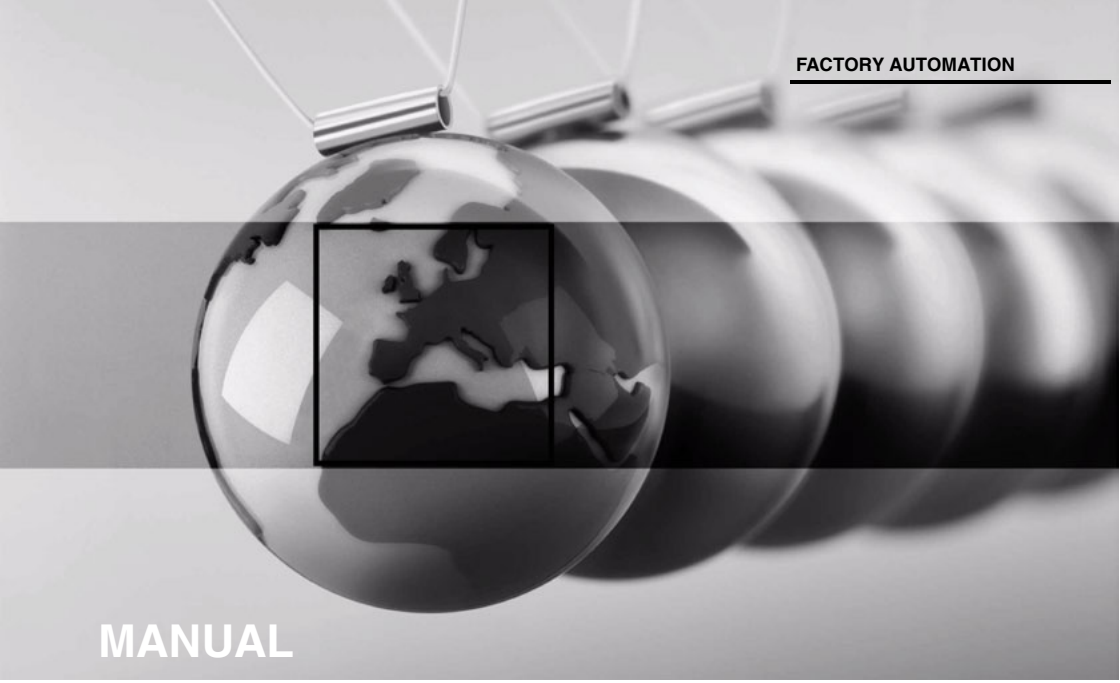


MANUAL

IC-KP-B12-V45

Implementation to a SIMATIC S-7 400 PLC

PEPPERL+FUCHS
SENSING YOUR NEEDS

With regard to the supply of products, the current issue of the following document is applicable: The General Terms of Delivery for Products and Services of the Electrical Industry, published by the Central Association of the Electrical Industry (Zentralverband Elektrotechnik und Elektroindustrie (ZVEI) e.V.) in its most recent version as well as the supplementary clause: "Expanded reservation of proprietorship"

2011-01

1 Safety

1.1 Symbols relevant to safety

Danger!
This symbol indicates a warning about a possible danger.

In case of ignoring the consequences may range from personal injury to death.

Warning!
This symbol indicates a warning about a possible fault or danger.

In the event the warning is ignored, the consequences may course personal injury or heaviest property damage.

Caution!
This symbol warns of a possible fault.

In case of ignoring the devices and any connected facilities or systems may be interrupted or fail completely.

1.2 Intended use

The IDENTControl IC-KP-B12-V45 is a control interface including an Ethernet interface for identification systems. The device can be used as a control cabinet module or for field applications. Besides the Ethernet connection, suitable inductive R/W heads, microwave antennas or trigger sensors can be connected. Wiring suitable for the system design must be used.

1.3 General notes on safety

Only instructed specialist staff may operate the device in accordance with the operating manual.

Independent interventions and separate modifications are dangerous and will void the warranty and exclude the manufacturer from any liability. If serious faults occur, stop using the device. Secure the device against inadvertent operation. In the event of repairs, send the device to Pepperl+Fuchs.

The connection of the device and maintenance work when live may only be carried out by a qualified electrical specialist.

The operating company bears responsibility for observing locally applicable safety regulations.

Store the not used device in the original packaging. This offers the device optimal protection against impact and moisture.

Ensure that the ambient conditions comply with regulations.

2011-01

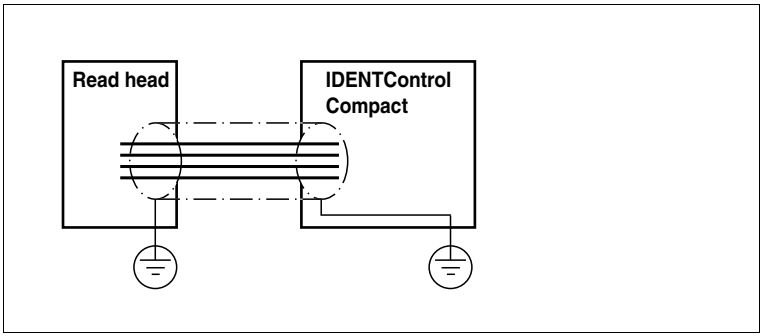
Note!

Disposal

Electronic waste is hazardous waste. When disposing of the equipment, observe the current statutory requirements in the respective country of use, as well as local regulations.

1.4 Contact protection

Our housings are manufactured using components made partly or completely from metal to improve noise immunity.



Danger!

Electric shock

The metallic housing components are connected to ground to protect against dangerous voltages that may occur in the event of a fault in the SELV power supply!

2 Installation

2.1 Equipment and devices

In the following table you can find all components which are necessary to connect the identification system IDENTControl IC-KP-B6 to SIMATIC S7-400 PLC with Ethernet interface.

