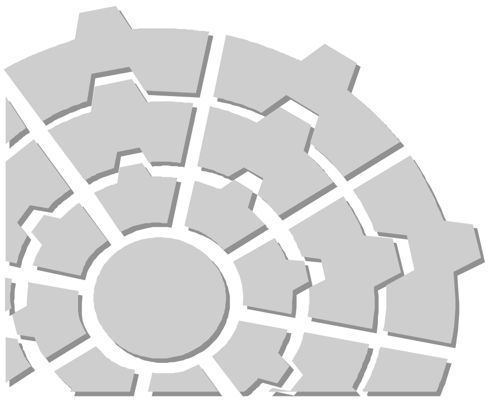
DeltaLink DECODER
Decoder software for FactoryLink

User Manual



Version 6

Printed: 4 June, 1997

Table of Contents

This page is left blank intentionally.

1. Introduction

Thank you for buying the decoder! We hope you will enjoy using this product.

1.1. Scope of this document

This manual is written for a technician who is familiar with the FactoryLink® IV software. This document can be used both as a training manual as well as a reference manual.

Note: *Please check the contents of the shipment with the list as described in the next chapter.*

The first section of this manual deals with the installation of hardware and software in your FactoryLink workstation. This part is split into a platform independent and a platform specific part. Please read carefully through both parts to make sure both hardware and software are installed correctly.

The second part explains the operation principles of the decoder and the communication with an external device (for example a PLC). Here all terms and definitions are explained to the reader. It explains terms like "Dataset" and "Read/Write'. This part should be read by both the PLC programmer and the FactoryLink programmer to make sure that the optimum performance can be achieved.

The third part explains the exact tables associated with the decoder. This part is useful only to FactoryLink programmers and can be used as a reference. This part is also an example of how to use the decoder with a protocol driver. It shows the entries made for the pre-configured demo which comes with a protocol driver package. The demo program can be used to check if the communication is working without making a complete application.

The last part are the Appendices which contain summarized data.

This page is left blank intentionally.

2. Contents of shipment

Please check the package you received with the checklist below. Should there be an item missing contact DeltaLink to correct the problem. There is a limit of 90 days after shipment to report problems!

This package includes the following:

1 diskette labelled "DeltaLink Decoder"

Files:	\AC\DECODER.AC	
	\AC\DECODER.H	
	\BIN\DECODER.EXE	(W95 - OS2 - NTI)
	\BIN\DECODER.ACR	
	\BIN\DECODER	(UNIX)
	\CTGEN\DECODER.CTG	
	\HELP\DE\DECODER.HLP	
	\HELP\EN\DECODER.HLP	
	\HELP\FR\DECODER.HLP	
	\KEY\DE\DEC_CNV.KEY	
	\KEY\DE\DEC_WR.KEY	
	\KEY\EN\DEC_CNV.KEY	
	\KEY\EN\DEC_WR.KEY	
	\KEY\FR\DEC_CNV.KEY	
	\KEY\FR\DEC_WR.KEY	
	\MSG\DE\DEC_AC.TXT	
	\MSG\DE\DECODER.TXT	
	\MSG\EN\DEC_AC.TXT	
	\MSG\EN\DECODER.TXT	
	\MSG\FR\DEC_AC.TXT	
	\MSG\FR\DECODER.TXT	
	\OPT\DELTA.OPT	
	\INSTALL.BAT	(W95 - NTI)
	\INSTALL.CMD	(OS2)
	\INSTALL	(UNIX)
	\INST_SEQ.EXE	(W95 - OS2 - NTI)
	\INST_SEQ	(UNIX)
	\FLBUILD.ID	
	\FLXMEDIA	(W95 - OS2 - NTI)
	\UPDATE.EXE	(W95 - OS2 - NTI)
	\UPDATE	(UNIX)
	\DECODER.$$$	

❷ 1 DeltaLink hardware key or authorisation sequence containing DECODER option.

❸ This manual (Which seems to be present).

You should also have:

☑ A IMX based protocol driver e.g. DeltaLink Sinec H1 protocol driver.

☑ The correct hardware and software for a communications card in the FactoryLink workstation.

This page is left blank intentionally.

3. Installation

3.1. Installation of the FactoryLink software

To install the FactoryLink task and its related tables please follow the following steps.

Before installing

Before installing the decoder on the system FactoryLink must have been installed error free. It is very important that all the environment settings are made for the FactoryLink system such as the *FLINK*, *FLOPT* etc.

First:

Copy the files from the "DeltaLink Decoder" media to the appropriate directory. This will be automatically done by running the install utility placed on the installation media. The installation procedure differs for UNIX and W95-OS/2-NTI systems.

For W95-OS/2-NTI systems follow the next procedure:

```
a:\↵
install↵
```

For UNIX systems first an install directory must be created. The files on the install floppy must be first copied to the install directory using the *tar* command. From this directory the install utility can be run.

For UNIX systems follow the next procedure:

```
mkdir install
cd install
tar xv↵
install↵
```

Second:

After you installed the software you need to activate the tables in the FactoryLink Configuration Manager (FLCM). The installation automatically appends the *decoder.ac* entry into the *{FLINK}/AC/titles* file[1]. The place of this entry is also the place where the option appears in the FLCM Main Menu. Therefore check the validity of the entry and move it to the place where you want to appear it in the Configuration manager. The entry must match the decoder entry in the following table:

```
file: {FLINK}/ac/titles:

...
persist.ac
spool.ac
decoder.ac
sinec_h1.ac
...
```

Third:

[1]{FLINK} is the working directory for the FactoryLink programs.

To make sure all the Configuration Tables (CT's) are generated after a change, the install utility automatically adds the decoder entry at the end of the *{FLINK}/ctgen/ctlist* file. The place of this entry is not important. Check if this entry has the same format as in the next table:

```
file: {FLINK}\ctgen\ctlist:

...
rp: rptovr rpthdr
sinec_h1: sinech1m sinech1x sinech1p sinech1d
decoder: decoderm decoderp decoderd decodert
timer: itimer etimer
...
```

Fourth:

To enable the help functionality for the Decoder tables in the Configuration Manager, the installation utility reindexes the ‘help-index’ for the Configuration Manager. If desired reindexing of the ‘help-index’ can be started from the command line prompt.

```
mkhelp↵
```

Fifth:

The FactoryLink Configuration Manager uses a map file, {FLINK}/ac/ac2ct.map, to be aware of the different configuration tables which can be located behind one entry in the main menu. Upon startup, the Configuration Manager reads the map file instead of all the table configuration files, mainly because reading all these files at once takes too much time. An account manger is present to update this conversion file, and can be started from the command line.

```
acctmgr  -c -d -t{FLINK}/ac/titles↵
```

Sixth:

The decoder must, with the FactoryLink Configuration Manager (FLCM), be entered in the System Configuration table. An entry of an existing task which will not be used at run-time can be overwritten or a new entry can be created with (as a minimum) the following data:

Task Name	*Description ....*	*Executable File*
DECODER	***Basic Decoder***	***bin/decoder***

The Task Name and name of the executable file are fixed and should not be altered by the user.

This completes the installation of the FactoryLink (software) parts.

3.2. Installation of the protection

The decoder is been protected via the DeltaLink option file. This file contains authorization sequence codes for DeltaLink modules. The protection is linked to the serial number of the FactoryLink package.

3.2.1. The DeltaLink option file

The installation media of the decoder contains an option file, named ‘delta.opt’, in the 'opt' directory. This file contains the unique authorization sequence which enables the decoder to run. The install utility automatically copies the authorization sequence into the {FLOPT}/delta.opt file. It is also possible to enter the authorization sequence manually into the {FLOPT}/delta.opt file. For more information on the delta.opt file refer to ***Appendix A***.

Note that the task will only run on the FactoryLink system with the same serial number. The *delta.opt* file on the installation diskette contains, for reference, the serial number of FactoryLink.

3.2.2. Demo installation

It is possible to install the decoder without an authorization code. This will be done from a normal installation media. In this case the task will start up but only runs in so called ‘demo’ mode. This means that the decoder runs only for a limited period of time (five hours). After this period has expired the task will shutdown and can not be restarted before the complete FactoryLink system has been restarted.

After installation of this demo version an authorization code can be ordered and installed which enables the task to run without the time and restart limitation. The authorization code must be entered manually in the {FLOPT}/delta.opt file. For more information on entering the authorization code in the delta.opt file refer to ***Appendix A***.

The limitations of a demo version of the Decoder are:

- five hours of consecutive run-time
- not restartable (FactoryLink must be restarted)

This page is left blank intentionally.

4. Principle

4.1. The decoder RAP Ddriver principle

RAPD stands for Rapid Application Protocol Driver. The RAPD principle was adopted so that protocol drivers can be easily and rapidly configured for a FactoryLink application. RAPD is based on the Intertask Mail Exchange Standard or IMX, which defines a way for a protocol driver task to communicate with an I/O Translator task (e.g. Decoder, high speed logger). The RAPD system consists of a protocol driver which communicates with external devices (RTUS, PLC's, etc.) and a translator which controls data storage (going to and coming from a protocol driver) in the FactoryLink real-time database. All data collected by the protocol driver is referenced as contiguous blocks or ranges within the device. This enables communications between the driver and a device to be very efficient. All data is referenced between the driver and the translator in terms of datasets. Datasets, described in the next section, define memory regions or locations of data within a device.
The protocol driver and the IOX communicate with one another via FactoryLink mailbox tags, according to the IMX standard. Every task (translator and protocol drivers) has its own mailbox, so for full communication between a translator and a protocol driver a mailbox database element for every task has to be defined. The IMX standard is especially designed for the following situation. To use <u>one</u> decoder and <u>several</u> protocol drivers. For example the decoder together with the Sinec H1 protocol driver <u>and</u> the Modbus Plus protocol driver. Aside from storage duties, the translator provides data conversions (i.e. analog, IEEE conversions, etc.) for I/O data to/from a protocol driver.

