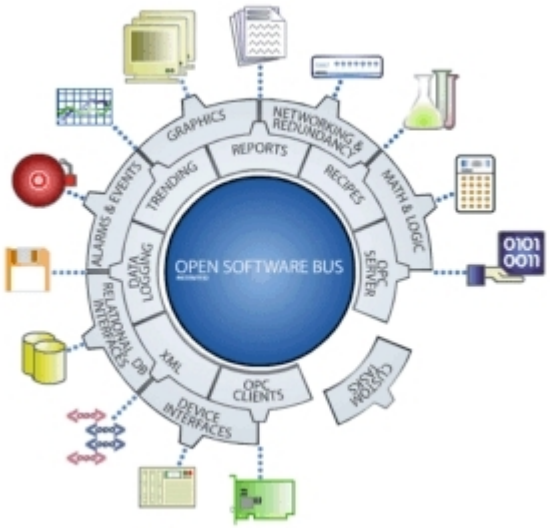
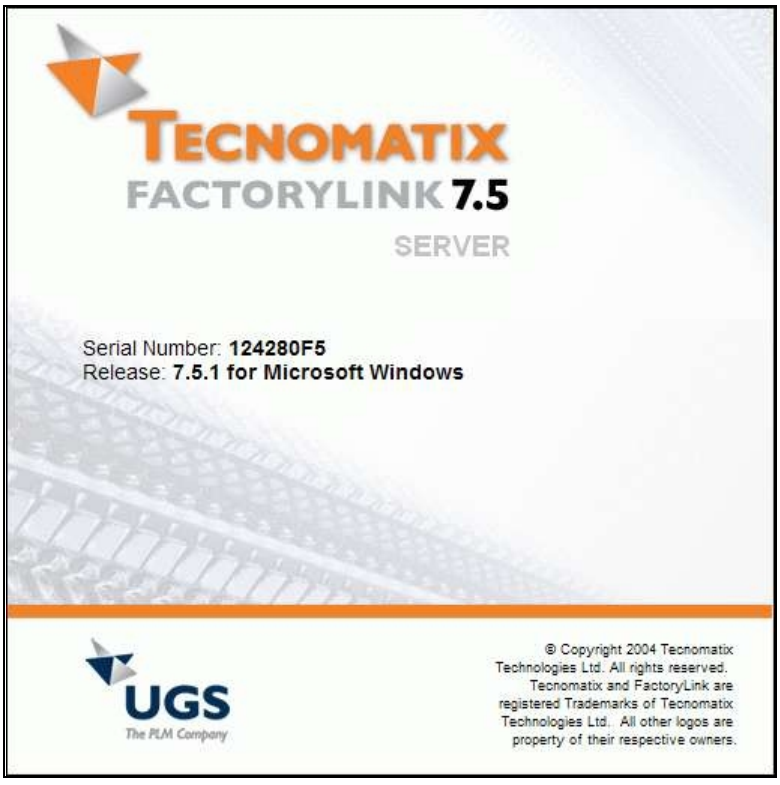
Tecnomatix®



FactoryLink SAPI Protocol Driver

for

FactoryLink 7.x.x



SAPI_S7

Version 1.0
Printed: donderdag, augustus 02, yyyy

Table of Contents

SAPI Protocol Driver

Provides the following features:

- ☑ Supports the S7-300/400 PLC range.
- ☑ Runs on Siemens Softnet, which supplies range of different network types and protocols.
- ☑ Implements the RAPD driver principle of Usdata, which means that this driver communicates with the IOXlator.
- ☑ Full asynchronic communication on Read, Write and Receive.
- ☑ Implements the cyclic read functionality for very fast communication.
- ☑ Selectable boards. Makes it possible to have more then one board in the same PC, and even switch communication from one board to another.
- ☑ Full control over communication.
- ☑ Enable tags on all functions per PLC.
- ☑ Ease of use. Just specify which areas to Read, Write or Receive.

Third Party Software / Hardware Dependencies:

- ☑ Siemens communication board, choice depends on network type: CP1413, CP1416 for Ethernet or TCP/IP, CP5412, CP5611, CP5613 for Profibus.
  For TCP/IP you can also use standard ethernetcards supported by the platform.
- ☑ Programmable Logic controller of type Siemens S7 300/400 series. PLC's must be equipped with a communication card, e.g. CP143 (Ethernet).

Supported Protocols:

- ☑ Supports the protocols supplied by Siemens Softnet which can be of the following: Profibus, MPI, Industrial Ethernet (TP4 or TCP/IP).

1 Introduction

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

2 Scope of this document

This manual is written for a technician who is familiar with both the FactoryLink® software and the Siemens S7 or Siemens Programmable Logic Controllers (PLC's). This document can be used both as a training manual as well as a reference manual.

The first section of this manual deals with the installation of Siemens S7 driver. Note that for installation of a complete working system also the Softnet product of Siemens has to be installed. Make sure to install this first according to the documentation of Siemens Softnet.

The second part explains the operation principles of the communication with the Siemens PLC and the translator. Here all terms and definitions are explained to the reader. It explains terms like "Dataset" and "Virtual Connections'. 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 this driver. 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 this driver with a translator like the ioxlator. It shows the entries made for the pre-configured demo, which comes with this package. The demo program can be used to check if the communication is working without making a complete application.

The last part is the appendices, which contain summarised data.

DeltaLink bv, Pioenroosstraat 26, 5241AB Rosmalen, The Netherlands,

tel + 31 73 523 1234
fax +31 73 523 1222

This page is left blank intentionally.

3 Installation

4 Installation of the FactoryLink software

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

Before installing

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

First

Copy the files from the "DeltaLink S7 driver" media to the appropriate directories. This will be done automatically by running the install utility placed on the installation media.

For the Windows platform go to the directory with the install build and execute the following command:

```
install↵
```

Second

After you installed the software you need to activate the tables in the FactoryLink Configuration Manager (FLCM). The installation utility automatically appends the *sapi_s7.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 sapi_s7 entry in the following table

```
file {FLINK}/ac/titles

...
windhdr.ac
loxlator.ac
sapi_s7.ac
spool.ac
...
```

Third

To make sure all the Configuration Tables (CT's) are generated after a change, the install utility automatically adds the *sapi_s7* 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