Equipment	Manufacturers name
Simatic S-7 400 power supply	PS 407 4A
Simatic S-7 400 CPU	CPU-412-2
Simatic S-7 400 communication processor	CP 443-1
Control interface unit IDENTControl	IC-KP-B12-V45
4 x read / write heads	IQH-18GM-V1
4 x connection cable read / write heads	V1-G-0,6M-PUR-V1-W
Network cable	V45-G-10M-V45-G
Data carrier	IQC21-58
1 x connection cable IDENTControl	V1-G-2M-PUR

Table 2.1: Used hardware equipment

The equipment is an extract of the test composition explained in this manual. The commissioning of the IDENTControl IC-KP-B12-V15 can also be execute by other PLC devices with the same functionality. You can get information of the installation of the PLC devices out of the manuals of the PLC.

2.2 Configuration and Installation

Following graphic shows the installation of the hardware equipment.

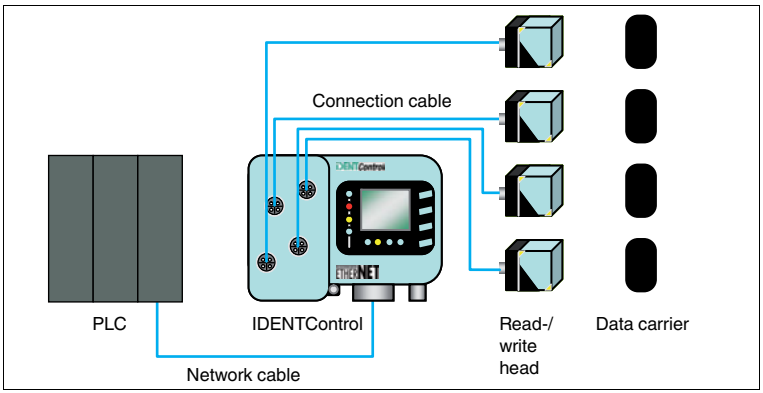


Figure 2.1: Installation of the hardware equipment

2011-01

Installation

1. Firstly connect the cables of the read/write heads to the associated females of the IDENTControl.
2. Consider the distance between two heads because the heads influence eacht other.
3. Take the accurate distance out of the data sheet of the head.
4. Consider that the data carriers are located in the detection range of the read/write heads. But take also care that there is only **one** carrier inside the detection range of the head.
5. Afterwards connect the IDENTControl to the power supply 20…30 V_{DC}.
6. Finally establish a communication access between IDENTControl and PLC by the Ethernet Network cable.

Note!

You don't get more information about the build-up of the different PLC equipment. For more information about the commissioning see the manual "PLC S7-400 Installation guide". For more information about the installation of the IDENTControl see the "Manual IC-KP-B12-V45" on www.pepperl-fuchs.com.

2.3 Configuration of the PLC

In this chapter the description of the Hardware configuration takes place inside the Software “SIMATIC Manager“.

The configuration is shown with an example.

Adjust the parameters of the operational environment for their specific configuration.

Set the parameters directly on the control panel buttons of the IDENTControl.

To parameterize the IDENTControl see the "Manual IC-KP-B12-V45" on www.pepperl-fuchs.com.

Consider that there are already set default parameters in the delivery status of the IDENTControl. You can get this default setting out of the "Manual IC-KP-B12-V45", too.

The following table is a list of the example's setted parameter.

Parameter	Value
IP address IDENTControl	172.16.11.25
IP address S7-400 PLC	172.16.11.26
Net mask	255.255.0.0
IP address Router	172.16.11.222

Table 2.2: Parameter of hardware configuration

Commissioning PLC

1. For commissioning the IDENTControl you have to start a new project.
2. After the start of the new project the components of the PLC have to involve into the Hardware Configuration.
3. Firstly the SIMATIK 400 Station has to be implementing.

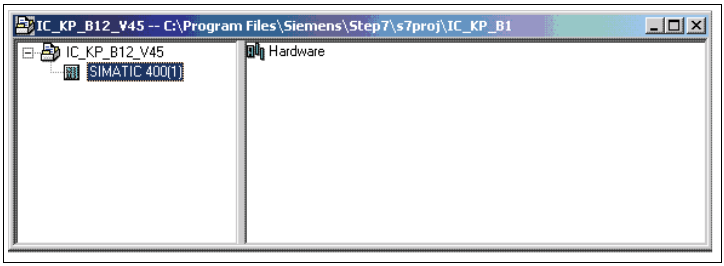


Figure 2.2: Integration SIMATIC400 station

4. Afterwards the hardware configuration is called up by double click on the symbol “Hardware”.
5. Afterwards the hardware configuration is called up by double click on the symbol “Hardware”. Insert the required components rack, power supply, CPU as soon as communication processor into the hardware configuration.

   ↳ The following picture shows the involvement of the hardware components.

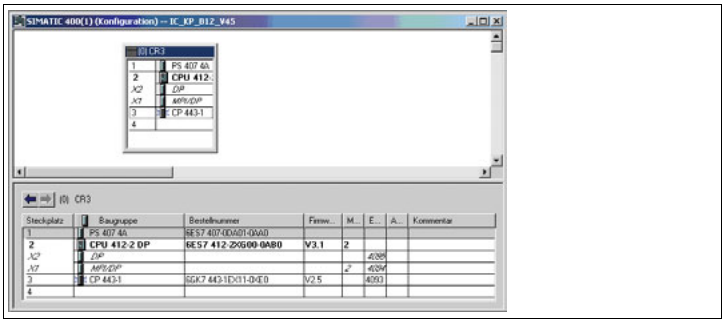


Figure 2.3: Hardware configuration

6. In the next step the attributes of the communication interface of the communication processor CP443-1 must be defined. For this you must open the attributes window of the CP443-1. The window opens by double click of the associated symbol in the picture above.
7. In the window the active attributes are shown. To change the active attributes you must click the button "Attributes“.

   ↳ A window opens. In this window the parameter of the Ethernet interface of the communication CP adjusted.