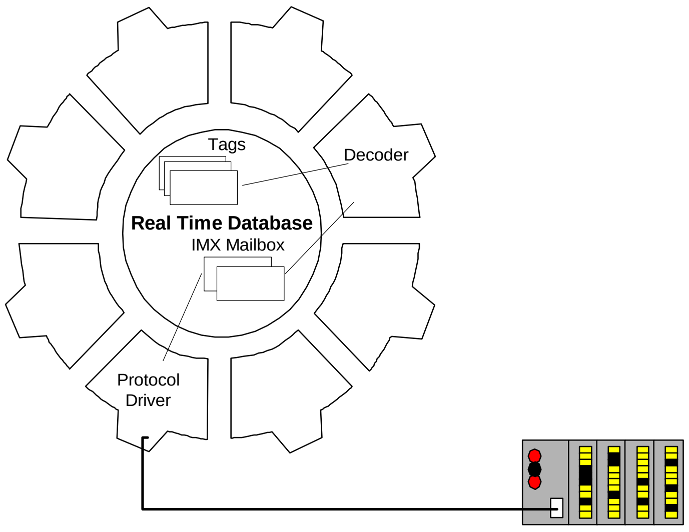


Figure 4.1.1 The RAPD principle.

4.2. Datasets

A dataset is a (contiguous) area (register, memory location, etc.) of data in the external device. All datasets are defined as digital tags, the dataset control tag. The protocol driver defines all specifics (data type, starting address, length, etc.) about datasets while the translator merely references the dataset by tag name for read and write operations.
When the translator initiates a trigger on a dataset, the IMX system will notify the protocol driver that an event has occurred on that dataset. The event notification will in turn cause a protocol specific message to be generated and sent to an external device. Similarly, the scenario works in reverse when the protocol driver receives unsolicited data from a device.

Dataset		External device
Referenced by dataset control tag		Memory location, defined by area type, start address and length
Tag 1	$\longleftrightarrow$	Address: n
Tag 2	$\longleftrightarrow$	Address: n + 1
Tag 3	$\longleftrightarrow$	Address: n +2
Tag 4	$\longleftrightarrow$	Address: n + 3
Tag x	$\longleftrightarrow$	Address: n + m

Figure 4.2.1 Mapping between dataset and tags.

A dataset can be read or written to with one command, this is a trigger. Except actions on the complete dataset, most protocol drivers are capable of addressing specific elements (or a group of elements) in the dataset. How these actions are performed depends on the size of a basic element in the dataset, see the next chapter. The size of a basic-element of a dataset can be bit, byte, word or long, depending on the data area in the device or the communication protocol.

4.3. External device communication

4.3.1. Communication protocol

Data exchange between a FactoryLink workstation and an external device is performed by using the *read* and *write* functions of the device's communication protocol. The protocol driver task uses the *read* function to read (or fetch) data from the device, and it uses the *write* function to write data to the device. Both actions are initiated by the combination decoder/driver. A device may also, in an unsolicited fashion, send data to the FactoryLink workstation.

4.3.2. Read and block write

The read and block write commands are performed on a contiguous block of data in the external device specified by datasets. Both commands are initiated by the FactoryLink workstation. The developer can configure digital triggers for the decoder task to start the read/write commands. Data received from a device as a result of a read request ,is stored in FactoryLink tags. For a write request, data is taken out of FactoryLink tags and sent to the device.

4.3.3. Unsolicited receive

For an unsolicited receive the device sends a dataset to the FactoryLink workstation, though the workstation does not originate the request to do so. The decoder task places the data received from the device into FactoryLink tags.

4.3.4. Exception write

An exception write is initiated by the FactoryLink workstation, and causes a write of data to the external device. Exception writes are most often used when a tag value has changed within the real-time database. The protocol driver will receive a request for an exception write from the decoder. The first action the driver will take is to check if the write can be accomplished with one write operation or if the data must be read first then written (results in more than one operation).
In situations where the exception data element size is smaller than the size of the device data type element, the data area (i.e. register) must be read first. Then the data to be written out must be masked or ORed into the value just read. Finally, the new value is written back to the device. If this is not done, data will be unintentionally overwritten in the device.
To illustrate an exception write: if a bit in a device has to be set and the boundary of the data area is a 16 bit word (implies word addressing), then the specific data word has to be read first because the smallest element in this device data area is a word. The bit has to then be patched into the word and the word is then written back to the device.
In the illustration just mentioned, it is possible that the device may change the word value during the read-before-write operation. In this case, once the driver writes the new word value, elements of the last word change will have been lost.
To avoid the latency problem associated with read-before-write operations, an encoded write is an alternative option. An encoded write can perform a write operation in one request instead of the two or more (depending on the protocol) requests generated by a read-before-write operation. The encoded write is discussed in the next section.

4.3.5. Encoded write

The encoded write function is almost similar to the exception write in the regard that both functions write single data elements. The difference is that the exception write directly accesses the desired data area in the external device and that data are is changed. On the other hand, the encoded write composes an encoded write command (in reality it is simply a write command). The encoded write command is nothing more than a protocol message to the device referencing a certain memory location. A device program (i.e. ladder logic algorithm in a PLC) running within the device would

detect that the memory location has been written to. This will cause the device program to then make the change to a single data element.
The advantage of this method is that only one write command is generated from the protocol driver and the device's internal program does the actual operation on the data element (no reading before writing). Another advantage is that the device program can control all encoded writes because they all are written to one location in the device.

4.4. FactoryLink domain selection

The standard domain for a protocol driver is the SHARED domain. The protocol driver communicates with a dedicated piece of hardware, there fore only one task should be able to access the hardware. If only one program accesses the hardware, the task should be started by the shared runtime-manager.
Important: The protocol driver and the decoder must be in the same domain (either SHARED or USED).

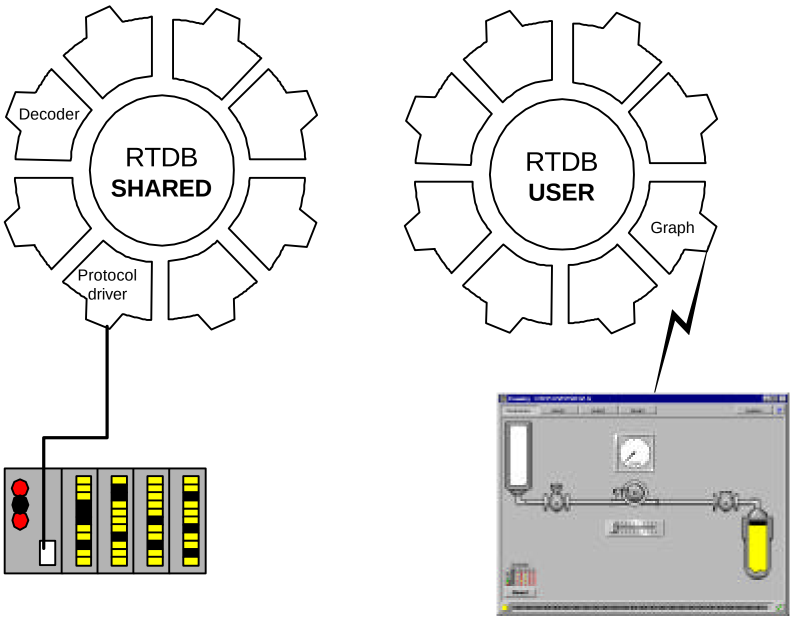


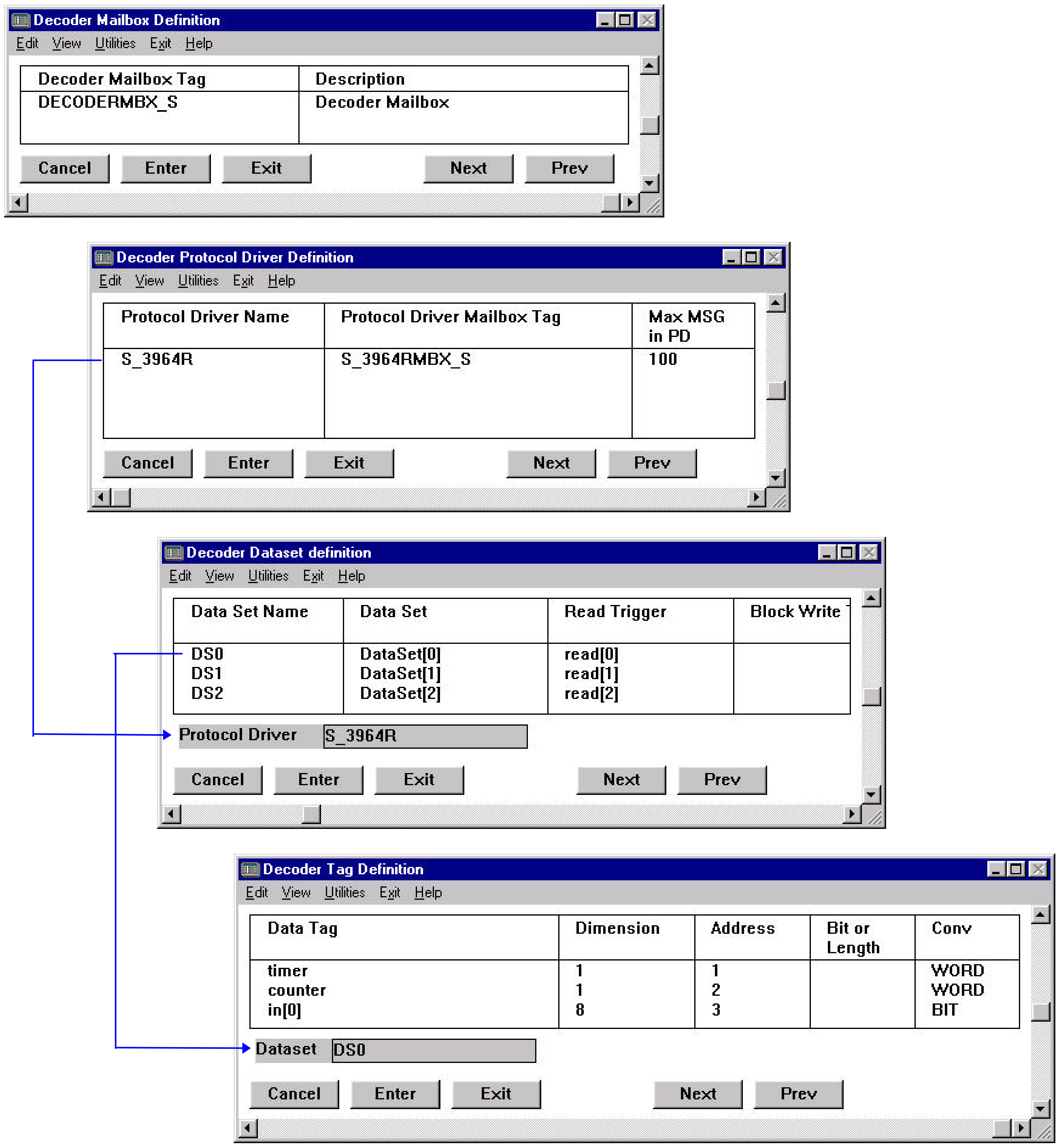
Figure 4.4.1: Standard domain selection.