...
recipe rcphdr rcpovr
loxlator loxlatorm loxlatorp loxlatord loxlatort
sapi_s7 sapis7d sapis7p sapis7x sapis7m
iml imltags imltrig
...
```

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

Fourth

To enable the help functionality for the Siemens S7 driver tables in the Configuration Manager, the install 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 manager utility is present to update this conversion file, and can be started from the command line. This utility will also be started automatically from the install utility.

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

Sixth

The protocol driver 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*
Sapi_s7	***Siemens Sapi S7 Protocol Driver***	***bin/sapi_s7***

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.

5 Installation of the protection

The S7 driver is protected via the DeltaLink option file. This file contains authorization sequence codes for DeltaLink modules. The protection is linked with the serial number of the FactoryLink package.

6 The DeltaLink option file

The installation build of the S7 driver contains an option file, named *delta.opt*, in the *opt* directory. This file contains the unique authorization sequence which enables the Sapi S7 driver to run unlimited. 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 B*.

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.

7 Demo installation

It is possible to install the S7 driver 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 driver 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 B*.

The limitations of a demo version of the Siemens Sapi S7 driver are:

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

Note that when authorizing the S7 driver only the authorization code has to be entered, the driver need not to be installed again.

8 Principle

9 The Translator RAPD driver 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. Ioxlator, high speed logger). The RAPD system consists of a protocol driver which communicates with external devices (RTUs, PLCs, 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 translator communicate with each 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 one translator and several protocol drivers. For example the translator together with the S7 protocol driver and 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.

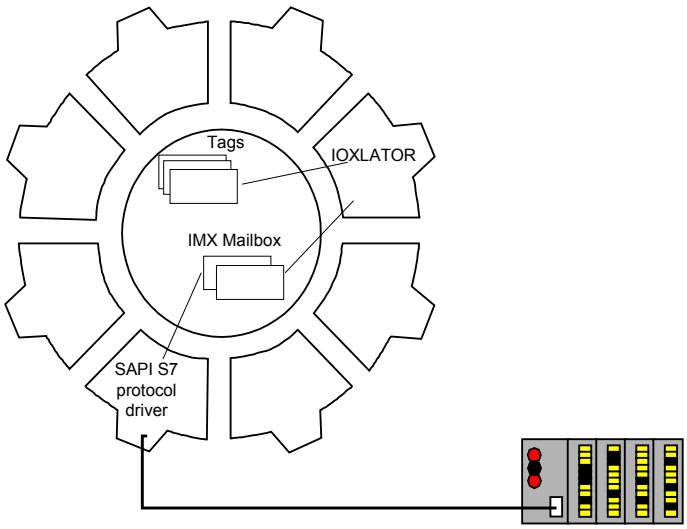


Figure 4.1.1 The RAPD principle.

10 S7 Communication Principle

11 Siemens Softnet

The DeltaLink S7 driver has been developed on top of Softnet from Siemens, using the SAPI interface. The abreviation SAPI stands for Simple Aplication Programmers Interface and is an generic interface for use with different protocols. This means that the S7 driver is capable of driving different protocols just by installing the correct version of Softnet.

At the moment it is possible to use the S7 driver with industrial ethernet (TP4 or TCP/IP), profibus or MPI. Just by selecting the right communication protocol within Softnet these options become possible. The S7 driver will be configured independently from the type of communication protocol. Only references will be made to addressing information configured in Softnet. This means that the S7 driver is completely unaware of which communication protocol is being used. It is possible to change from one communication stack to another without changing the configuration of the S7 driver. For more information on Softnet refer to the Siemens documentation on this subject.

12 Datasets

A dataset is a contiguous area of data in the PLC, which has certain characteristics such as a boundary, start and length. The S7 driver can operate on this dataset by performing IMX commands, which can be reading, writing or unsolicited receiving. The dataset will be uniquely identified by the dataset control tag, which will be defined in the S7 driver. Only one unique dataset control tag per dataset must be defined because it will refer to the data area in the PLC.

Operations on data within a dataset will be done outside the S7 driver by a translator module. This module will extract single elements from the dataset and convert them to the correct type. Datasets of the S7 driver can be of the following type:

Siemens S7data type	
Data type entry	Description
DB	Data Block
FB	Flag byte
IB	Input byte
OB	Output byte
PB	Peripheral byte
CW	Counter word
TW	Timer word
SW	System Data word
DX	Expanded Data Block
DE	Datablock in extended memory
EB	Expanded Peripheral byte

Table 3.2.2.1 Dataset types

13 Virtual connection

The S7 module uses virtual connections to communicate with the PLC. Per PLC one connection has to be established over which all IMX services will execute. The following services are supported:

Read/Write Service	Supported
Block read	Yes
Block write	Yes
Exception write	Yes
Encoded write	Yes
Unsolicited Receive	Yes
Cyclic Read	Yes

Table 3.2.3.1 Services supported

Before communication can take place, a connection has to be established with the PLC. To establish this connection addressing information on how to reach the PLC has to be configured. This addressing information will not be configured in the S7 driver but in the Softnet product of Siemens. This addressing information differs according to the protocol stack chosen within Softnet. Within Softnet two utilities must to be used for configuration: *Setting the PG/PC Interface* and the *COML S7* utility. The first utility is to choose the communication stack and card being used, the second to enter addressing information for connections. These connections will be given a logical name. For more information on how to configure Softnet refer to the Siemens documentation on this subject.

In the configuration panels of the S7 driver, connections will be addressed by referring to the logical connection name in Softnet. Each PLC must be related with a unique logical connection name. This name must be entered in the *Connection* field of the *Siemens SAPI device definition.* This means that the S7 driver is unaware of which protocol stack is being used. This way it is very easy to exchange protocol stacks.

14 Block read and block write

The block read and block write commands are performed on a contiguous block of data in the PLC: the dataset. The FactoryLink station initiates both commands. The translator module sends requests to the S7 driver for reading or writing an entire dataset. For configuring the translator refer to the manual of the translator. The S7 driver will perform the requested command and send a reply of the command back to the translator.

15 Exception write

An exception write is initiated by the FactoryLink workstation, and writes data to the PLC. Not a complete dataset but one element from the dataset is written to the PLC. The protocol driver will receive a request for an exception write from the translator. The first action the driver will take is to check if the write can be accomplished with one write or must be read first.

If the exception data element size is smaller than the size of the PLC data type element (boundary) the data must be read first, patched with exception data and written back. If this is not done, data will be unintentionally overwritten in the PLC. Whether this is the case depends on the data type of the dataset.

For example a S7 plc with a bit in a Data Block (DB) has to be set then the specific data byte has to be read first because the smallest element in a Data Block is a byte. The bit has to be patched into the byte and written back.

In the example mentioned above it is possible, that the device changes the contents of the byte after the driver has read the byte but before the new value is written. This means that after the write of the protocol driver the content change of the PLC is lost. For applications which require that single elements can be accessed both by the PLC and the S7 driver, an encoded write can be used. The encoded write will be discussed in the next section.