2011-01

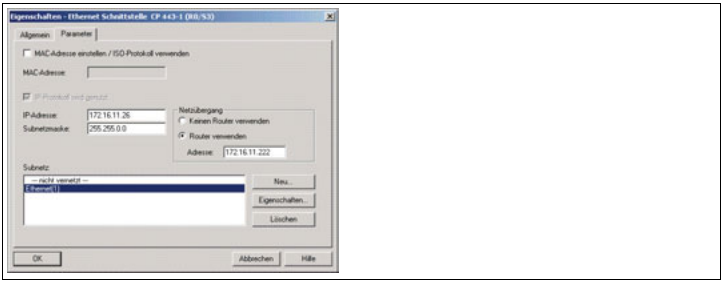


Figure 2.4: Adjustment Ethernet interface PLC

8. The appropriate IP address and the subnet mask must be defined.
9. The net gateway must be defined. The gateway is realized by a Router and the appropriate label must be selected.
10. The address of the Router has to insert in the intended field.
11. The subnet have to be networked. The new adjusted parameters memorised by the button “OK”.
12. The IDENTControl has to be connected to the communication network. The connection is realised by the assortment “Other Station“. The insertion of the “Other Station” can either made inside the SIMATIC-Manager or inside the net configuration NetPro.
13. Inside the net configuration the item "Other Station“ have to be insert in the field of the communication members. The item can name accorded to the IDENTControl "IC-KP-B12-V45“.

    ↳ Thereby following view accrued.

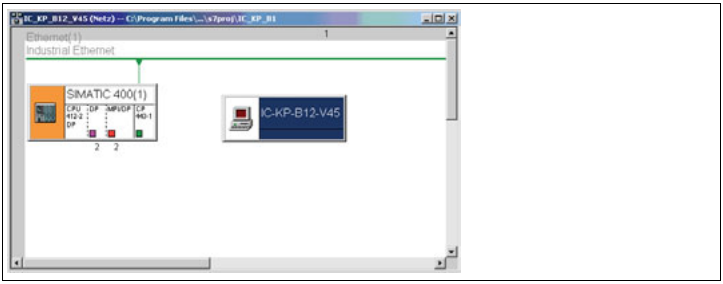


Figure 2.5: Integrating IDENTControl inside netconfiguration NetPro

14. By double click to the symbol “IC-KP-B12” you reach inside the menu to adjust the communication.
15. With the slider “Interface” you can define the interface of the IDENTControl. For this you must select with the button "New" the interface type "Industrial Ethernet“.

↳ Afterwards a window opens where the IP address of IDENTControl and Router can be defined. For the adjustment it applies to consider that the Ethernet subnet with the button “New” is connected.

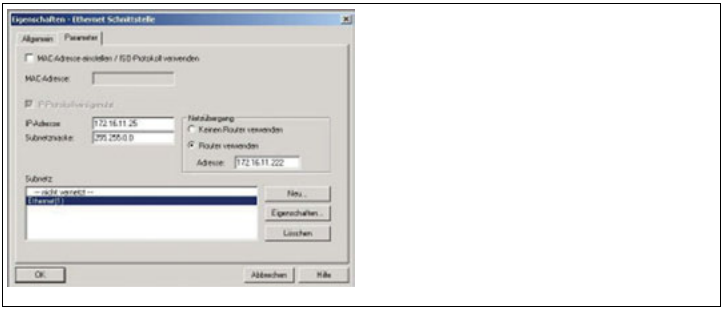


Figure 2.6: Adjustment of Ethernet interface IDENTControl

16. The attributes of the net are active by click on the button "OK“.
17. Inside the net configuration the new network connection is to insert in the connection table of the CPU. The connection table is inside the net configuration NetPro under the field of the communication members.

Figure 2.7: Net configuration with connection table

18. By click to the right on the first line of the connection table you reach inside a menu selection. Here you must chose the selection “Insert new connection“and click on the symbol.

↳ Afterwards following window opens.

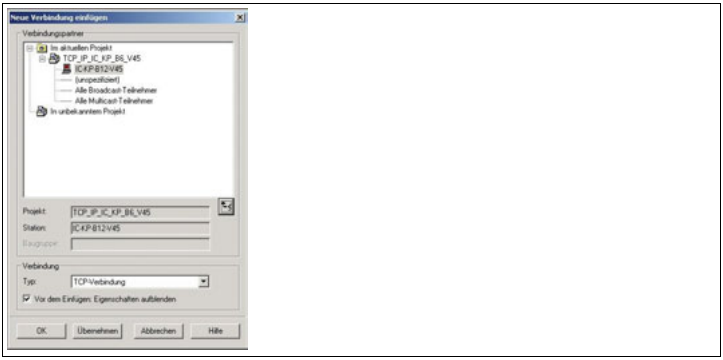


Figure 2.8: Communication members

19. Adjust the members of communication.
20. A member of the communication network is “Other Station“ or “IC-KP-B12-V45“.
21. The connection type is "TCP-Connection“.
22. The adjusted parameters are accepted by click on the button "Accepting“.

    ↳ Than a window “Attributes TCP-Connection” opens. With the slider “general“ the activated communication number is shown.
23. Open the slider "Address“.

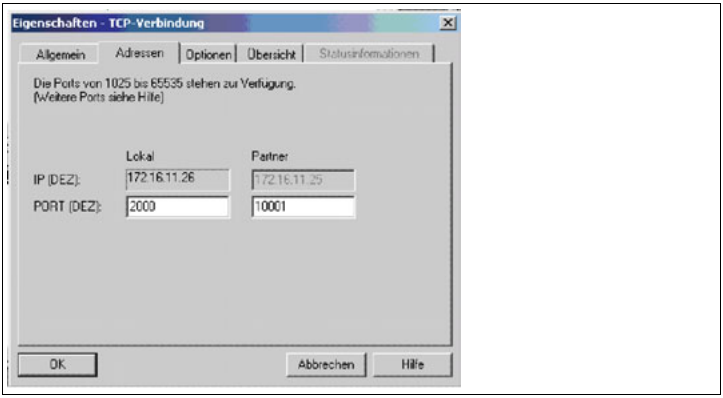


Figure 2.9: Adjustment port number

24. The port numbers of the communication members must be defined. The IDENTControl have got two different port numbers 10000 and 10001. If you chose port 10000 the length of a telegram is 34 Byte. And if you select port 10001 the telegram length is 66 Bytes.

    ↳ Now the hardware configuration is finished.

3 Commands

3.1 Types of Commands

There are two types of commands of the IDENTControl executed by the PLC program. You can differentiate between a single command and an enhanced command.