This page is left blank intentionally.

5. Configuration tables

In the Configuration Manager Main Menu, select **Decoder**. Four tables appear, with the titles of all panels visible for direct access. To access a specific panel position the cursor on a visible area and press the left mouse-button, or use the Next/Prev buttons.

Note: For general information about entering data in FactoryLink configuration tables, refer to the FactoryLink Fundamentals Manual.


Decoder Mailbox Definition
Decoder Protocol Driver Definition
Decoder Dataset definition
Decoder Tag Definition
Edit View Utilities Exit Help
Cancel Enter Exit Next Prev

Decoder Mailbox Definition

Decoder Mailbox Tag	Description
DECODERMBX_S	Decoder Mailbox

Decoder Protocol Driver Definition

Protocol Driver Name	Protocol Driver Mailbox Tag	Max MSG in PD
S_3964R	S_3964RMBX_S	100

Decoder Dataset definition

Data Set Name	Data Set	Read Trigger	Block Write
DS0	DataSet[0]	read[0]	
DS1	DataSet[1]	read[1]	
DS2	DataSet[2]	read[2]	

Protocol Driver: S_3964R

Decoder Tag Definition

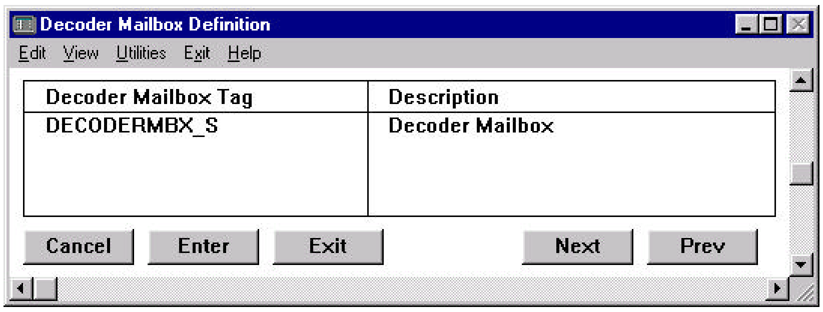
Data Tag	Dimension	Address	Bit or Length	Conv
timer	1	1		WORD
counter	1	2		WORD
in[0]	8	3		BIT

Dataset: DS0

Figure 5.0.1 Decoder configuration panels.

5.1. Mailbox definition

From the display of all the panels, select the *Decoder mailbox definition* panel.



Decoder Mailbox Tag	Description
DECODERMBX_S	Decoder Mailbox

Figure 5.1.1 Decoder Mailbox definition panel.

The Decoder Mailbox Definition panel allows the user to initialize one mailbox tag for the decoder task. Only one Mailbox is needed for to enable IMX. Specify the following information.

- **Decoder Mailbox Tag**
  Tag name of the decoder mailbox element that the application developer defines, to be referenced by a protocol driver (**N.B.** IMX must be supported by the protocol driver task). The decoder task uses this mailbox to receive requests from a protocol driver.

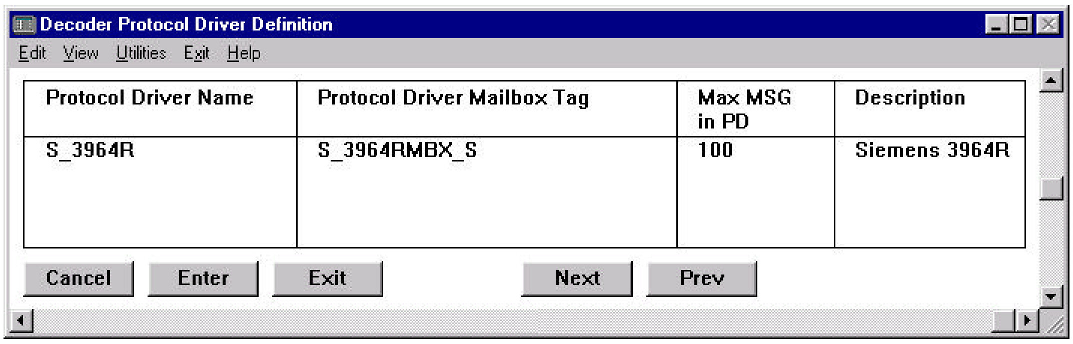
entry:	Required.
entry type:	Standard FactoryLink tag name.
valid entry:	MAILBOX.

- **Description**
  Description of the decoder mailbox element defined by the application developer.

valid entry:	Output only.

5.2. Protocol driver definition

From the display of all the panels, select the *Protocol Driver Definition* panel.


Decoder Protocol Driver Definition
Edit View Utilities Exit Help
Cancel
Enter
Exit
Next
Prev

Protocol Driver Name	Protocol Driver Mailbox Tag	Max MSG in PD	Description
S_3964R	S_3964RMBX_S	100	Siemens 3964R

Figure 5.2.1 DeltaLink Protocol Driver definition.

The Protocol Driver Definition panel allows the user to specify one or more protocol driver(s). Every protocol driver is associated with a mailbox tag. Only one Mailbox Tag, for every protocol driver, is needed to enable IMX. Specify the following information.

- **Protocol Driver Name**
  Logical name assigned by the application developer for a particular protocol driver. The field is used as a selection criteria for the next table, Decoder Dataset Definition.

entry:	Required.
entry type:	Alphanumeric string.
valid entry:	String of up to 16 characters.

- **Protocol Driver Mailbox Tag**
  Tag name of a protocol driver mailbox element, to be referenced by the protocol driver task (**N.B.** IMX must be supported by the protocol driver). The decoder task uses this mailbox to send data to a protocol driver.

entry:	Required.
entry type:	Standard FactoryLink tag name.
valid entry:	MAILBOX.

- **Max MSG in PD**
  The maximum number of requests for the protocol driver, which can be queued in the mailbox tag. A value of 100 messages is recommended. The number of messages is limited by the size of an integer value (9999), but more importantly by the amount of available memory. The memory needed for a request depends on the size of the configured datasets.

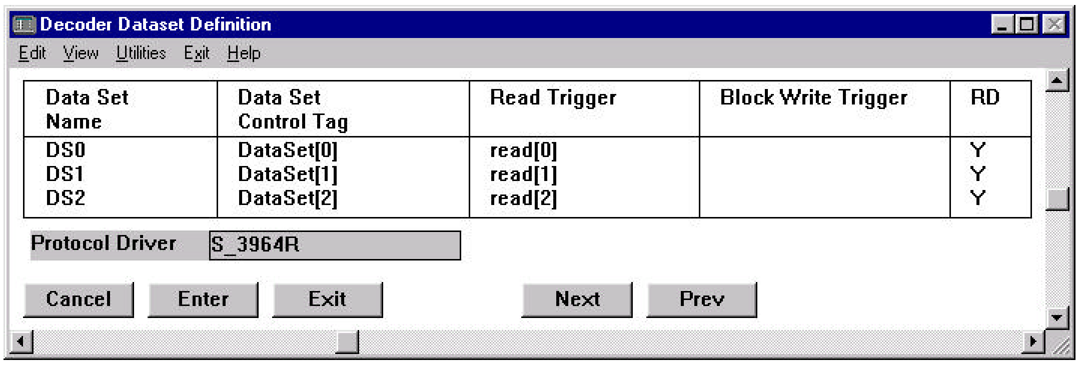
entry:	Required / Default: 100.
entry type:	Decimal number.
valid entry:	1 .. 9999.

- **Description**
  Description of the protocol driver mailbox element defined by the application developer.

valid entry:	Output only.

5.3. Dataset definition

From the display of all the panels, select the *Decoder Dataset definition* panel.



Data Set Name	Data Set Control Tag	Read Trigger	Block Write Trigger	RD
DS0	DataSet[0]	read[0]		Y
DS1	DataSet[1]	read[1]		Y
DS2	DataSet[2]	read[2]		Y

Protocol Driver: S_3964R

Figure 5.3.1 Decoder dataset definition.