16 Encoded write

The encoded write function is almost similar to the exception write in the fact that both functions write single data elements on exception. The difference is that the exception write accesses directly the destination address in the PLC whereas the encoded write composes an encoded write command and writes this command to a location in a certain data block. The PLC program reads and decodes this encoded write command and accesses the actual destination location.

The advantage of this method is that for every element to write only one write command is needed because the PLC program does the actual operation on the data element (no reading before writing). Another advantage is that the PLC program can control all encoded writes because they all are written to one location in the PLC.

The location where the encoded write command will be placed is part of the definition of a logical device; a detailed description is in the next chapter.

17 Unsolicited receive

In case of an unsolicited receive the PLC sends a dataset to the FactoryLink workstation, without a specific request from that workstation. This means that the action is initiated by the PLC. The protocol driver will check if the received data is configured in one of the datasets. If this is the case then the data will be sent with the dataset information to the translator which converts and updates the data. If no dataset is specified regarding to the received data, the data will be discarded.

The unsolicited receive function is a very powerful functionality for fast communication because no request has to be placed by the protocol driver. This way less overhead is involved in communication. Data which alters unpredictably such as alarms are very suitable for the unsolicited receive function. Some extra programming effort has to be done in the PLC because the PLC is in charge of sending the data.

18 Cyclic Read

The cyclic read functionality is a very powerful type of communication implemented by the S7 protocol. The principle of this communication method is as follows: With cyclic read the CP in the PLC is instructed to send data with a fixed interval to the FactoryLink station. This sending must be started and will continue indefinite until a command to stop will be send. This service maps to the IMX unsolicited receive functionality. This means that data received from the cyclic read functionailty will be treated as if it were unsolicited data and forwarded to the translator.

This type of communication has some very big advantages. First of all the CPU load on both the FatoryLink system as the PLC will be reduced. Because the FactoryLink system doesn't have to initiate IMX requests but only receive unsolicited data, the CPU load will be drastically reduced. The CP on the PLC side will handle complete communication, no additional programming has to be done which also reduces CPU load on the PLC drastically.

Another advantage is that the performance of communication is very high because the unsolicited receive functionality is being used which means that no triggering has to be done on both sides. Together with the ease of configuration, the cyclic read is a very suitable type of communication.

The starting and stopping together with the update interval has to be configured in the S7 driver. This has to be done in the panel *Siemens S7 Dataset Definition* at the *Cyclic Read* field. In case the change flag of the tag is set the the S7 driver will read the value of the tag and start the cyclic read with the specified update rate. An update rate of zero means in this case to stop the cyclic read of the dataset.

19 FactoryLink domain selection

The standard domain for the S7 driver is the SHARED domain. The protocol driver communicates with a dedicated piece of hardware, therefore only one task should be able to access the hardware. If only one program accesses the hardware, the task should be located in the shared domain and therefore started by the shared runtime-manager.

Important The protocol driver and the translator must be in the same domain (either SHARED or USER).

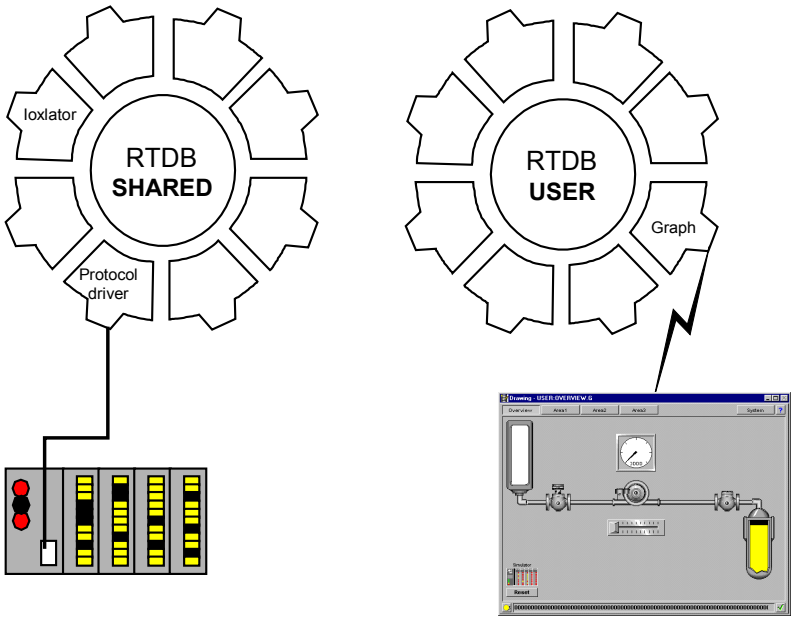


Figure 3.3 Standard domain selection.

20 Configuration tables

In the Configuration Manager Main Menu, select **Siemens SAPI S7 Protocol Driver**. 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.

21 Mailbox definition

From the display of all the panels, select the *Siemens SAPI Mailbox Definition* panel.

The Siemens SAPI mailbox definition panel allows the user to configure one mailbox tag for the SAPI S7 protocol driver. This mailbox will be used by the driver to receive IMX requests from other modules within FactoryLink. Specify the following information.

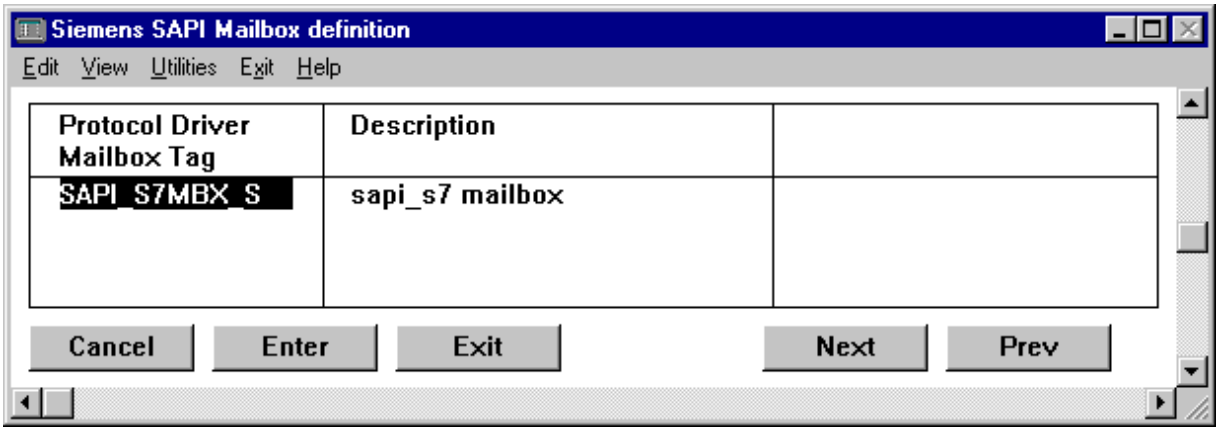


Figure 4.1.1 DeltaLink SAPI Mailbox Definition.

- **Protocol Driver Mailbox Tag**

  Tag name of the SAPI S7 protocol driver mailbox element to be referenced by a translator task i.e. the ioxlator module. The ioxlator module uses this mailbox to send requests to the SAPI S7 protocol driver.

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

- **Description**

  Description of the SAPI S7 protocol driver mailbox element defined at creation of this tag.

  valid entry Output only.

22 Ioxlator definition

From the display of all the panels, select the *Siemens SAPI Translator definition* panel.

The SAPI S7 Translator Definition panel allows the user to specify one or more translators i.e. the ioxlator. This mailbox will be used by the driver to reply the IMX request to the right translator which will then decode the IMX mailbox message. The panel shows an example of a configuration.

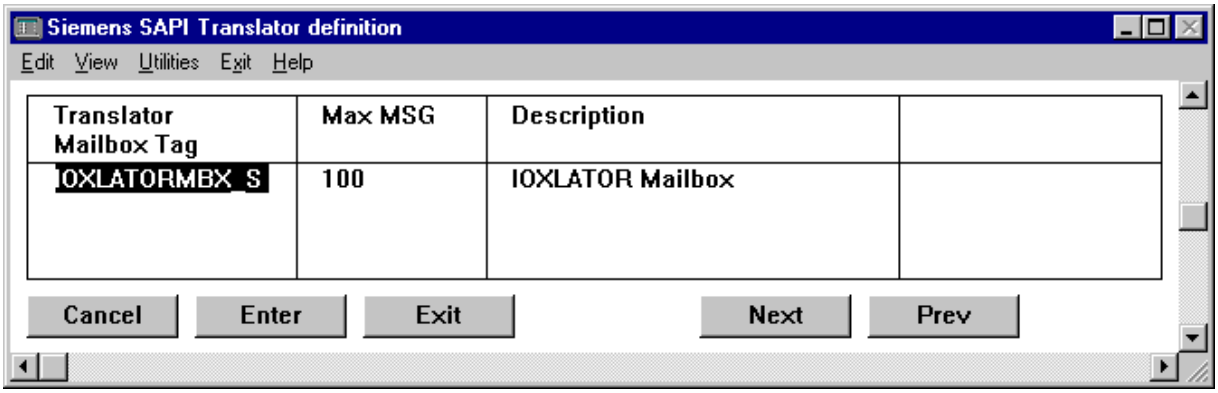


Figure 4.2.1 Siemens SAPI Translator Definition.

- **Translator Mailbox Tag**

  Tag name of a translator mailbox tag, i.e. the ioxlator module, to be referenced by the S7 module. The S7 module will write IMX responses, which are meant for a specific translator into this mailbox. The translator module uses this mailbox to receive data from the S7 driver and decode this data.

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

- **Max MSG**

  The maximum number of messages defines a limit of the number of IMX responses that the translator mailbox can contain. This value has been implemented as a security option to prevent the translator mailbox from filling up. This can happen for some reason in case the translator can not handle all IMX responses fast enough. The number of IMX responses should be monitored at run-time and should be normally zero with occasionally peaks. This value can be tuned depending on these peaks.

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

- **Description**

  Description of the translator mailbox element defined at tag creation.

  valid entry Output only.

23 Device definition

From the display of all the panels, select the *Siemens SAPI S7 Device Definition* panel.

The Siemens SAPI S7 Device definition panel must be used to specify logical devices to communicate with. The logical device will be related to the Softnet product from Siemens through the *Board* and *Connection* field. Within Softnet the actual addressing must be configured according to the type of communication protocol selected, for example profibus or industrial Ethernet.

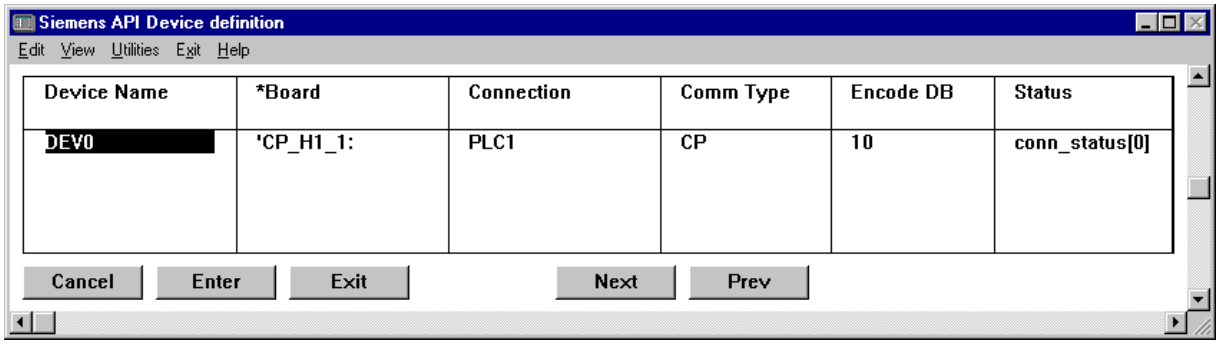


Figure 4.3.1 DeltaLink SAPI_S7 Device Definition.

- **Device Name**

  Logical name to define a communication device. This name will only be used internally in the S7 module and will also appear as the link in the next panel *DeltaLink S7 Dataset Definition*

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