Single command

If there is a code/data carrier in the detection range of the read/write head, the command execution will only take place once.

If there isn't any code/data carrier in the detection range of the read/write head, the command execution will be retried by the PLC program until the command is completely executed or the maximum number of retries is exceeded.

Note!

The maximum number of retries has to be set by the user in order to prevent a hangup of the read/write head caused by the continuing command execution.

Enhanced command

The command execution, which is necessary for different applications, will be retried by the PLC program until the process will be canceled by a Quit command.

3.2 Command Cycle

The PLC program transmit command parametres which are necessary for the execution of a command to the IDENTControl.

The IDENTControl accepts this command and sends a status response (FF)h back to the PLC.

The status response signalize that the IDENTControl is accepting and editing the command.

Afterwards the IDENTControl transmits the command to the specific read/write head and waits for a response of the head.

The response of the head passes to the PLC by the IDENTControl.

Received files will be evaluated within the PLC and made available to the user.

The IDENTControl transfers replies to the PLC during the implementation of an enhanced-command as long as the command will be canceled.

3.3 Command Structure

Before the IDENTControl can execute a command you have to transmit the command parameter to the IDENTControl. The command parameters are structured byte by byte and form a command telegram. The number of transmitted parameters and the command code is different but the command structure is always the same. The following chart shows the structure of the command telegram of a single read data command. This telegram will be transmitting by the PLC to the IDENTControl and cause the import of Data.

Byte	Contents / Variable	Bit Allocation							
0	#Head_X.OutData.TelegrammLengthHigh	0	0	0	0	0	0	0	0
1	#Head_X.OutData.TelegrammLengthLow	0	1	0	0	0	0	1	0
2	#Head_X.OutData.CommandCode	0	0	0	1	0	0	0	0
3	#Head_X.OutData.Channel	Number data words				Channel number			
4	#Head_X.OutData.Wordadr_High	Start address high Byte							
5	#Head_X.OutData.Wordadr_Low	Start address low Byte							
6..65	#Head_X.OutData.DW 1 ... DW 15	unused							

Table 3.1: Structure telegram single read data command

The first two Bytes of the telegram comprised the length of the telegram. This is realized by the parameters **#Head_X.OutData.TelegrammLengthHigh** and **#Head_X.- OutData.TelegrammLengthLow**. Inside the parameter **#Head_X.OutData.CommandCode** is the command identification number of the command which is to be executed by the IDENTControl. For execution of a Single Read Data command the command identification number is binary coded (10000)b. A detailed description of the different command identifications is inside the manual “IDENTControl IC.KP-B6” or in the command list in the addendum of this manual. With the parameter **#Head_X.OutData.Channel** two different command attributes were transmitted. By the high nipple the number of read/write data blocks is defined. The number of the data blocks is inside the variable **#Head_X.WordNum** and is transferred inside the FB into this parameter. The data blocks have got a double word format and a size of 4 Bytes. The maximal number of transmitted data blocks depends on the port definition inside the Hardware configuration. By the definition of port 10000 you can maximal transmit 7 data blocks. In this example the definition port 10001 is used. So you can maximal transmit 15 data blocks. Inside the low nipple is the information on which head the command has to be executed. The following chart list the different channel codes to respond the different heads.

Read / Write Head	Coding	
	dez	dual
1	2	(0010)
2	4	(0100)

Read / Write Head	Coding	
	dez	dual
3	6	(0110)
4	8	(1000)

Table 3.2: Channel coding of the Read / Write heads

By the coding of the channel you consider that the LSB is the Toggle bit. But if the communication between IDENTControl and PLC is realized by TCP/IP the Toggle bit is not used. The parameters **#Head_X.OutData.Wordadr_High** and **#Head_X.OutData.Wordadr_ Low** define a start address. Starting from that the data blocks can read or write. So you can respond different memory areas on the data carrier. The parameters **#Head_X.OutData.DW1...DW15** contained the data. But by the execution of the Single Write command this parameters are not used. The IDENTControl responds after the acceptation of the command telegram with a response telegram. The structure of the response telegram is shown in following table.

Byte	Contents / Variable	Bit Allocation							
0	#Head_X.InData.TelegrammLengthHigh	0	0	0	0	0	0	0	0
1	#Head_X.InData.TelegrammLengthLow	0	1	0	0	0	0	1	0
2	#Head_X.InData.CommandCode	0	0	0	1	0	0	0	0
3	#Head_X.InData.Channel	Number data words				Channel number			
4	#Head_X.InData.Status	Status							
5	#Head_X.InData.ExecutionCounter	Event counter							
6..(33) 65	#Head_X.InData.DW 1 ... DW (7) 15	00 ... FF							

Table 3.3: Structure response telegram Single-Read-Data command

The response telegram send back the first four Bytes of the command telegram. Also the response telegram contained a status message. The status message gives information about the execution of the command. With the help of the Event counter the number of command events can determine. The Event counter will be increment on every command event. A command event is for example a status change from 00h to FFh. The other parameter of the response telegram contains the import data. The number of import data depends on the parameterization inside the Hardware configuration.

3.4 Execution of the Initialization

Initialization

1. At the beginning of the treatment of the function block initialize the read/write heads.
2. Perform the initialization for each head in turn.

↳ During the initialization routine a change tag command will be sent to the IDENTControl. This command informs the head which tag he is talking to.

Based on the state response the functional block recognizes whether a read/write head is connected to the appropriate channel.

If the initialization for one head is finished, the bit **#Head_X.ExistTC** or **#Head_X.NotExist** will be set.

The bit **#Head_X.ExistTC** will be set, if a read/write head is connected to the appropriate channel. Otherwise the bit **#Head_X.NotExist** is set.

During the execution of the initialization consider that the correct Tag ID has to be assigned.

↳ The Tag ID is assigned to the variable **#Head_X.TagType**

The Tag ID is assigned to the variable #Head_X.TagType and specified by the user. The indication of the Tag ID taken place inside the program in hexadecimal form. The Tag ID of all tag useably you can see in the manual "IDENTControl IC-KP-B12-V45“near to the description of the change tag command. It is to be considered that the tag ID inside the manual is in ASCII form. The transformation into hexadecimal form can be making with the help of the ASCII chart inside the manual. The Tag ID can also get out of the addendum of this manual. After successful finished execution of the initialisation of all heads a command can execute by the IDENTControl.