The Decoder Dataset Definition panel allows the user to specify datasets for the selected protocol driver. The name of the protocol driver is displayed in the field *Protocol Driver*, at the bottom of the panel.

♦ **Data Set Name**
Logical name assigned by the application developer for a particular dataset. The field is used as a selection criteria for the next table, Decoder Tag Definition.

entry:	Required.
entry type:	Alphanumeric string.
valid entry:	String of up to 16 characters.

♦ **Data Set Control Tag**
Tag name representing a (unique) logical name for a data set. The tag is used internally in the decoder and protocol driver for activating a data fetch/write from/to the external device. A dataset is referenced, internally by the decoder and the protocol driver, by the tag name. Therefore the tag specified in this field must be unique.

entry:	Required.
entry type:	Standard FactoryLink tag name.
valid entry:	DIGITAL.

♦ **Read Trigger**
Real-time database element set by the user to trigger a read-command for the specified protocol driver on the dataset. The read-command is performed in case the database element has the value ON and the change flag is ON. If there is no tag specified and the read command is enabled (see *RD* entry) the dataset is read continuously.

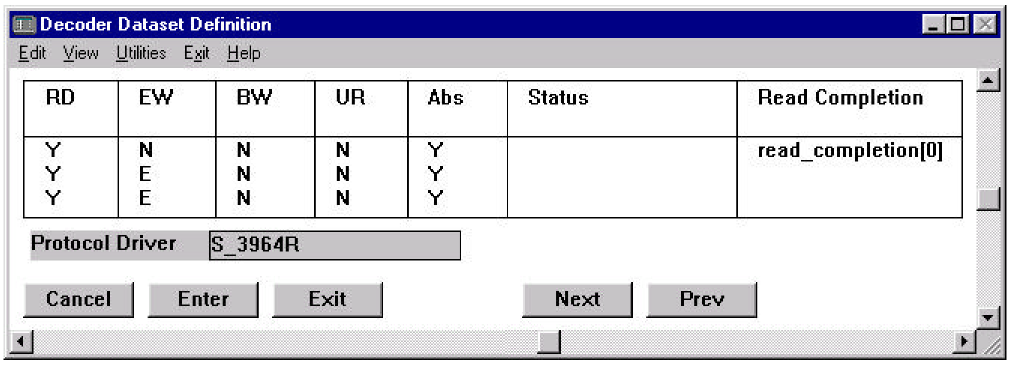
entry:	Optional.
entry type:	Standard FactoryLink tag name.
valid entry:	DIGITAL.

¨ **Block Write Trigger**
Real-time database element set by the user to trigger a block write-command for the specified protocol driver on the dataset. The write-command is performed in case the database element has the value ON and the change flag is ON.

entry:	Optional.
entry type:	Standard FactoryLink tag name.
valid entry:	DIGITAL.

- **RD**
  Enable/disable block read commands. For the entry N(o) the block read commands are disabled, although a *Read Trigger* can be specified no block reads will be performed. To enable block reads on this dataset the entry must be Y(es).

entry:	Required / Default: N.
entry type:	Alphanumeric string.
valid entries:	N, Y.



RD	EW	BW	UR	Abs	Status	Read Completion
Y	N	N	N	Y		read_completion[0]
Y	E	N	N	Y		
Y	E	N	N	Y		

Protocol Driver: S_3964R

Figure 5.3.2 Decoder dataset definition.

- **EW**
  Define the type of exception write. Three types of exception writes are possible: None (disabled), Exception write and enCoded write. For a specific discussion of the different write types, see the manual for the protocol driver.

entry:	Required / Default: N.
entry type:	Alphanumeric string.
valid entries:	N, E, C.
description:	

Exception write	Description
N	None, exception writes disabled
E	Exception, normal exception writes
C	enCoded, encoded exception writes

- **BW**
  Enable/disable block write commands. For the entry N(o) the block write commands are disabled, although a *Block Write Trigger* can be specified no block writes will be performed. To enable block writes on this dataset the entry must be Y(es).

entry:	Required / Default: N.
entry type:	Alphanumeric string.
valid entries:	N, Y.

- **UR**
  Enable/disable unsolicited receive commands. For the entry N(o) unsolicited received data will be ignored. To enable the reception of unsolicited data on this dataset the entry must be Y(es). For a specific discussion of the unsolicited receive refer to the manual of the protocol driver.

entry:	Required / Default: N.
entry type:	Alphanumeric string.
valid entries:	N, Y.

- **Abs**
  Define the address coding in a dataset. Absolute address coding means that data elements in a dataset are addressed according to the definition of the data-area in the external device (to

enable this type of addressing the entry must be N(o)). For relative addressing (absolute addressing disabled, entry: Y(es)) the address of the first element in a dataset is zero (0) based. Successive addresses are found by incrementing by one (1).

entry:	Required / Default: N.
entry type:	Alphanumeric string.
valid entries:	N, Y.

- **Status**

Real-time database element used to indicate the status of the last executed command, read, write or exception. This element, if entered, is updated by the decoder task just before a completion trigger is set (see the next three entries). As IMX allows routing of error coded from a protocol driver to the decoder, the decoder is aware of the result of a command.

entry:	Optional.
entry type:	Standard FactoryLink tag name.
valid entry:	ANALOG.

- **Read Completion**

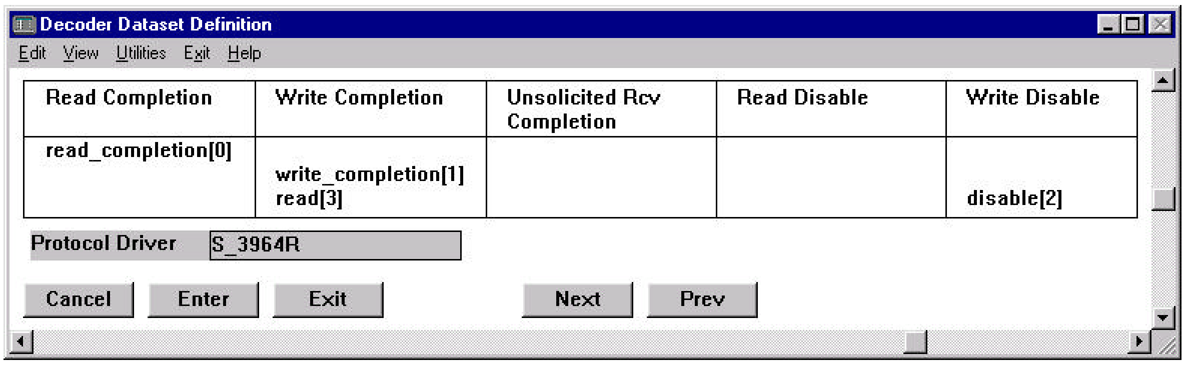
Real-time database element used to indicate that a read command has completed. This element, if entered, is forced ON (on completion of the command) by the decoder and can be referenced by any other FactoryLink task.

entry:	Optional.
entry type:	Standard FactoryLink tag name.
valid entry:	DIGITAL.

- **Write Completion**

Real-time database element used to indicate that a write command has completed. This element, if entered, is forced ON (on completion of the command) by the decoder and can be referenced by any other FactoryLink task.

entry:	Optional.
entry type:	Standard FactoryLink tag name.
valid entry:	DIGITAL.



Read Completion	Write Completion	Unsolicited Rcv Completion	Read Disable	Write Disable
read_completion[0]	write_completion[1] read[3]			disable[2]

Protocol Driver: S_3964R

Figure 5.3.3 Decoder Dataset definition.

- **Receive Completion**

Real-time database element used to indicate that an unsolicited receive command is completed. This element, if entered, is forced ON (on completion of the command) by the decoder and can be referenced by any other FactoryLink task.

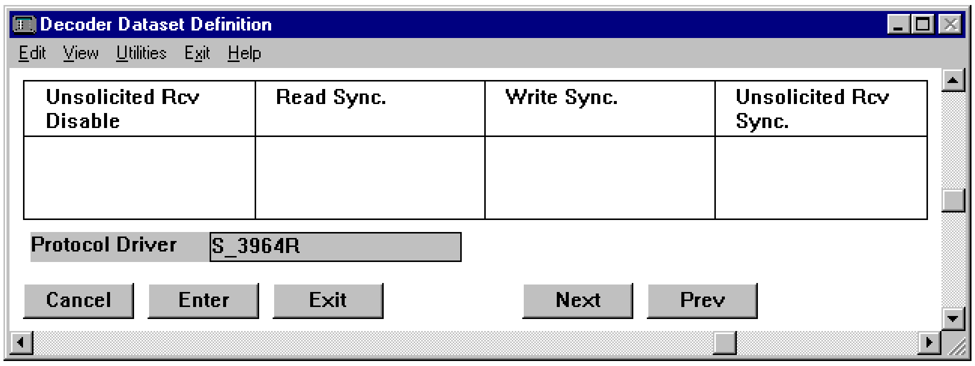
entry:	Optional.
entry type:	Standard FactoryLink tag name.
valid entry:	DIGITAL.

- **Read Disable**
  Real-time digital database element used to enable/disable read commands from the external device for this particular dataset. Read commands are enabled in case there is no tag defined, or the status of the digital tag is OFF. Read commands are disabled if the status of the tag is ON.

entry:	Optional.
entry type:	Standard FactoryLink tag name.
valid entry:	DIGITAL.

- **Write Disable**
  Real-time digital database element used to enable/disable write commands to the external device for this particular dataset. Write commands are enabled in case there is no tag defined, or the status of the digital tag is OFF. Write commands are disabled if the status of the tag is ON.

entry:	Optional.
entry type:	Standard FactoryLink tag name.
valid entry:	DIGITAL.



Unsolicited Rcv Disable	Read Sync.	Write Sync.	Unsolicited Rcv Sync.