- ***Board**

  Logical name to select the communication board for use. This logical name must relate to a definition of a board in the Siemens Softnet configuration. This configuration must be done in the utility *Setting the PG/PC Interface* at the *Access Point of Application* selection. On how to configure a specific type of communication within the Softnet product of Siemens refer to the documentation of Softnet. This field can be configured either a constant or as a tag. In case a tag has been configured then it is possible to change the communication board at run-time.

  entry Required.
  entry type Alphanumeric string or tag name
  valid type String of up to 16 characters.

♦ Connection

The connection field refers to a connection definition within Softnet. This referral determines to which PLC will be connected and thus communicated with. The entry of this field is a logical name, which must be defined in Softnet. The connection must have been defined in Softnet using the *COML S7* utility (SIMATIC NET Configuration Management Local for COML S7). With this utility a connection can be defined or every type of communication. On how to define these connections refer to the documentation of the *COML S7* product.

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

♦ Comm Type

The S7 module supports two types of communication: CP or CPU. The major difference between the two types is that CP communication handles all the request in the CP module of the PLC i.e. CP434 and the CPU type handles all request in the CPU module of the PLC. Recommended is to use the CP type because almost no configuration in the PLC has to be done and the performance at run-time is higher because there is almost no load on the CPU. In combination with the cyclic read functionality this will give the best performance and CPU load results (on both sides, PLC and PC). The only reason to use the CPU type is because it can handle larger blocks of data.

entry	Required.
entry type	Selection key field.
valid entry	CP or CPU.

♦ Encode DB

This field defines a datablock in the PLC where the encoded write commands will be written to. The encode FB in the PLC will check this datablock for incoming requests and perform the actual writing.

entry	Optional.
entry type	Decimal number.
valid entry	Positive integer 1..255.

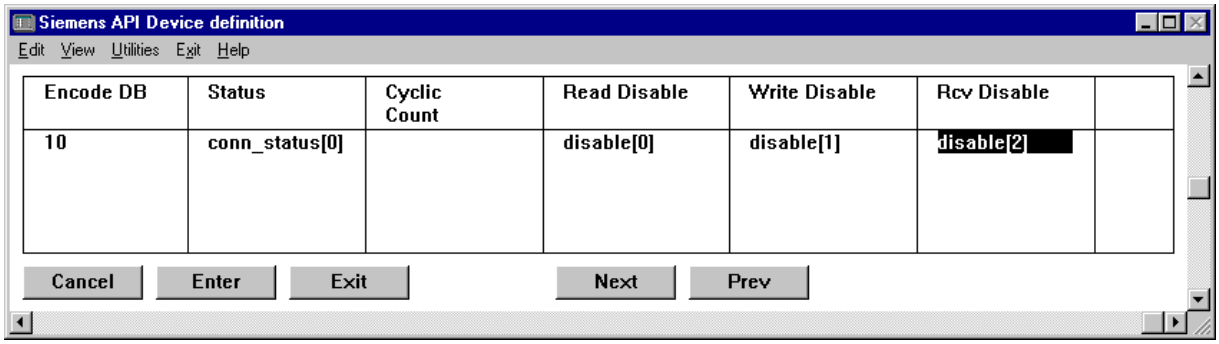


Figure 4.3.2 DeltaLink SAPI S7 Device Definition.

Status

The S7 driver uses this tag to update the status of the connection with the PLC. A value of zero indicates no error, any other values represents an error.

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

Cyclic Count

The cyclic count element holds the value of the total number of cyclic count currently active. With this value it can be checked if all values are executing or not.

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

Read disable

Real-time digital database element used to enable/disable read commands for this logical device. Read commands are enabled if there is no tag defined, or the status of the digital tag is OFF. Read commands are disabled if a tag is defined and 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 for the logical device. Write commands are enabled if there is no tag defined, or the status of the digital tag is OFF. Write commands are disabled if a tag is defined and the status of the tag is ON.

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

Receive disable

Real-time digital database element used to enable/disable receive commands for the logical device. Receive commands are enabled if there is no tag defined, or the status of the digital tag is OFF. Receive commands are disabled if a tag is defined and the status of the tag is ON.

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

24 Dataset definition

From the display of all the panels, select the *Siemens SAPI Dataset definition* panel. This panel allows the user to specify datasets for a particular logical device. Different logical device names are selected in the panel *Siemens SAPI Device Definition*. The name of the logical device is displayed in the field *Device Name*, at the bottom of the table.

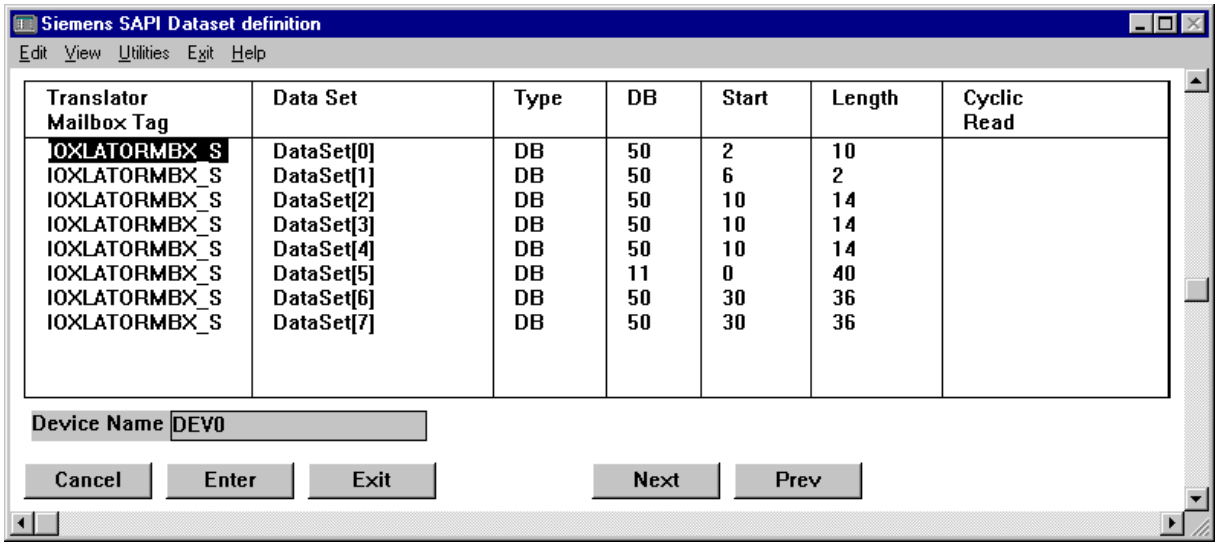


Figure 4.4.1 Siemens SAPI Dataset definition.

- **Translator Mailbox Tag**

  Translator reply mailbox, the S7 driver will reply IMX responses from this dataset into this mailbox (**N.B.** IMX must be supported by the translator task). The translator task will read this mailbox in order to receive IMX responses.

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

- **Data Set**

  Specify the dataset control tag in this field. This tag will be used by IMX to uniquely identify this dataset. This tag name will also be referenced in the translator task. This tag must be unique and only be used to identify this dataset throughout the whole application.

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

Type

Data type definition of the data area in the PLC. Note that exception write behaviour depends on the type of data area. To know how an exception write is executed, one should regard the boundary of the data area and the element size involved with the exception write.

entry Required.
entry type Alphanumeric string.
valid entries DB, FB, IB, OB, PB, CW, TW, SW, DX, DE, EB.

Siemens S7data type	
Data type entry	Description
DB	Data Block
FB	Flag byte
IB	Input byte
OB	Output byte
PB	Peripheral byte
CW	Counter word
TW	Timer word
SW	System Data word
DX	Expanded Data Block
DE	Datablock in extended memory
EB	Expanded Peripheral byte

DB

Data block number for the datatypes: DB, DE or DX.

entry Optional.
entry type Decimal number.
valid entry Positive integer 1 ..255.

Start

The starting address of the dataset.

entry Required / Default 0.
entry type Decimal number.
valid entry Positive integer 0 ..2048.

Length

The total length of the dataset. This length can be in bytes or in words depending on the data type. For example the length of a dataset will be in words and the length of flag bytes will be in bytes. The reason for this is because a datablock has a word boundary and flag bytes a byte boundary.

entry Required / Default 1.
entry type Decimal number.
valid entry Positive integer 1 ..2048.

♦ Cyclic Read

The cyclic read functionality can be started and stopped via the Cyclic Read tag. In case of starting this tag must hold the interval value with which the CP in the PLC will update the data. The update rate will be set if the change flag of this tag changes. To stop the cyclic read a value of zero must be written into this tag. The update rate is in tenths of seconds for example a value of 10 means 1-second update rate.

entry	Required / Default 1.
entry type	Decimal number.
valid entry	Positive integer 1 ..2048.

Appendix A. The delta.opt file

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

The authorisation sequence of every independent DeltaLink module must reside in the {FLOPT}/delta.opt file in order to take effect. If the task has been ordered with an authorisation 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. If the task has been previously installed from a demo diskette and the authorisation 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 authorisation 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 1998 DeltaLink bv. All Rights Reserved
*
* DeltaLink bv
* Pioenroosstraat 26
* NL-5241 AB Rosmalen
* The Netherlands
* Tel (int) 31 73 5231234
* Fax (int) 31 73 5231222
*
*
* FactoryLink Serial Number 12345NTI
*
* DeltaLink module option
*
* MODULE = taskname of always 8 characters with trailing spaces
* .      = <space>
* X      = authorisation code supplied by DeltaLink
*
* MODULE..XXXX.XXXX.XXXX.XXXX.XXXX.XXXX.XXXX.XXXX.XXXX
SAPI__S7 2D6F E047 2534 036E 1215 EC80 92F8 1EFC C4C0
```