3.5 Execution Single Command

A single command is executed by the IDENTControl for one time. For execution of the command it is necessary to the command parameter of the head specific data field inside the PLC to the IDENTControl. Before the out data field is sent to the IDENTControl it has to assign with the associated command parameter and data. To the command structure here is no more information. For more information see the section “Command Structure”.

Before execution of a single command the user has to configure different IN variables. The parameter #HeadXDataFixcode specifies whether an access to the memory area (#HeadXDataFixcode = 0) or an access to the Fixcode (#HeadXDataFixcode = 1) is executed. A single command is executed if the variable #HeadXSingleEnhanced is not set. The command execution starts by setting the variables #HeadXRead or #HeadXWrite. By setting on of these two parameters the command assignment inside the function block is started. Afterwards the out data field with the command parameters transferred to the IDENTControl and executed. The IDENTControl sent after accepting of the command telegram a status massage back to the PLC. The status massage is used inside the function block to realize an error analysis. The exact meaning of the specific status values is inside the manual “IDENTControl IC-KP-B12-V45”.

By the analysis of the status values specific status bits can be generated. On the basis of the status bits you can get information about the execution status of the executed command.

The status bit #Head_X.Busy point out that the command is executed by the head at the moment. If the Bit is set, no other command can executed on this head. This bit is reset inside the function block if a command execution is finished by the IDENTControl.

The end of the command execution is identified by the status bit #Head_X.Done. After finalization of the command execution of the IDENTControl the bit is automatically set inside the function block. A new command execution is only possible if the bit #Head_X.Busy is not set and the bit #Head_X.Done is set. If there is no data carrier is inside the field of the head by execution of a command, the status bit #Head_X.NoDataCarrier would set. Than the command execution is automatically repeated by the IDENTControl. The maximum number of repetitions is defined by the user with the help of the variable #RetrySingleCommand. Maximum number applies for all heads.

If a Timeout existed by the command execution the bit #Head_X.TimeoutOccured is set. A Timeout existed, if the command execution is not finished in the time period defined by the IN-Variable #Timeout. This considered as an Error and the bit #Head_X.Error is set.

The bit #Head_X.Error signalizes the occurrence of an error in the command execution of the IDENTControl. Once the bit is set, the associated head is locked for other command executions. The blockade can disabled by the IN-Variable #QuitError- HeadX.

The import user data located inside the function block in the data field #Head_X.InData.DW1…DW15. After finalization of the command execution the import data can use.

The user data, which to be written on the data carrier are inside the data field #Head_X.OutData.DW1… DW15. The user data can be defined the user and have to be allocated before the start of the command execution. The settings, which are necessary for the execution of a single command, can be seen in the command list in the addendum.

3.6 Execution Enhanced Command

An Enhanced command is executed by the IDENTControl as long as the command is abort by a quit command. By the execution of the enhanced command the Reading / Writing as the data transmitted to the data carrier or the data imported. In contrast to the single command the enhanced command remains active after the Reading / Writing. After the data carrier leaves the field of the head a "new“ one can be write or read by the head. But only one data carrier can stay at the same time inside the field of the head.

For the execution of the Enhanced command different In-Parameter has to be defined. Compared to the single command the parameter #HeadXDataFixcode define whether an access to the memory area (#HeadXDataFixcode = 0) or the Fixcode (#HeadXDataFixcode = 1). The variable #HeadXSingleEnhanced = 1 specifies the execution of an enhanced command. The variable has to be set before beginning of the command execution. By setting the variables #HeadXRead or #HeadXWrite the command execution is started.

2011-01

The command already executed by the IDENTControl. This is signalized by the bit #Head_X.Busy. The bit is set as long as the command is executed or aborts by an error or quit command.

The bit #Head_X.Done signalizes by the execution of an enhanced command that the Reading / Writing of a data carrier is finished. In contrast to a single command this bit not signalizes the end of a command execution. If a data carrier leaves the field of the head, the bit #Head_X.Done automatically reset but the command is already active.

The execution of a new command is only then possible if the bit #Head_X.Busy is reset and the bit #Head_X.Done is set. This condition is reached after the execution of a quit command.

As the execution of a single command the user data which written to the data carrier are inside the data field #Head_X.OutData.DW1… DW15. The Output data have to be defined before the execution of an enhanced command is started. A change of the output data has no influences of the execution of the enhanced command.

The input data stored in the data field #Head_X.InData.DW1…DW15 of the data block. The settings, which are necessary for the execution of an enhanced command, can be seen in the command list in the addendum.

3.7 Execution Special Command

By using the Special command the user can parameterize a command single handed. Primary the command is used to execute commands which are not executable with the command list of the function block. But you can also execute standard read / write commands with the help of a special command.

Before starting the execution of a special command the command parameter have to transfer into a particularly data field (#Head_X.SpecialCommand) inside the instanz data block. The command code of command which can be executed have to assigned to the variable #Head_X.SpecialCommand.Code. The assignment of the channel identification is not necessary for the execution of a special command. An appropriate assignment is making automatically inside the function block. During the execution of a read / write command the variable #Head_X.SpecialCommand.Channel contains the number of transferred data blocks. The number is inside the high nipple of this variable.

For the parameterization of a special commend there are further variables to use. With the variables #Head_X.SpecialCommand.Parameter1… 6 you can assign further command parameter.

More information you get out of the manual “IDENTControl IC-KP-B12-V45”. The execution of a special command is started by setting the IN-Variable #HeadXSpecialCommand. Afterwards the data field #Head_X.SpecialCommand assigns to the output data field #Head_X.OutData and transmitted to the IDENTControl. Following the transmitted command is executed. If an enhanced command is executed with the help of a special command the command have to abort by a quit command at the appropriate head.