Protocol Driver: S_3964R

Figure 5.3.4 Decoder Dataset definition.

- **Unsolicited Rcv Disable**
  Real-time digital database element used to enable/disable unsolicited receive commands from the external device. Receive commands are enabled in case there is no tag defined, or the status of the digital tag is OFF. Receive commands are disabled if the status of the tag is ON.

entry:	Optional.
entry type:	Standard FactoryLink tag name.
valid entry:	DIGITAL.

- **Read Sync.**
  Real-time database element to enable synchronization of read commands. All received data from a protocol driver is placed on (data) tags if there is no entry in this field or if the elements value is zero (OFF for a digital element). Whenever the decoder receives data for a read request the tag is set to a value not equal to zero. For a digital tag the value is ON, for an analog tag the value is 1 (one) indicating the data is the result from a read request. Any other task can write to this tag and set the value to zero, allowing the decoder to update the tags in this dataset with new values. **N.B.** In case the protocol driver fails to fetch the data, an error message is send to the decoder and the sync. tag will also be set to a value not equal to zero.

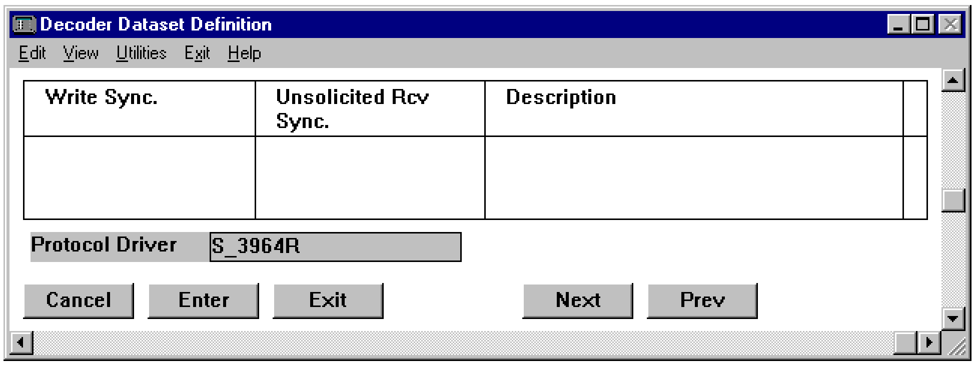
entry:	Optional.
entry type:	Standard FactoryLink tag name.
valid entry:	DIGITAL, ANALOG.

- **Write Sync.**
  Real-time database element to enable synchronization of write command completion's. A write completion (received from a protocol driver) can generate a completion and/or an update of the status tag, these tag(s) are updated if there is no entry in this field or if the elements value is zero (OFF for a digital element). Whenever the decoder receives a write completion the tag is set to a value not equal to zero. For a digital tag the value is ON, for an analog tag the value is 2 (two) indicating the data is the result from a write completion. Any other task can write to this tag and set the value to zero, allowing the decoder to update the completion and/or status tag for this dataset.

entry:	Optional.
entry type:	Standard FactoryLink tag name.
valid entry:	DIGITAL, ANALOG.

- **Unsolicited Rcv Sync.**
  Real-time database element to enable synchronization of unsolicited received data. All received data from a protocol driver is placed on (data) tags if there is no entry in this field or if the elements value is zero (OFF for a digital element). Whenever the decoder receives unsolicited data the tag is set to value not equal to zero. For a digital tag the value is ON, for an analog tag the value is 3 (three) indicating the data is the result from an unsolicited receive. Any other task can write to this tag and set the value to zero, allowing the decoder to update the tags in this dataset with new values. **N.B.** In case the protocol driver fails to fetch the data, an error message is send to the decoder and the sync. tag will also be set to a value not equal to zero.

entry:	Optional.
entry type:	Standard FactoryLink tag name.
valid entry:	DIGITAL, ANALOG.



Write Sync.	Unsolicited Rcv Sync.	Description

Protocol Driver: S_3964R

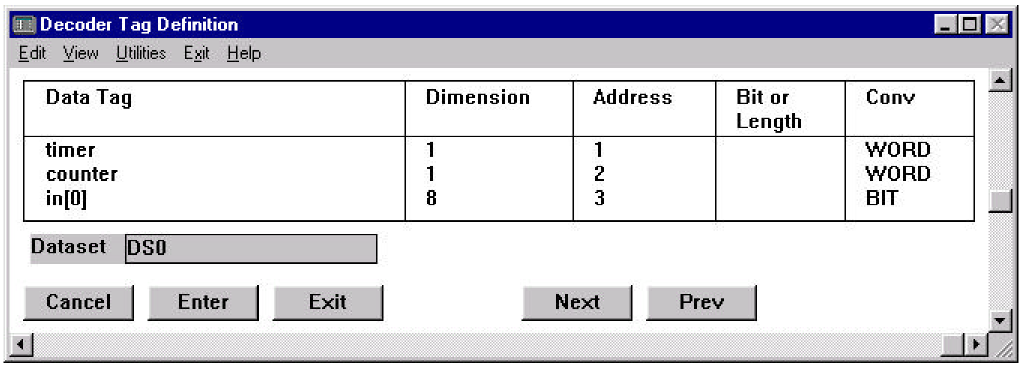
Figure 5.3.5 Decoder Dataset definition.

- **Description**
  This is the description of the dataset control tag (entered when the tag is defined), see the first column of this panel and is for informative purpose only.

valid entry:	Output only.

5.4. Tag definition

From the display of all the panels, select the *Decoder Tag Definition* panel.



Data Tag	Dimension	Address	Bit or Length	Conv
timer	1	1		WORD
counter	1	2		WORD
in[0]	8	3		BIT

Dataset: DS0

Figure 5.4.1 Decoder Tag Definition.

The Decoder Tag Definition panel allows the user to specify the link between datasets and the external device data area. The name of the dataset is displayed in the field *Dataset,* at the bottom of the panel. Specify the following information.

- **Data Tag**
  Real-time database element from or to which the external device data is to be transferred. The name of the dataset is displayed in the field *Dataset*. The decoder task can work in conjunction with the USDATA scaling task, for a detailed description see the next chapter. Remember that the here specified element is replaced (internally in the task) by the raw scaling tag if an entry is found in the panel of the scaling task!

entry:	Required.
entry type:	Standard FactoryLink tag name.
valid entry:	DIGITAL, ANALOG, LONGANA, FLOAT, MESSAGE.

- **Dimension**
  Tag array length of the *Data Tag* as it should be used by the decoder. The decoder allows tag arrays to be defined by the application developer, with the same conversion but with ascending address numbers.

entry:	Required / Default: 1.
entry type:	Decimal number.
valid entry:	1 .. 65535.

- **Address**
  Number specifying the address of a data element in the dataset. The addressing can be relative or absolute. Relative addressing means that the first element of a dataset has address zero, the next address one and so on. For absolute addressing the address numbers are equal to the addresses in the external device. The selection for relative/absolute addressing for a dataset is made in the *Decoder Dataset Definition* table.

entry:	Required / Default: 1.
entry type:	Decimal number.
valid entry:	1 .. 65535.

- **Bit or Length**
  The entry made in this field is used (by the decoder task) in conjunction with the next entry *Conv*. The number entered here represents a bit number for conversion BIT. For the conversion MSG (message, string) the number represents the length of the converted text string. For any other conversion then the ones mentioned above, the value entered will not be evaluated by the decoder task.

entry:	Optional / Default: 0.
entry type:	decimal number.
valid entry:	0 .. 9999.

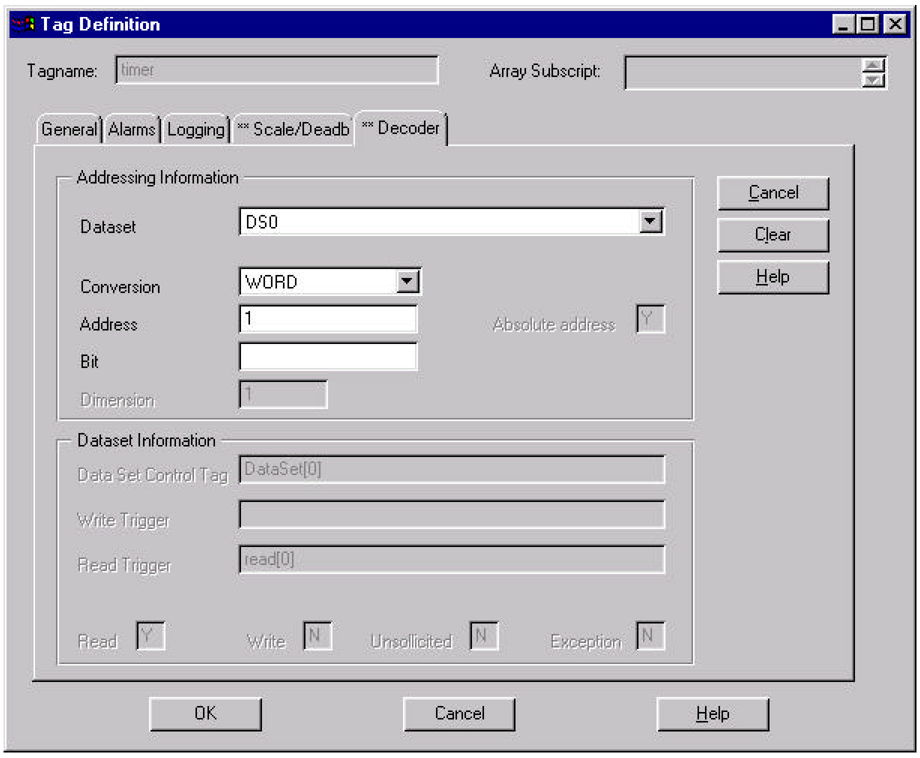
- **Conv**

The value of a dataset element is converted according to the entry made here, before the value is written to the tag. The tag is defined in the column *Data Tag*. Vice versa is the value of the tag converted before writing to the dataset. A few possible conversions are: BIT (BIT conversion), BR (Byte Right conversion), BL (Byte Left conversion), WORD (WORD conversion), MSG (MeSsaGe conversion), SIE (SIEmens float conversion). For a full list of possible conversions refer to *Appendix E*.

entry:	Required / Default: WORD..
entry type:	Alphanumeric string.
valid entries:	See *Appendix E*.