If a full installation diskette with authorisation 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 authorisation code must exactly match the format as listed in the header. If this is not the case the module will not recognise the authorisation sequence and start up in demo mode.

The format of an authorisation 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 SAPI__S7. 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 authorisation 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 authorisation 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 authorisation sequence a normal editor can be used. If all modules with the right authorisation 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 9. If no number is specified the default level will be 1. The debug output will be visible in the 'window' of the protocol driver, or 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/sinec_h1.log
-v*n*	Debug option, the level of debug information is set with the number *n*. The range of this number is from 1 until 9. If no number is specified the default level will be 1. The dubug information is sent to the standard debug output of the Microsoft operating system Microsoft supports debugging of applications with a 'DebugViewer', an application that is capable of showing, filtering and saving debug information. In your shipment is included a debug viewer application.
-LS*n*	Local Station id with number *n*, needed if the Ioxlator and protocol driver reside on different nodes (or FactoryLink workstations). This must be an unique number for the network.
-b	The option – installs a crash handler for the protocol driver. The crash handler will pop up in case of a rash, and will give several options to handle the information collected prior to the crash.

Appendix C. S7 error codes

These error codes will be generated by Softnet and will be returned to the S7 driver which will in turn display them. For more detailed information about these error codes turn to the Softnet documentation.

Error #	general errors
0	No error
1	Unknown error
2	Wrong CP description
3	No resources available
7	Invalid parameter
8	Too long data
9	Too many DLL users
10	Wrong indication confirmation
11	Service not supported
20	Invalid configuration reference
23	Connection name not found
25	no connection database file
30	invalid order id
31	order id already used
40	invalid remote order id
41	remote order id already used
42	no remote order id
50	object undefined
51	object attributes inconsistent
53	object access denied
80	invalid data size
81	receive buffer is full
82	invalid data range or type
83	invalid segment
90	controller or CMI or FW problems
100	error mini DB type
101	error mini DB value
112	service VFD already used
113	service connection already used
120	connection arboreta
121	invalid connection state
122	maximum number of requests reached
123	error connection confirmation
130	invalid cyclic read state
140	error installing
141	internal error
142	no sinec service
143	no license installed
150	error symbolic address
151	symbolic address inconsistent
160	error remote brecv
161	error remote bsend
162	remote bsend cancelled
163	remote database too small
170	error no rcv block

The following errors are reported by the S7 driver.

Error #	H1 data package errors
201	Unidentified data received.
202	Encode value not supported.
203	Unknown encode type.
204	The size of the H1 packet was too small
205	Error in received H1 protocol header

Error #	H1 data package errors
300	No error.
301	Incorrect Q/Z type at handling block.
302	Area not present in AG.
303	Area in AG too small.
304	AKD Error in the AG.
305	Error with condition code word.
306	No valid ORG format.
307	No free data buffer.
308	No free transport links.
309	Remote error with Read/Write.
310	Data Link error.
311	Handshake error.
312	Initiation error.
313	Abort after reset.
314	Job with bootstrap function.
315	Job not present.
316	Area in AG too small.

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

Error #	FactoryLink errors
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 Ioxlator
487	Max. No. of messages in Ioxlator MBX reached

Error #	S7 driver errors
850	read functionality has been disabled
851	write functionality has been disabled
852	unsolicited receive has been disabled
853	invalid addressing method on bit boundary
854	exception write not possible to destination
855	S7 driver has shutdown
856	an unknown error occured in the comm layer
857	no data received
859	encoded write is not possible
861	no connection established
862	communication board not defined in Softnet
863	connection not defined in Softnet
864	invalid cyclic read interval
865	error starting the cyclic read cycle

Error #	S7 driver errors
866	error stopping the cyclic read cycle
867	maximum of cyclic reads reached
870	wrong response received with request
871	Request timed out
880	No VFD found on the specified board
881	The specified VFD is not found on the specified board

Appendix D. Messages

If an error condition occurs in the protocol driver task during run-time mode, a message to that effect will appear on the runtime manager graphics screen to the right of "SAPI_S7". The error messages that may be displayed are as follows

Running in DEMO mode

The task did not find the DeltaLink authorization code at start up. This is a non fatal error, however the task continues running in DEMO mode. The DEMO mode has a limitation timed for 5 hours. After shutdown the task can not be restarted without stopping the complete FactoryLink application.

Demo shutdown, licensed to run 5 hours

The task did not find the DeltaLink authorization code, when starting up and continued running in DEMO mode. The DEMO has a limitation timed for 5 hours. After shutdown the task can not be restarted without stopping the complete FactoryLink application.

Demo restart prohibited, restart FactoryLink

The task did not find the DeltaLink protection code, when starting up. The option file could not be present or damaged. This is a non fatal error, however the task continues running in DEMO mode. The DEMO has a limitation timed for 5 hours. After shutdown the task can not be restarted without stopping the complete FactoryLink application.