With the help of the special command you can transmit commands to the IDENTControl which are not executed by the heads. These command called IDENTControl commands and they used to parameterize the IDENTControl. An example for this command is the Set Multiplexed Mode command. As a special command the parameter of the command is inside the data field #Head_1.SpecialCommand. The commands are not executed by the heads, so that is no channel identification is necessary. The execution of the IDENTControl command starts by setting the INVariables #Head1SpecialCommand and #IC_Command_on_Head1. Following the data field #Head_1.SpecialCommand transferred to the output data field #Head_1.OutData and sends to the IDENTControl and execute.

3.8 Execution Error Analysis

It can be possible that an error occur by the execution of a command. The function block offers the possibility to analyse the occurred error. If an error occurred the bit #Head_X.Error is set. Afterwards the associated head is disabled for any command executions. To analyse the command error different parameter exist. In the following table different parameters listed to analyse the error.

In der nachfolgenden Tabelle sind die verschiedenen Parameter zur Fehlerauswertung aufgeführt.

Parameter	Meaning
#Head_X.Error	Error command execution
#Head_X.InvalidResponse	Invalid response of IDENTControl
#Head_X.TimeoutOccured	Timeout exhausted
#Head_X.Error_FC_Recv	Error execution FC 60
#Head_X.Error_FC_Send	Error execution FC 50
#Head_X.RetVal_FC_Recv	Error code FC 60
#Head_X.RetVal_FC_Send	Error code FC 50
#Memory_Error_FC_Recv	Error execution FC 60
#Memory_RetVal_FC_Recv	Error code FC 60
#DynErrorTCPConnection	Failure in the communication
#Error_FC_Recv	Error execution FC 60
#Ret_Val_SFC20	Error code SFC 20

Table 3.4: Error Analysis

The error locking of the head can enabled with the execution of a quit command. The quit command starts by setting the IN-Variable #QuitErrorHeadX. By this command all bits are reset which disable the command execution. Afterwards the commands can execute if no error caused a new error locking.

4 Used Modules and Functionality

The connection of the IDENTControl to a PLC is realised by a function block. The function block is user programmable and can optimize of associated application. The data, which are necessary by the execution of the function block, are memorized into an instanz data block. Also there are different other system functions with different functionality necessary to realize the communication between the IDENTControl and the PLC. Following table shows used blocks and their functionality.

Block	Type	Function
OB1	Organization Block	Cicely goes through by the operating system.
FB 10 "IDENTControl"	Function Block	Operate the communication between the IDENTControl and the PLC.
DB 10 "InstDB"	Data Block	Memorize local data.
FC 50 "AG_LSEND"	Function	Sending of data over parameterized Ethernet connection.
FC 60 "AG_LRECV"	Function	Receiving of data over parameterized Ethernet connection.
SFC 20 "BLKMOV"	System Function	Coping of memory area.
SFB 5 "TOF"	System Function Block	Realize release delay.
SFC 58 "WR_REC"	System Function	Writing data block to device.
SFC 59 "RD_REC"	System Function	Reading data block from device.

Table 4.1: Used Blocks and Functionality

The correlations between the different blocks are shown in the following picture.

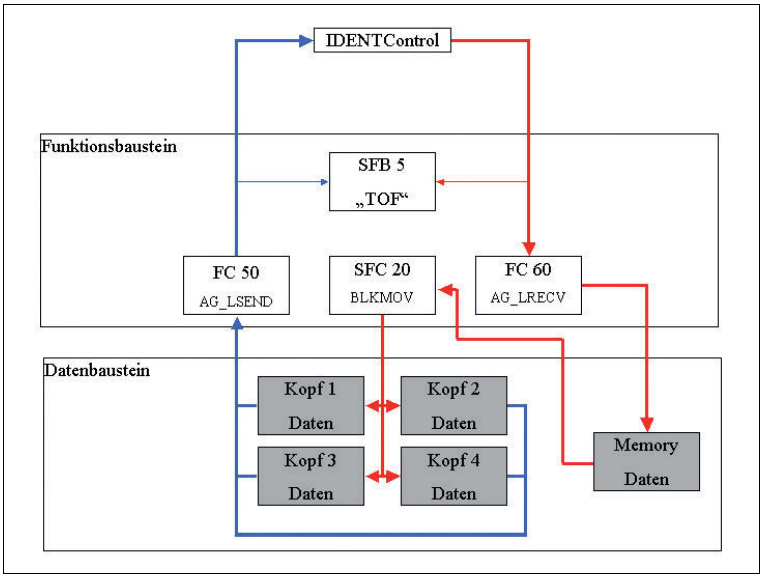


Figure 4.1: Correlation of the Blocks

In this graphic you see the different blocks and the data flow exchange between the IDENTControl and the PLC. The data fields which are appropriate to a head are divided into an input and output data field. The sending of data to the IDENTControl is realised by the function FC 50. This function transmits data over a parameterized connection. Inside the function the SFC 58 is called. With the help of the SFC 58 data transferred to the device. By sending data to IDENTControl a release delay (SFB 5) is activated to check the response time of the IDENTControl. The receiving of the data is realised by the system function FC 60. Inside the function the SFC 59 is called. The SFC 59 import data from a parameterized device. The release delay is disabled as data import from the IDENTControl. The import data memorised in the memory data field inside the instanz data block. Afterwards the data analysed and copied by the system function SFC 20 in the head specific input data fields.

5 Organization Block OB 1

The organization block OB 1 is cyclical passing through by the operating system. The OB 1 is the interface between the operating system of the CPU and the user program.

Firstly the data carrier identification is transferred into the instanz data block. The assignment of the data carrier is neccessary for the initialisation routine and has to assign before starting the initialisation. By the assignment of the carrier identification must considered that the identification is in hexadecimal form. The carrier identification is transferred to the variable #Head_X.TagType. The assignment of the carrier identification is necessary for all connected heads.

Afterwards the number of transferred data block must be assign. The number of transferred data blocks is depends on the hardware configuration. The user can define the number within the given boarders. The assignment is transmitting into the variable #Head_X.WordNum. You must consider that the number is fort he time period of command execution is constantly.

In the next step the function block "IDENTControl“ and the associated data block "InstDB“ is called. Both blocks are called with the function call "Call“. Example: Call "IDENTControl“, "InstDB“

The function block is able for multiinstanz. Multiinstanz is that you can assign different data blocks for the function block. So you can connect several IDENTControl to the PLC with the help of on function block. For this you must call the function block for several times.

