03.2018	R. Heitzmann	- Extend Functions FdxCmdEDESubCreateEx1 to have variable Message size

[2] Rev. A and Rev. B were preliminarry distributet without updated history details.

[3] Rev. A and Rev. B were preliminarry distributet without updated history details.