6. Single point tag configuration

Single point tag configuration is launched from within the Application editor. After installation of the decoder an extra ‘tab’, decoder, is added to the single point tag configuration panel. Here you can specify the same information as for the table *Decoder Tag Definition* in the configuration manager.


Tag Definition

Tagname: timer

Array Subscript:

Addressing Information

Dataset: DS0

Conversion: WORD

Address: 1

Bit:

Dimension: 1

Absolute address: Y

Dataset Information

Data Set Control Tag: DataSet[0]

Write Trigger:

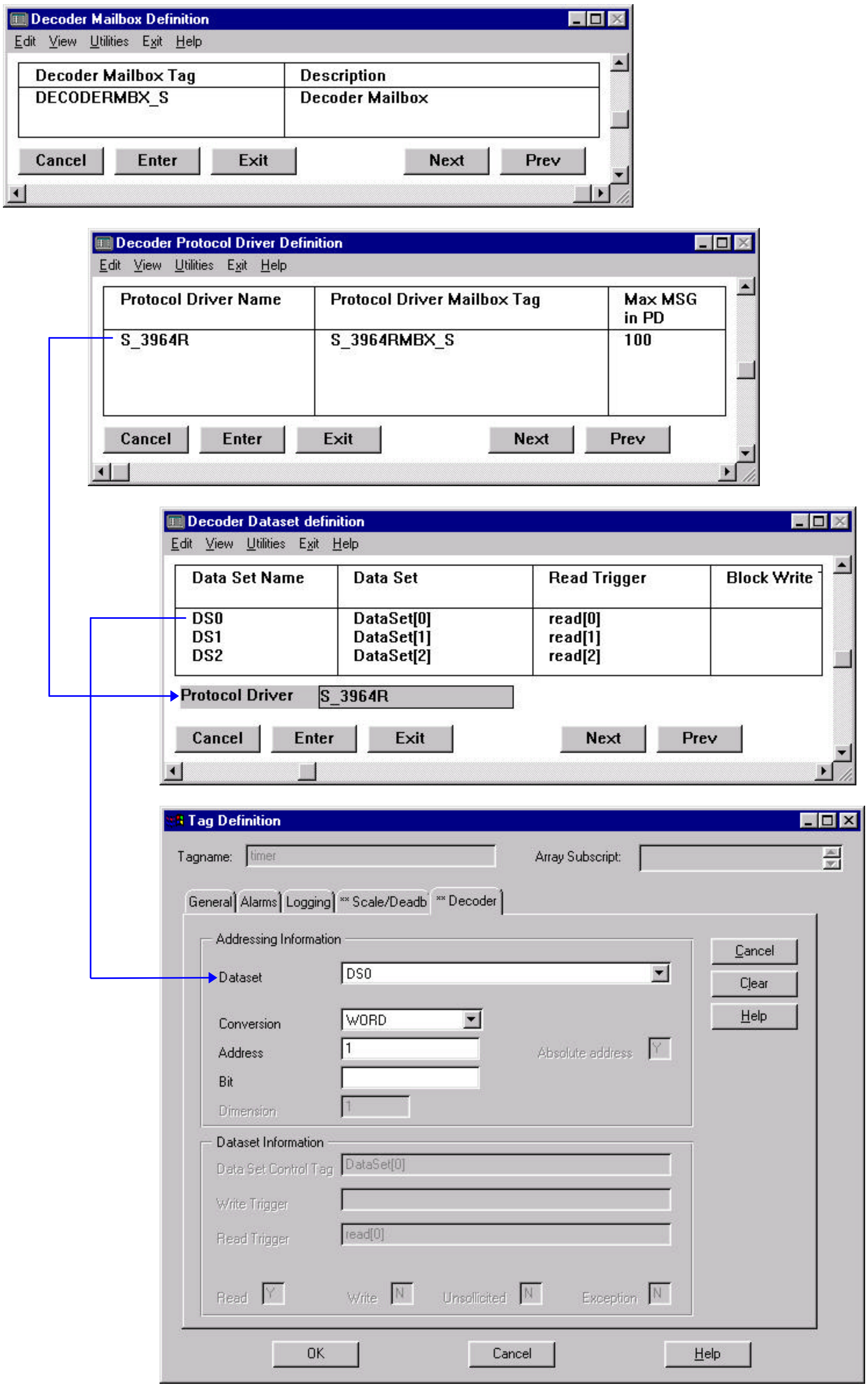
Read Trigger: read[0]

Read: Y
Write: N
Unsollicited: N
Exception: N

Figure 6.0.1 Decoder single point tag configuration.

Note: For general information about entering data in FactoryLink single point tag configuration table, refer to the FactoryLink Fundamentals Manual.

6.1. Single point tag definition for decoder task


Decoder Mailbox Definition
Edit View Utilities Exit Help
Cancel Enter Exit Next Prev
Decoder Protocol Driver Definition
Edit View Utilities Exit Help
Cancel Enter Exit Next Prev
Decoder Dataset definition
Edit View Utilities Exit Help
Cancel Enter Exit Next Prev
Tag Definition
General Alarms Logging ** Scale/Deadb ** Decoder
Cancel Clear Help
OK Cancel Help

Decoder Mailbox Tag	Description
DECODERMBX_S	Decoder Mailbox

Protocol Driver Name	Protocol Driver Mailbox Tag	Max MSG in PD
S_3964R	S_3964RMBX_S	100

Data Set Name	Data Set	Read Trigger	Block Write
DS0	DataSet[0]	read[0]	
DS1	DataSet[1]	read[1]	
DS2	DataSet[2]	read[2]	

Protocol Driver: S_3964R

Tagname: timer
Array Subscript:

Addressing Information
Dataset: DS0
Conversion: WORD
Address: 1
Absolute address: Y
Bit:
Dimension: 1

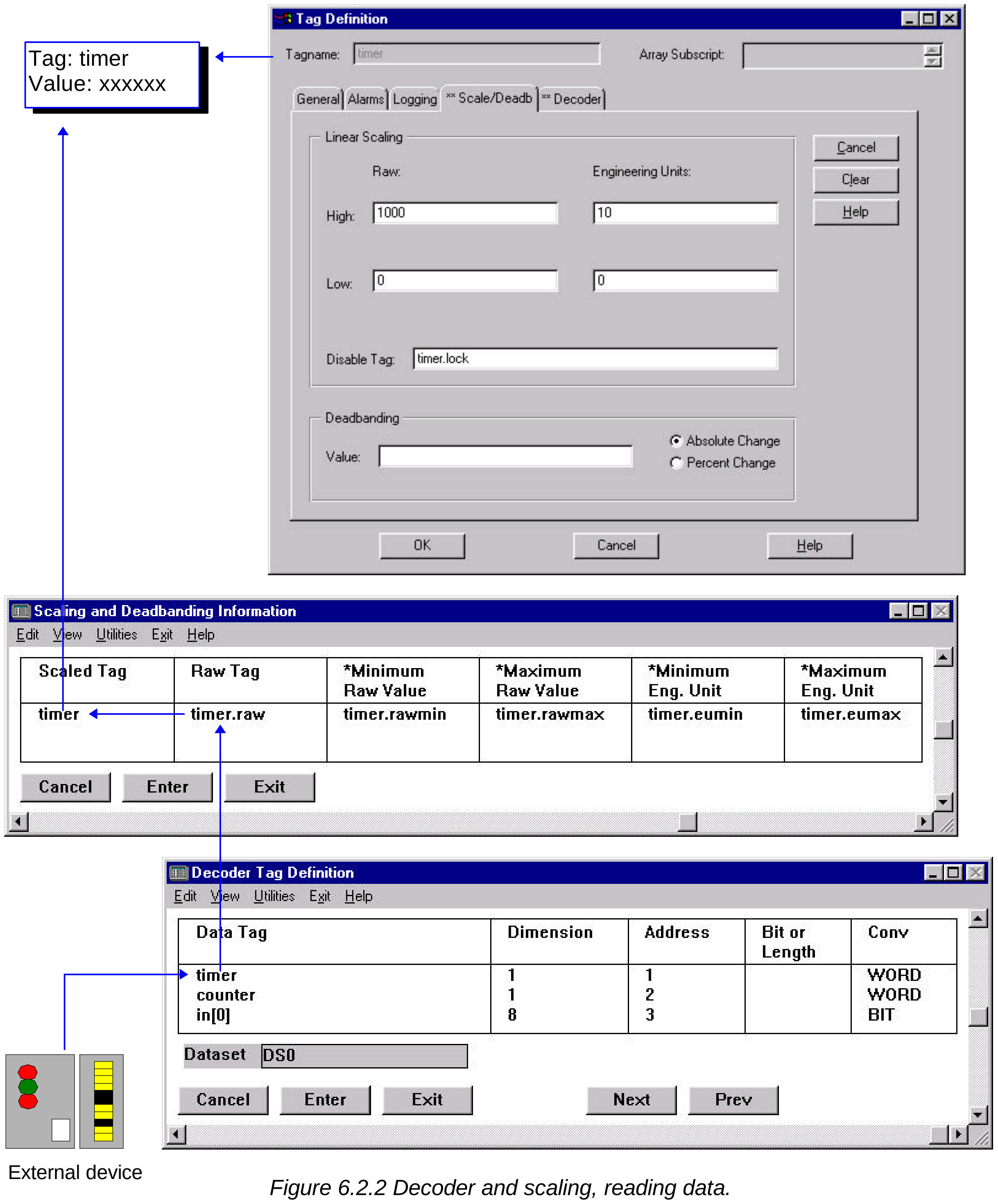
Dataset Information
Data Set Control Tag: DataSet[0]
Write Trigger:
Read Trigger: read[0]
Read: Y Write: N Unsolicited: N Exception: N

Figure 6.1.1 Single point tag configuration.