Example: Call "IDENTControl“, "InstDB1“ Call "IDENTControl“, "InstDB2“

6 Function Block "IDENTControl“

The function block “IDENTControl” has got the task to connect the identification system IDENTControl to the PLC. The function block is user programmable. Thereby the functionality of the function block can adjust to the application of the customer. The function block is divided in several networks. Following chart shows the several networks and the functionality.

Network	Function
1	Start-Up-Sequence
2	Initialization Head 1
3	Initialization Head 2
4	Initialization Head 3
5	Initialization Head 4
6	Timeout Control
7	Transformation IN- to STAT-Variables
8	Import the response
9	Command Allocation Head 1
10	Command Allocation Head 2
11	Command Allocation Head 3
12	Command Allocation Head 4
13	Command Execution Head 1
14	Command Execution Head 2
15	Command Execution Head 3
16	Command Execution Head 4
17	Restart-Routine
18	Acceptation
19	Error-Routine
20	Analysis FC 50
21	Analysis FC 60
22	Analysis Input Data Fields

Table 6.1: Networks of the Function Block “IDENTControl”

6.1 Procedure Start-UP-Sequence

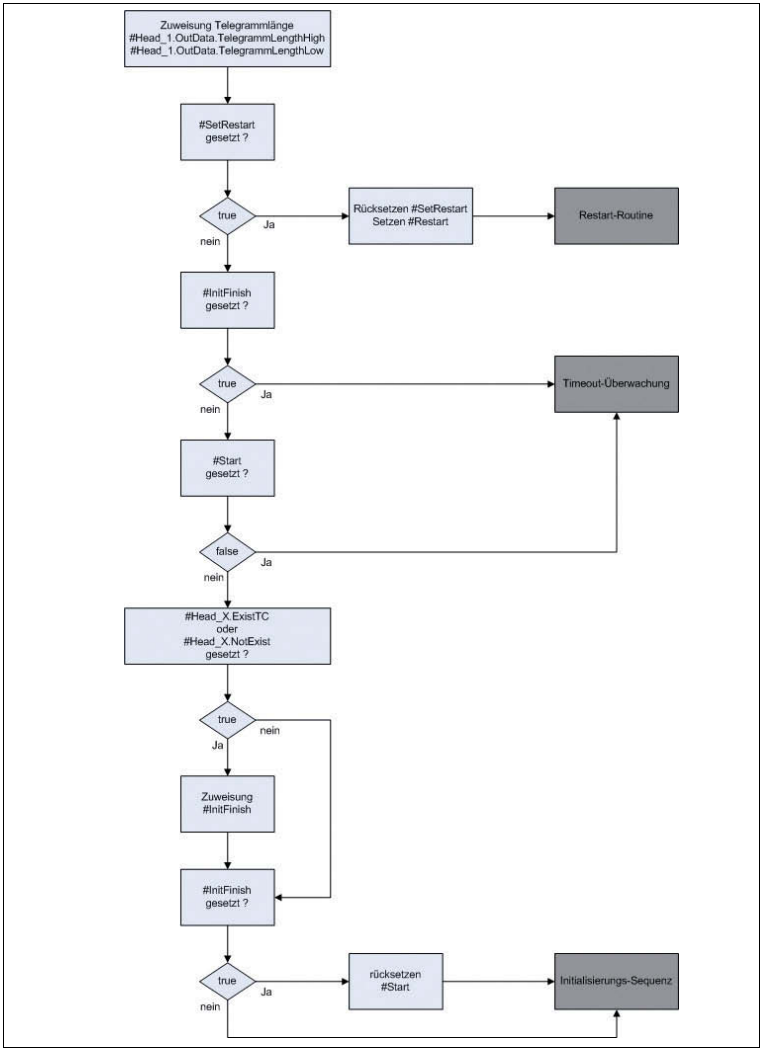


Figure 6.1: Program flow chart Start-Up-Sequence

The Start-Up-sequence of network one has the task to recognizes if the heads of the IDENTControl are already initialised. With the help of the initialization the function block checked which heads are connected to the IDENTControl.

At the beginning of the network the value of the telegram length (IN-variable #TCP_Telegrammlength) transferred to the variable #Head_X.OutData.-TelegrammLengthLow inside the out data field.

Secondly the bit #SetRestart is checked. The condition of a restart is fulfilled, if the bit is set. Then the program jump to the point res in the network 19.

If there is no restart of the function block the bit #InitFinish is checked. The bit is set, if the initialization is finished for all heads. In this way the program jumps to point end into network 6 to the Timeout Control. Also there is a jump to point end if the bit #Start is not set. The bit #Start signalize that a restart was done before.

In the next part the successful initialization finish of each channel is checked. For this the bits #Head_X.Exist and #Head_X.NotExist are checked. If one of them is set the initialization is finished. The head is connected to the IDENTControl if #Head_X.Exist is set. There is no head connected to the IDENTControl if #Head_X.NotExist is set. The check is doing for all heads. If the check for all heads is finish the bit #InitFinish is set. This bit signalizes the successful ending of the initialization of all heads. Afterwards the bit #Start reset.

Now the Start-Up-Sequence is finished and the initializations of each head followed if they are not done in the cycles before.