Out of RAM

There is not enough RAM memory to load the complete configuration and/or task. This is a fatal error.

Error (%d) reading RTDB element

An error occurred reading a tag value. The error number is displayed.

Error (%d) forcing RTDB element

An error occurred forcing a tag value. The error number is displayed.

Error (%d) writing RTDB element

An error occurred writing a tag value. The error number is displayed.

Error (%d) retrieving event

An error occurred while waiting for an event to happen. The error number is displayed.

Can't open CT file

The task was unable to open the configuration table file, generally because it does not exist. This is a fatal error. Check the *$FLINK/ctgen/ctlist* file fo the sapi_s7 entry or try to generate the file with the *ctgen* utility

No triggers defined

The task was able to open the configuration table file, but contained not enough information to continue running. This is a fatal error.

Error reading CT index

An error occurred reading the index of the CT file. The sapi_s7 ct file is probably corrupted, try to generate new one with the *ctgen* utility. This is a fatal error.

Error reading CT header

An error occurred reading the header of a table in the CT file. The sapi_s7 ct file is probably corrupted, try to generate new one with the *ctgen* utility. This is a fatal error.

Error reading CT record

An error occurred reading a record of a table in the CT file. The sapi_s7 ct file is probably corrupted, try to generate new one with the *ctgen* utility. This is a fatal error.

Initialization critical parameters failed

Use of critical sections will not be possible, but is still required for full functionality of the protocol driver. Try stopping the complete appliaction and start up again. This is a fatal error.

Event not registered to FL kernel

Tag event is not registered to the FactoryLink kernel. This is a fatal error.

No loxlators defined

There are no loxlators (or mailbox) defined. Ckeck the configuartion panels of the S7 driver. This is a fatal error.

No datasets defined

No dataset is defined, the protocol driver has nothing to do. Therefore this considered to be a fatal error.

Unknown loxlator table %s record %d type %d

The protocol driver configuration table file contains an unknown loxlator mailbox tag, for the specified table, record number and type of dataset. This is a fatal error.

IMX Bad Mailbox TAG table %s record %d type %d

IMX error, internal error of the protocol driver task. Specified are table number, record number and type number.

IMX dataset not unique %s

IMX error, dataset control tag is not unique, the name of the dataset control tag is specified. This is a fatal error, a dataset control tag can only be used once, to define a dataset!

IMX (%d) Initialization failed

IMX initialization failed, the error number is reported. This is a fatal error.

IMX (%d) dataset registration failed %s

Registration of the reported dataset failed, the cause can be determined by the error number. This is a fatal error.

IMX semaphore creation failed

Creation of semaphores, needed for IMX failed. This is a fatal error.

IMX failed to start thread

The protocol driver could not start a thread. This is a fatal error.

IMX maximum msg for loxlator %d reached

The reported loxlator failed to process the queued messages, the configured maximum length of the queue is reached. The protocol driver stops queuing until the number decreases.

IMX (%d) send error to mailbox %s

The protocol driver could not queue a message for the reported loxlator due to an error, the corresponding error number is reported.

IMX (%d) reply to mailbox %s

The protocol driver could not queue a message for the reported loxlator due to an error, the corresponding error number is reported.

Com building device failed

Communication error building of a communication device failed. This is a fatal error.

Com unknown DataSet of type %d

Unknown dataset type specified, the type invalid type id is reported. This is a fatal error.

Com init comm. layer %d
An error occurred during initialising the communication layer Specified is the error number.

Com (%d) connecting '%s' %s
An error occurred during connecting to a device. Specified are the device name, functionality and the error number.

Com (%d) listenening %s %s
An error occurred during listening. Specified is the error number.

Com (%d) accepting connection %s
An error occurred during accepting a connection to a device. Specified are the device name and the error number.

Com (%d) disconnecting '%s' %d
An error occurred during disconnecting from a device. Specified are the device name, functionality and the error number.

Com (%d) read %s
An error occurred during reading from a device. Specified are the device name and the error number.

Com (%d) reading for exception %s
An error occurred during reading from a device for an exception write. Specified are the device name and the error number.

Com (%d) encode write %s
An error occurred during writing for an encoded write to a device. Specified are the device name and the error number.

Com (%d) write %s
An error occurred during writing to a device. Specified are the device name and the error number.

Com (%d) receive %s
An error occurred during receiving from a device. Specified are the device name and the error number.

Com (%d) reject connection %s, %s
An incoming connection request is rejected. Specified is the error number.

Appendix E. Configuring Softnet

The S7 driver uses Softnet as the protocol stack for communication with the PLC's. In order to use Softnet with the S7 driver it has to be configured. First step in configuring softnet is to point out which adapter and protocol to use. This must be done via the Setting the PG/PC Interface utility of Softnet.

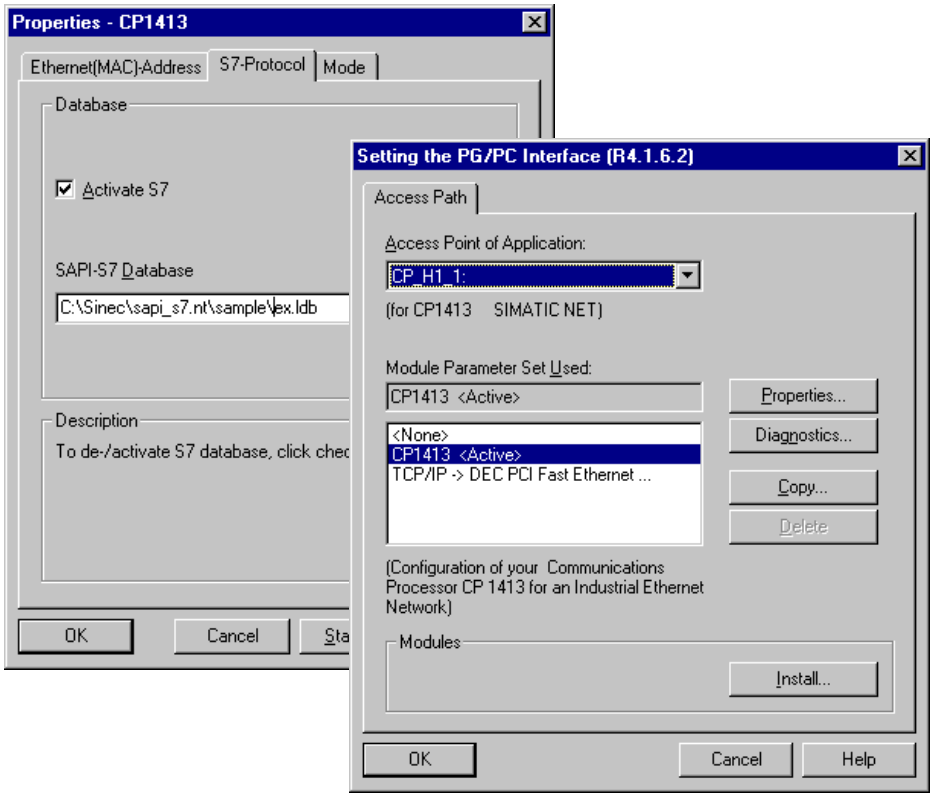


Figure E.1 Setting the PG/PC Interface utility

The install button allows you to choose a communication adapter and protocol first. When the install button is pressed then the window as shown in figure E.2 will pop up. Add here the board and protocol that you will be using. Then close the window again.

In the main window choose now the right protocol adapter combination in the *Module Parameter Set Used* list box and press the *Properties* button. The back window of figure E.1 will pop up. At the *S7-Protocol* tab check *Activate S7*. At the *SAPI-S7 Database* entry a configuration database which is generated with the *COML S7* must be specified. At this moment this database doesn't exist because we didn't configure it yet. The future database location and name can be configured here or it can be configured later.