Before the single point tag definition for the decoder can be used in the application editor, datasets must be defined. The datasets are defined with the configuration manager by filling in the first three tables of the decoder, see the figure above.

6.2. Single point scaling in conjunction with decoder

In conjunction with the decoder, scaling of the data read/written from/to an external device is possible. To do so the scaling 'tab' has to be filled in, as is done in the figure below. When configuring the same is used for scaling and decoding out of a dataset. Internally there is additional tag created, which will be used by the scaling and decoder task. Normally this has the same name as the one for which you are configuring, but added to its name the extension '.raw'. This tag is called the raw value tag. For example when reading from an external device, the decoder updates the raw value with the value received from the protocol driver. As a result the scaling task scales the raw value (from the raw value tag) and places the result in the (normal) tag. However when you go to the configuration manager and access the *Decoder Tag Definition* panel of the decoder, you will find only the tag name without the extension '.raw'.



Tagname: timer
Array Subscript:

Linear Scaling

	Raw:	Engineering Units:
High:	1000	10
Low:	0	0

Disable Tag: timer.lock

Deadbanding

Value:

(● Absolute Change) ○ Percent Change

Scaled Tag	Raw Tag	*Minimum Raw Value	*Maximum Raw Value	*Minimum Eng. Unit	*Maximum Eng. Unit
timer	timer.raw	timer.rawmin	timer.rawmax	timer.eumin	timer.eumax

Data Tag	Dimension	Address	Bit or Length	Conv
timer	1	1		WORD
counter	1	2		WORD
in[0]	8	3		BIT

Dataset DS0

Figure 6.2.2 Decoder and scaling, reading data.

Appendix A. The delta.opt file

In order to run the task permanently with full functionality an authorization sequence for the task option is needed. This appendix describes the authorization sequence together with the option file.

The authorization sequence of every independent DeltaLink module must reside in the {FLOPT}/delta.opt file in order to take effect. In case the task has been ordered with an authorization sequence then this sequence resides on the install diskette (in the /opt/delta.opt file) and will be automatically added to the {FLOPT}/delta.opt file with the installation. In case the task has been previously installed from a demo diskette and the authorization sequence has been purchased later then the sequence must be added manually to the {FLOPT}/delta.opt file.

The format of the delta.opt file consists of two parts. The first part is the comment header. This part remains always at the beginning of the file. Every line of the comment part starts with an '*' character. The second part contains the authorization sequences of every independent DeltaLink module. Every line must hold one sequence code and must apply to a strict format.

An example of the option file looks like this:

Protection file: *delta.opt*

```
*
* Copyright 1994 DeltaLink bv. All Rights Reserved
*
* DeltaLink bv
* Saffierborch 6c
* NL-5241 LN Rosmalen
* The Netherlands
* Tel: (int) 31 4192 20025
* Fax: (int) 31 4192 15451
*
*
* FactoryLink Serial Number: 123450S2
*
* DeltaLink module option:
*
* MODULE = taskname of always 8 characters with trailing spaces
* .      = <space>
* X      = authorization code supplied by DeltaLink
*
* MODULE..XXXX.XXXX.XXXX.XXXX.XXXX.XXXX.XXXX.XXXX.XXXX
DECODER   2D6F E047 2534 036E 1215 EC80 92F8 1EFC C4C0
```

In case a full installation diskette with authorization diskette has been ordered then the header in the option file on the diskette contains the serial number of FactoryLink. The task will run only on the FactoryLink package with this serial number. If the serial number is not listed in the {FLOPT}/delta.opt file due to a previous demo installation then this number can be added in the header of this file.

The authorization code must exactly match the format as listed in the header. If this is not the case the module will not recognize the authorization sequence and start up in demo mode.

The format of an authorization sequence line is as follows:

MODULE<s><s>XXXX<s>XXXX<s>XXXX<s>XXXX<s>XXXX<s>XXXX<s>XXXX<s>XXXX<s>XXXX<CR><LF>

The **MODULE** field contains the DeltaLink module name in this case DECODER. This field must always be 8 characters long. If the module name is shorter then 8 characters then the name must be filled out with spaces to 8 characters.
After the MODULE field one space must be entered.

After the space field 9 records must be specified with the authorization code. One record is build up of one leading space (ASCII 0x20) and four sequence codes. The sequence codes must be entered exactly as specified by DeltaLink.
After the authorization code records a carriage return (ASCII 0x13) and linefeed (ASCII 0x10) must follow.

There may be no empty lines between the specification of more than one module. To add an authorization sequence a normal editor can be used. If all modules with the right authorization codes are specified according to the format described above then the modules will start with full functionality.

Appendix B. Command line parameters

The protocol driver accepts several command line parameters, these can be configured with the configuration manager in the ‘System Configuration’ table, column ‘Program Arguments’. An argument consists out of first a minus sign (‘-‘), followed by the a letter specifying the option. After the letter an optional number can be present, if this is supported by the option.

Option	Description
-d*n*	Debug option, the level of debug information is set with the number *n*. The range of this number is from 1 until 4. If no number is specified the default level will be 1. The debug output will be visible in the ‘window’ of the protocol driver, that of the run time manager.
-l*n*	Same as the previous option, difference is the output device. For this option an ASCII log file is generated, being the file: {FLAPP}/{FLNAME}/{FLDOMAIN}/log/s_3964r.log
-LS*n*	Local Station id with number *n*, needed if the decoder and protocol driver reside on different nodes (or FactoryLink workstations). This must be an unique number for the network.

This page is left blank intentionally.

Appendix C. Error codes

The error code is returned to the user in a user-defined status tag. These error codes will also be printed with the message of the DeltaLink DECODER in the run-time manager. The errors can be generated from different parts of the task which will be listed here:

Error #	FactoryLink errors
100	Bad message type
101	message with dataset control tag not found in queue
102	no messages available to query
103	bad receive mailbox tag
104	bad mailbox send tag
105	bad dataset control tag

Error #	FactoryLink errors
401	Internal error
402	Out of memory
403	Operating system error
404	Initialization not successful
405	Initialization not successful
406	Incorrect function
407	Incorrect argument
408	Incorrect data
409	Bad tag
410	Null pointer assignment
411	Change flag not set
412	Procedure table full
413	Bad procedure name
414	Bad user name
415	Bad option
416	Incorrect checksum
417	No options
418	No key
419	Bad key
420	No port available
421	Port busy
422	FL already active
423	No lock
424	Lock failed
425	Lock expired
426	Wait failed
427	Termination flag set
428	Q-size to big
429	Q-size changed
430	No tag list
431	Tag list changed
432	Wake up failed
433	No signals
434	Signaled
435	Not a mailbox
436	No messages
437	Access denied
438	Attribute failure
439	Invalid attribute
440	Attribute not defined
441	Application exists
442	RTDB does not exist
443	No task bit
444	Not a lite task

Error #	IMX errors
450	Bad message type
451	Message with dataset control tag not found in queue
452	No messages available to query
453	Bad receive mailbox tag
454	Bad mailbox send tag
455	Bad dataset control tag
456	Message cannot be adjusted
457	Operation to big for variable
458	Unknown boundary
459	Function not supported
460	No message for this index present
461	The remote dataset Is not defined on this system
462	The received dataset was not registered
463	The message is not queued
464	Message is rejected due error in the remote IMX
465	Illegal method of addressing bits on bit boundary
466	Element cannot be written
467	Invalid buffer specified
468	Block write function impossible
469	Maximum number of messages in MBX reached
470	No memory left
471	Error registering standard dataset
472	Error writing message in pipe
473	Not supported IMX message
474	Error creating pipe
475	Error starting threat
476	Error server connecting to pipe
477	Error child connecting to pipe
478	Error reading the pipe
479	Error number of bytes read from pipe
480	Error writing into the pipe
481	Error number of bytes written into the pipe
482	Error reading dataset from the mailbox
483	No communication buffers assigned
484	Error decrementing semaphore
485	Error writing to pipe, no space left
486	Error querying no. of messages for decoder
487	Max. No. of messages in decoder MBX reached

Appendix D. Messages

If an error condition occurs in the decoder task during run-time mode, a message to that effect will appear on the runtime manager graphics screen to the right of "DECODER". The error messages that may be displayed are as follows:

DeltaLink protection key missing
The DeltaLink protection key is not connected to the parallel port or there is no decoder task enabled in the key.

Can't open CT file
The decoder task was unable to open the configuration table file, generally because it does not exist. This is a fatal error.

No triggers defined
The decoder configuration table file does not contain an expected trigger. This is a fatal error.

Error reading CT index
An error occurred during reading a Configuration Table index, normally this means the CT-file is corrupted. This is a fatal error.

Error reading CT header
An error occurred during reading a Configuration Table header, normally this means the CT-file is corrupted. This is a fatal error.

Error reading CT record
An error occurred during reading a Configuration Table record, normally this means the CT-file is corrupted. This is a fatal error.

Out of RAM
There is not enough RAM memory to load the complete configuration and/or task.

Invalid tag number
The decoder task encountered an invalid TAG number. This is a fatal error.

CT read trigger Protocol Driver %d, ds %d, %d
The decoder task encountered an invalid TAG, for triggering the block read. Specified are the record number of the protocol driver and the TAG. This is a fatal error.

CT Block Write trigger Protocol Driver %d, ds %d, %d
The decoder task encountered an invalid TAG, for triggering the block write. Specified are the record number of the protocol driver and the TAG. This is a fatal error.

CT Protocol Driver not found
The decoder task could not find the configuration table file for the protocol drivers. This is a fatal error.

CT ds not found
The decoder task could not find the configuration table file for the datasets. This is a fatal error.

CT tag validity pd %d, ds %d, tag %d
The protocol driver did not validate the specified TAG.

CT unknown conversion ds %d element %d
The decoder task encountered an invalid conversion for the specified dataset and element of that dataset, both numbers are record numbers.

IMX error response %s error %d

Communication error, the decoder received an error response for a read/write command. Specified are the functionality and the error number. The error number is received from a protocol driver, the meaning of the error depends on the driver (see the appropriate manual).

IMX init %d

IMX initialization error, internal error of the decoder. Specified is the error number.

IMX maximum msg in PD MBX %d reached

IMX error, the maximum number of allowed messages in a protocol driver mailbox tag is reached. Specified is the record number of the protocol driver mailbox.

IMX unknown message received %d

IMX error, internal error of the decoder task. Specified is the number of an unrecognized command.

IMX check number message %s %d

IMX error, internal error of the decoder task. Specified are the protocol driver and the error number.

IMX send pd %d error %d

IMX error, internal error of the decoder. Specified are the record number of the protocol driver and the error number.

IMX index identification not supported %d

IMX error, protocol driver tried to identify a dataset by a number. This is not supported in this version of the decoder. Specified is the dataset number.

IMX unknown boundary %d

IMX error, internal error of the decoder, the protocol driver specified an unknown boundary. Specified is the boundary.

IMX error packet %s: nr %d

IMX error, internal error of the decoder. Specified are the functionality and the error number.

EW wrong conversion %d

Conversion error, conversion for exception write failed. Specified is the conversion type.

EW cannot convert

Conversion error, internal error of the decoder. An exception write can not be converted to the correct value/boundary.

FL reading data pd %d error %d

FactoryLink error reading the real-time database. Specified are the record number of the protocol driver and the error number.

FL writing data pd %d ds %d error %d

FactoryLink error writing the real-time database. Specified are the record number of the protocol driver, record number of the dataset and the error number.

FL forced write pd %d error %d

FactoryLink error forced writing the real-time database. Specified are the record number of the protocol driver and the error number.

FL change read pd %d error %d

FactoryLink error reading the real-time database on change. Specified are the record number of the protocol driver and the error number.

FL set change bit pd %d error %d

FactoryLink error setting change bits. Specified are the record number of the protocol driver and the error number.

FL reset change bit pd %d error %d

FactoryLink error resetting change bits. Specified are the record number of the protocol driver and the error number.

FL clear change flags %s error %d

FactoryLink error clearing change bits. Specified are the record number of the protocol driver and the error number.

This page is left blank intentionally.

Appendix E. Conversions

This section contains a full list of possible conversions with description.

General conversions

Conversion	Description
BIT	BIT conversion.
BYTE	Byte conversion.
BR	RIGHT byte conversion.
BL	LEFT byte conversion.
BCD2	Binary Coded Decimal byte conversion.
BCD	Binary Coded Decimal word conversion.
BCD4	Binary Coded Decimal word conversion.
BCD8	Binary Coded Decimal long conversion.
WORD	WORD conversion.
LONG	LONG conversion.
DBL	DOUBLE precision float conversion.
IEEE	IEEE single precision float conversion.
MSG	message conversion.

Time conversions

Conversion	Description
TIME	Time conversion, based on long conversion. External device time is long analog, starting on January 1, 1970, FactoryLink time starts on January 1, 1980.
TIM0	User defined time conversion: byte stream in external domain, FactoryLink time.
TIM1	User defined time conversion: byte stream in external domain, FactoryLink time.
TIM2	User defined time conversion: byte stream in external domain, FactoryLink time.
TIM3	User defined time conversion: byte stream in external domain, FactoryLink time.
TIM4	User defined time conversion: byte stream in external domain, FactoryLink time.
TIM5	User defined time conversion: byte stream in external domain, FactoryLink time.
TIM6	User defined time conversion: byte stream in external domain, FactoryLink time.
TIM7	User defined time conversion: byte stream in external domain, FactoryLink time.
TIM8	User defined time conversion: byte stream in external domain, FactoryLink time.
TIM9	User defined time conversion: byte stream in external domain, FactoryLink time.

Reversed conversions:

Conversion	Description
RLNG	Reversed LONG conversion
RFLT	Reversed IEEE single precision float conversion

Special conversion for Siemens

Conversion	Description

SIE	SIEMENS S5 floating point conversion.

Special conversion for the Interbus-S:

Conversion	Description
IBG	Interbus-S Gain for analog in/out.
IBAU	Interbus-S Analog Input/Output 12 bits unsigned.
IBAS	Interbus-S Analog Input/Output 12 bits two's complement.

Special conversion for the Valmet - Damatic:

Conversion	Description
FDAM	Valmet Damatic floating point conversion

Appendix F. Time Conversions

The time conversions: TIM0, TIM1, TIM2, TIM3, TIM4, TIM5, TIM6, TIM7, TIM8 and TIM9 convert date/time
from an external device, being a byte stream to a long integer in FactoryLink style. This means the value zero represent the date January 1, 1980.

The general settings for the decoder are located in the format file *decoder.txt*, this file is default located in the {FLINK}/msg directory. This file will always be loaded by the task. On top of this configuration an application dependent configuration file may be loaded by copying the format file or parts of the format file to a configuration file in the {FLAPP}/msg directory. The file name in this directory should also be *decoder.txt*. If a configuration item is present in both files, the last loaded one will be the active one, so this is the item in the file: {FLAPP}/msg/decoder.txt. Note that except the time conversions, there are also general translation items located in the configuration file.

Protocol settings file: *decoder.txt*

```
*-----------------------------------------------------------------------
*
* Format translation for user defined time conversions
*
*-----------------------------------------------------------------------
*
* Date-Time Translation can be specified by the user. The user has to define
* the date-time configuration from the external device. This definition is
* mainly specified as the byte order in which the date-time structure is
* received from (or send to) the external device.
* First specified in the conversion is the coding of the data, Binary Coded
* Decimal (BCD) or Binary (BIN). Second is the order inside the date-time
* structure. The next elements can be specified:
*
* Element     Size    Description
*
* X           byte    Byte to ignore.
*
* HR          byte    Hours (0 -23)
* MI          byte    Minutes (0 -59)
* SC          byte    Seconds (0 -59)
* MS          word    Milliseconds
*
* DOW         byte    Day of week (1 - 7, 1 = Sunday, 2 = Monday, etc.)
* DOW0        byte    Day of week (0 - 6, 0 = Sunday, 1 = Monday, etc.)
* DOW1        byte    Day of week (1 - 7, 1 = Sunday, 2 = Monday, etc.)
* DY          byte    Day of the month (1 - 31)
* MO          byte    Month (1 - 12, 1 = January, 2 = February, etc.)
* MO0         byte    Month (0 - 11, 0 = January, 1 = February, etc.)
* MO1         byte    Month (1 - 12, 1 = January, 2 = February, etc.)
* YP          byte    Hundreds of year, two digits
* YR          byte    Year, two digits
*
*-----------------------------------------------------------------------
*-----------------------------------------------------------------------
* Time        Conversion     Byte order
*-----------------------------------------------------------------------
TIM0          BCD            SC MI HR DY MO YR
TIM1          BIN            X DOW X MO X DY X YR X HR X MI X SC
TIM2          BIN            SC MI HR DY MO YR
TIM3          BIN            SC MI HR DY MO YR
TIM4          BIN            SC MI HR DY MO YR
TIM5          BIN            SC MI HR DY MO YR
TIM6          BIN            SC MI HR DY MO YR
TIM7          BIN            SC MI HR DY MO YR
TIM8          BIN            SC MI HR DY MO YR
TIM9          BIN            SC MI HR DY MO YR
*-----------------------------------------------------------------------
```

```
*-----------------------------------------------------------------------------
* Months of the Year
*
*-----------------------------------------------------------------------------
MO0             Jan
MO1             Feb
MO2             Mar
MO3             Apr
MO4             May
MO5             Jun
MO6             Jul
MO7             Aug
MO8             Sep
MO9             Oct
M10             Nov
M11             Dec

*-----------------------------------------------------------------------------
* Days of the Week
*
*-----------------------------------------------------------------------------
DOW0            Sun
DOW1            Mon
DOW2            Tue
DOW3            Wed
DOW4            Thu
DOW5            Fri
DOW6            Sat

*-----------------------------------------------------------------------------
* Hours of the Day
*
*-----------------------------------------------------------------------------
DAM             am
DPM             pm

*-----------------------------------------------------------------------------
* Date/Time Format (Translation to message tags)
* You can change the sequence of the following tokens
*
* yr : two digit year
* mo : two digit month (jan = 01)
* mon: three digit month (jan, feb etc)
* dy : two digit day   (01.. 31)
* dow: three digit day of the week (sun, mon, etc)
* hr : two digit military hour (00..23)
* ah : two digit hour (00..12)
* mi : two digit minutes (00..59)
* sc : two digit seconds (00..59)
* ap : two digits am/pm
*
*-----------------------------------------------------------------------------
* Default Format (Translation to message tags)
*
* DATETIME        mo/dy/year hr:mi:sc
*-----------------------------------------------------------------------------
DATETIME        mo-dy-year hr:mi:sc
*-----------------------------------------------------------------------------
```