The value listed at Access Point of Application is very important because it has to be referenced in the S7 driver configuration. This value must be specified in the panel *Siemens SAPI Device Definition* at the *Board* field (see section 4.3).

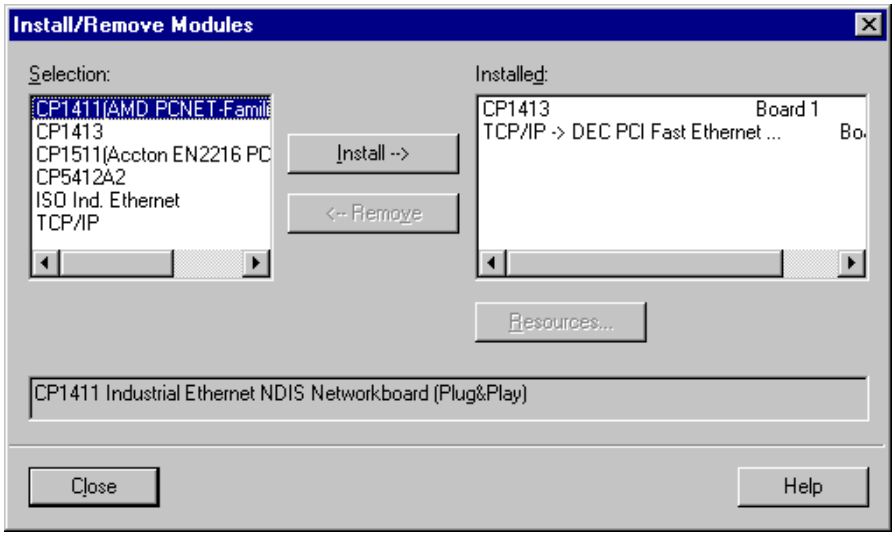


Figure E.2 Installing adapter and protocol

Addressing information per PLC has to be configured after setting the PG/PC interface. This must be done with COML S7 utility from Softnet. Per plc on logical connection must be configured as shown in figure E.3.

At the section *Node name* the name of the connection database must be specified. This must be the same name as the *SAPI S7 database* in figure E.1. At *Network type* the same type of protocol must be specified as in figure E.2. Next step is to configure one logical connection for each PLC on the network. This must be done on the right side of the window. The value at the name section is very important because it has to be referenced in the S7 driver. This value must be specified in the panel *Siemens SAPI Device Definition* at the *Connection* field (see section 4.3) and is the relation of the addressing information of the PLC.

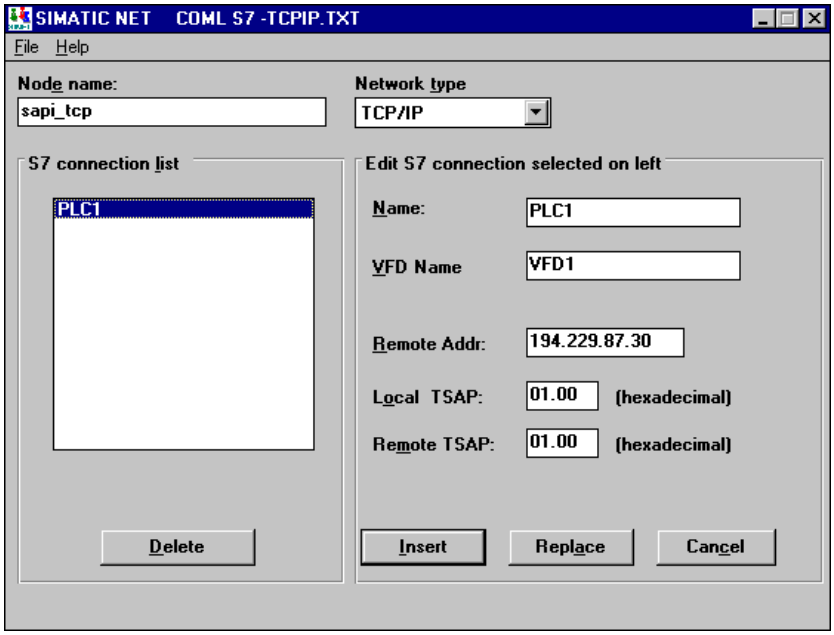


Figure E.3 COML S7 utility

Appendix F. Encoded write lay-out

The S7 driver is capable to perform an encoded write. An encoded write causes the protocol driver to place coded information in a data block in the PLC instead of directly updating or changing a variable.

The information, placed in the PLC by an encoded write, should be decoded by a PLC-program in order to update/change the correct variable. A standard function block (FB10) is provided by DeltaLink for decode actions inside the PLC. The standard function block FB10 is called "ENCODED", and uses SFC20 and an instance data block. The instance data block should contain the information from an encoded write. The protocol driver places the information in a special way in the encode DB.

Encode DB		
Data byte	Type	Description
DBB0	KF	Reserved
DBB1	KF	Reserved
DBB2	KY	Destination
DBB3	KY	destination type
DBB4+5	KY	DB no..
DBB6+7	KY	address no.
DBB8	KY	Bit no.
DBB9	KY	Change flags
DBB10	KY	Value
DBB11	KY	Value
DBB12	KY	Value
DBB13	KY	Value

Figure F.1 Encode Data Block.

Destination the information in this byte indicates the destination of the value which should be updated.

Value		Description
M	=	Flag type.
D	=	Data block.
A	=	Output.
E	=	Input.

Destination type the information in this byte indicates the destination type of the value which should be updated.

Value		Description
F	=	Flag for DB, F, A or E.
B	=	Byte for DB, F, A or E.
W	=	Word for DB, F, A or E.
D	=	Double word, for long integer or float.

DB number The information in this byte is the number of a DB. If the destination is not D the value in this byte is irrelevant. The value is between 0 and 4095.

Address number address number of the byte, word, data word, timer or counter. The value is between 0 and 4095.

Bit number the number of the bit in a word or a flag. If the destination type is not flag, the value is irrelevant.

Change flags For every encoded write the protocol driver writes the value 255 in this byte. Function block FB10 uses this value to detect if there is new information.

Value value may be up to longs, depending on the type of data. For flag and byte values DBB16 is used. For word values DBW10 is used. And for long integers and float DBD8 plus DBW10 is used.

FB10 ENCODED

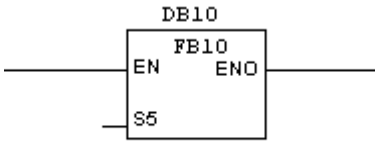


Figure F.2.2 Function Block FB10.

Parameters

Instance DB	The function block needs an instance DB, specify the desired block number.
S5	Boolean to specify if S5 addressing mode is to be used. A value of TRUE means that the S5 addressing mode is used. S7 addressing mode is used when the value is FALSE.

Appendix F. Debug Viewer

Introduction

DebugView is an application that lets you monitor debug output on your local system, or any computer on the network that you can reach via TCP/IP. It is capable of displaying both kernel-mode and Win32 debug output, so you don't need a debugger to catch the debug output your applications or device drivers generate, nor do you need to modify your applications or drivers to use non-standard debug output APIs.

DebugView works on Windows 95, 98, Me, 2000, XP, Windows Server 2003, Windows for x64 processors and Windows Vista.

DebugView Capture

Under Windows 95, 98, and Me DebugView will capture output from the following sources:

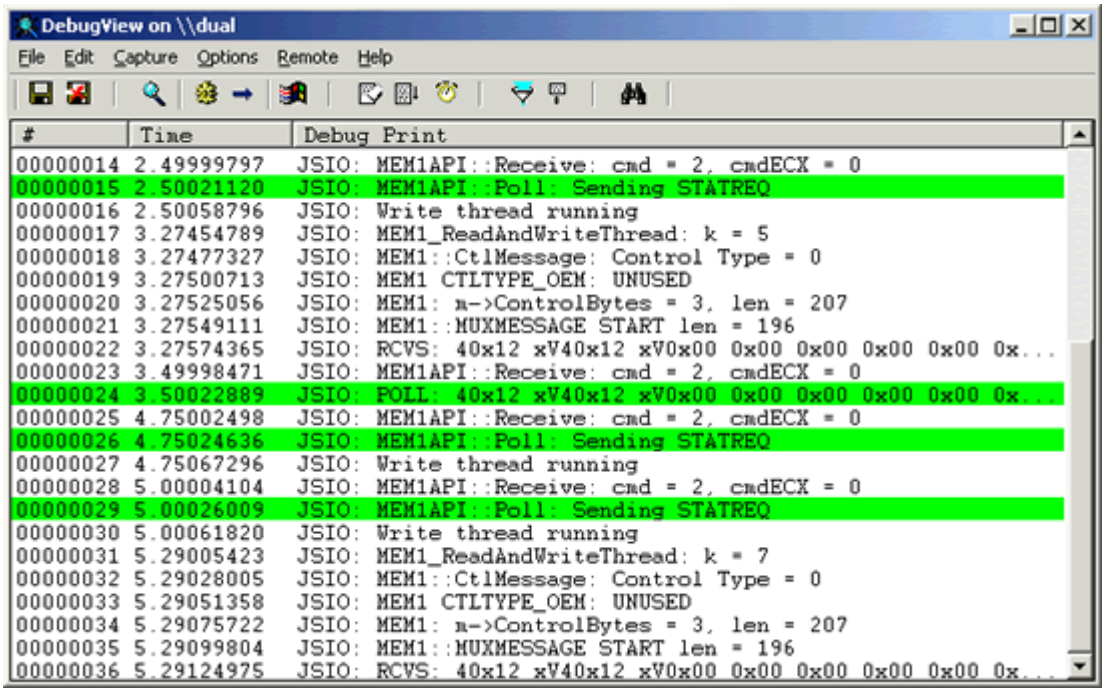
- Win32 OutputDebugString
- Win16 OutputDebugString
- Kernel-mode Out_Debug_String
- Kernel-mode _Debug_Printf_Service

Under Windows NT, 2000, XP, Server 2003 and Vista DebugView will capture:
- Win32 OutputDebugString
- Kernel-mode DbgPrint
- All kernel-mode variants of DbgPrint implemented in Windows XP and Server 2003

DebugView also extracts kernel-mode debug output generated before a crash from Window NT/2000/XP crash dump files if DebugView was capturing at the time of the crash.

Installation and Use

Simply execute the DebugView program file (dbgview.exe) and DebugView will immediately start capturing debug output. This is a screenshot of DebugView capturing Win32 debug output from a remote system. Note the presence of a highlighting filter.



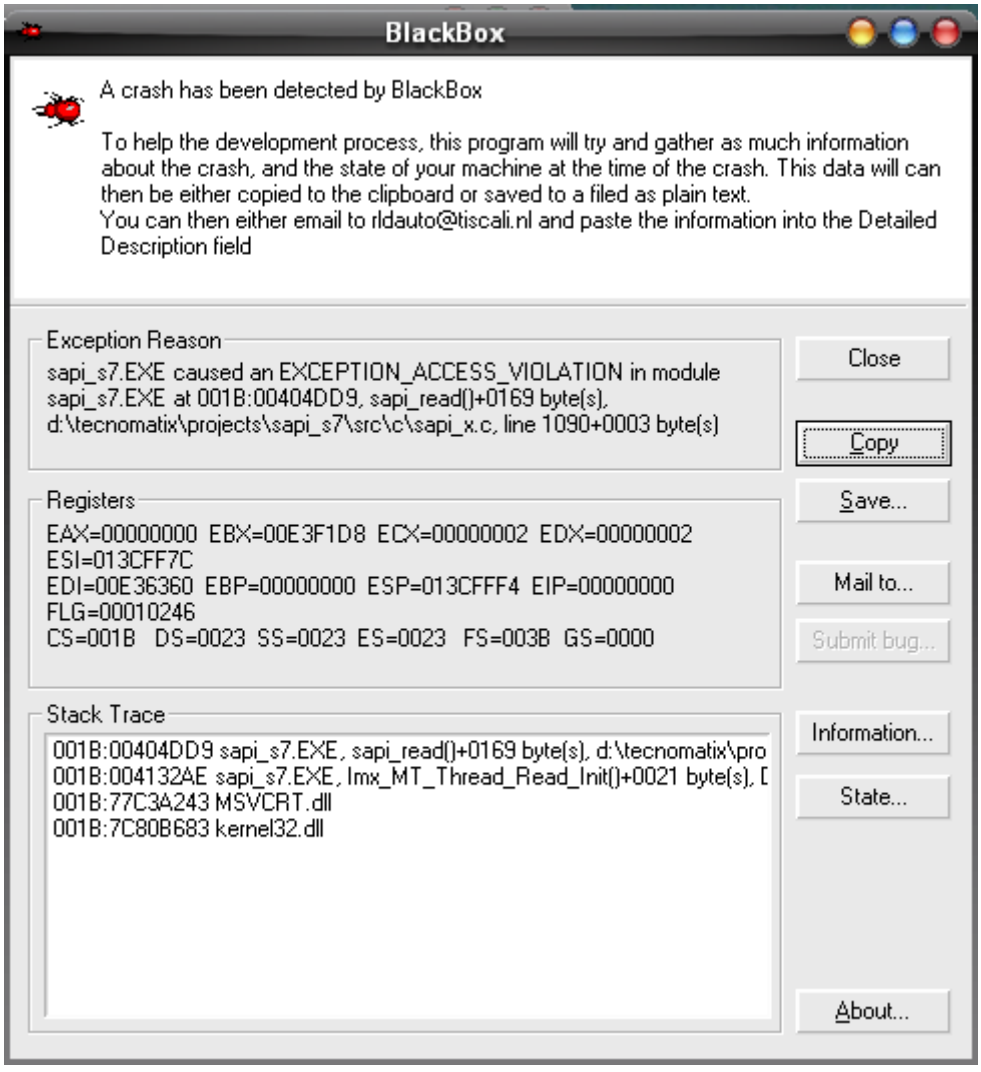
To run *DebugView* on Windows 95 you must get the WinSock2 update from Microsoft. Note that if you run *DebugView* on Windows NT/2K/XP you must have administrative privilege to view kernel-mode debug output. Menus, hot-keys, or toolbar buttons can be used to clear the window, save the

monitored data to a file, search output, change the window font, and more. The on-line help describes all of *DebugView*'s features.

Appendix G. Blackbox crash handler

Introduction

BlackBox is a dll that you simply load up and then forget about. It will set a crash handler as soon as the library is loaded (in DLLMain()), and then just waits patiently for your application/driver to crash. If and/or when your application/driver crashes, instead of just displaying an error message like "Error writing to memory address 0XDEADBEEF", a nice little dialog comes up that displays the information in a fashion similar to Netscape's TalkBack, or the new error reporting dialog that pops up with the newer version of Microsoft's Internet Explorer. This dialog allows the user to then copy or save the crash data, and then email or be taken to an appropriate website where they can submit the information. This information can then be used by a programmer to have a much better idea of exactly where the problem occurred, as well as information about the machine that it crashed on.



Usage

BlackBox displays a variety of information:

- The Exception Reason
- The Registers
- The Stack Trace, a complete stack trace from where the error occured
- The machine information, such as CPU, OS version, physical memory, etc. Clicking on the "Information..." button will display this in a dialog.

- The machine state - a list of ALL the processes running at the time of the crash, and any DLLs loaded by the processes. Clicking on the "State..." button will display this in a dialog.

As a user, you can copy the contents to the clipboard by clicking on the "Copy" button. Clicking the "Save..." button will prompt you to save the contents to a file, with the default name being of the form: "errorlog-<GMT time, dd-mm-yyyy>(<time, hh:mm:ss>).log".

Click the "Mail To..." button and an attempt will be made to execute the mailto: command on your system. Assuming a properly installed email program, it will be brought up with the appropriate email address.