6.2 Procedure Initialization

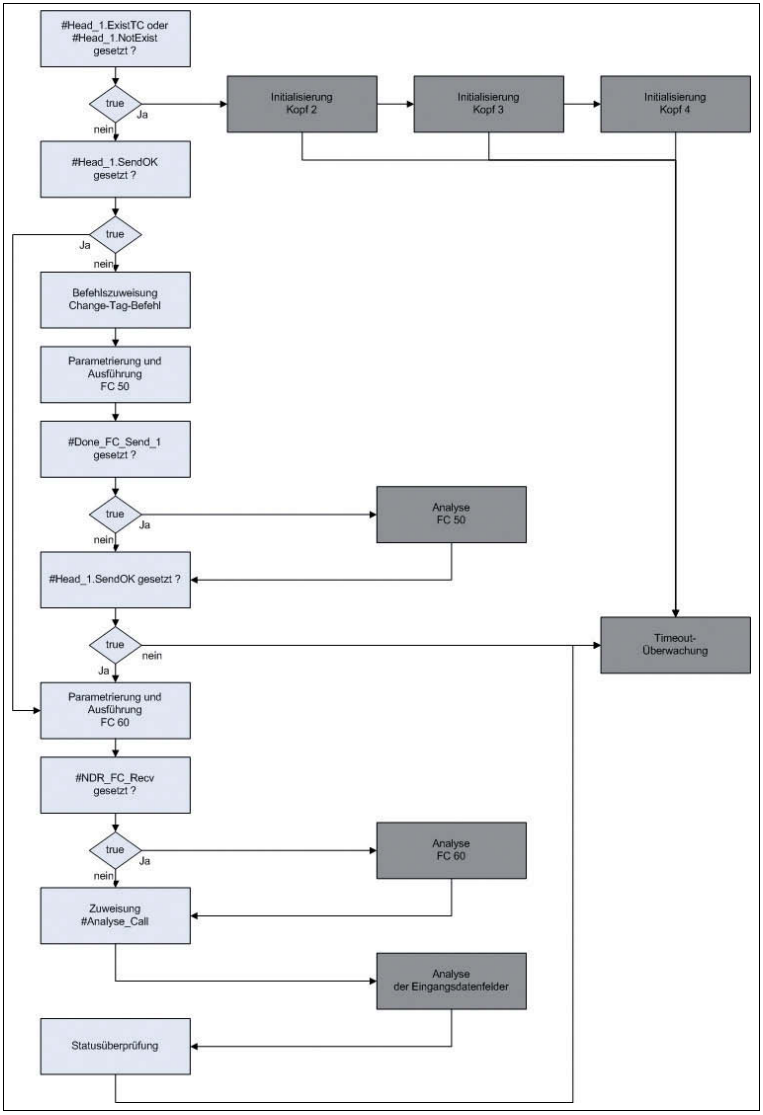


Figure 6.2: Program flow chart Initialization Head 1

The program part initialization is necessary to recognize if a head is connected to a channel of the IDENTControl. Inside the Initialization the change tag command is send to the IDENTControl on the appropriate channel. With this command the carrier identification is transmitted to the IDENTControl. In the following the initialization for channel 1 is described. The initialization for the other heads is analogue.

In the first part of this network the initialization of channel 1 is checked. For it the bits #Head_1.Exist and #Head_1.NotExist are checked. If one of them is set, the initialisation of channel 1 is finished and the program jump to point ini2. At this point there is the initialisation check of head 2.

If the initialisation is not finished successfully the bit #Head_1.SendOK is checked. If this bit is set the change tag command is send to the IDENTControl in the cycles before. Now the command no more does not have to be sending to the IDENTControl so the program jumps to point rec1. With this jump the parameter loading into the out data field and the sending to the IDENTControl is avoided.

Afterwards of the check of the status bits the command parameter of the change tag command transferred into the out data field of head 1.

After the parameter transfer the status bits are set or reset. Setting #Head_1.Busy signalize that a command is executed at channel 1 at the moment. By setting the bit #Head_1.TimeoutActive the timeout control is prepared.

After transferring the command parameter and setting the status bits the command is transmitted to the IDENTControl. The sending of the data via TCP/IP is realized by the function FC 50. Before the function is called, the function has to parameterize. The parameterization is doing by an ANY parameter. Afterwards the function FC 50 is called. After a successful execution the bit #Done_FC_Send_1 and the program jumps to the point aus1 to the network 20. At this point an error analysis is executed. After a successfully execution of the error analysis and no error is recognize the program jump back to the point F501 in network 2.

If the function is already in progress the bit #Done_FC_Send_1 is not set and the program jumps not to the error analysis. Instead the program jumps directly to point F501. At point F501 the program checked the bit #Head_1.SendOK whether the bit is set. The bit is set if the execution of the function FC 50 is successfully finished and without errors. In this case the further processing of the program is at point rec1. But if the bit #Head_1.SendOK is not set, the program jumps to the timeout control in network 6.

After the successful sending of the command parameter to the IDENTControl the import of the response followed. The import of the received data is realised by the function FC 60.

Firstly the value 0 is transferred to the variable #FC_Recv_Call. With the help of this variable the program realized the return to point F601 into this network. The necessary parameterization is transferred into the ANY parameter. Afterwards the function FC 60 is called. After a successfully execution of the function the bit #NDR_FC_Recv is set and the program jumps to point aus5 into network 21. At this place an error analysis for the function executes. After the correct execution of the analysis and no error is detected the return to point F601 into network 2 following.

2011-01

If the function is already in progress, the bit #NDR_FC_Recv is not set and the program does not jump to the error analysis. Instead the program jumps directly to point F601.

At point F601 the variable #Head_1.TimeoutActiv is reset. Afterwards the value 0 transferred to the variable #Analyse_Call. With the help of this variable the return to point ana1 inside this network is realized. Than the program jumps to point lyse into network 22. There the received data from the IDENTControl were analysed. After the analysis of the received data the program returns to point ana1 into network 2.

At this point the analysis of the status value executed. The response of the IDENTControl contains a status value. With the analysis of the status value you get information about the command execution of the IDENTControl. The status value has got a length of one byte and contains in the variable #Head_1.InData.Status. If the value of the status (06)h, then no head is connected to the IDENTControl on this channel and the bit #Head_1.NotExist is set. Because that no head is connected the status bits #Head_1.ExistTC and #Head_1.Error are reset. Afterwards the bit #Head_1.Done is checked, whether the bit is set. This bit signalizes, that new data can be analyse on head 1. Also the bit #Head_1.NotExist is checked, whether it is not set. If the two conditions are fulfilled a read write head is connected to channel 1 of the IDENTControl and the bit #Head_1.ExistTC will be set. Finally the program jump to point end into network 6 to the timeout control. After the processing of this part the initialization of head 1 is finished. In the next cycle the initialisation of head 2 follows unless it is not processing.

6.3 Procedure Timeout Control

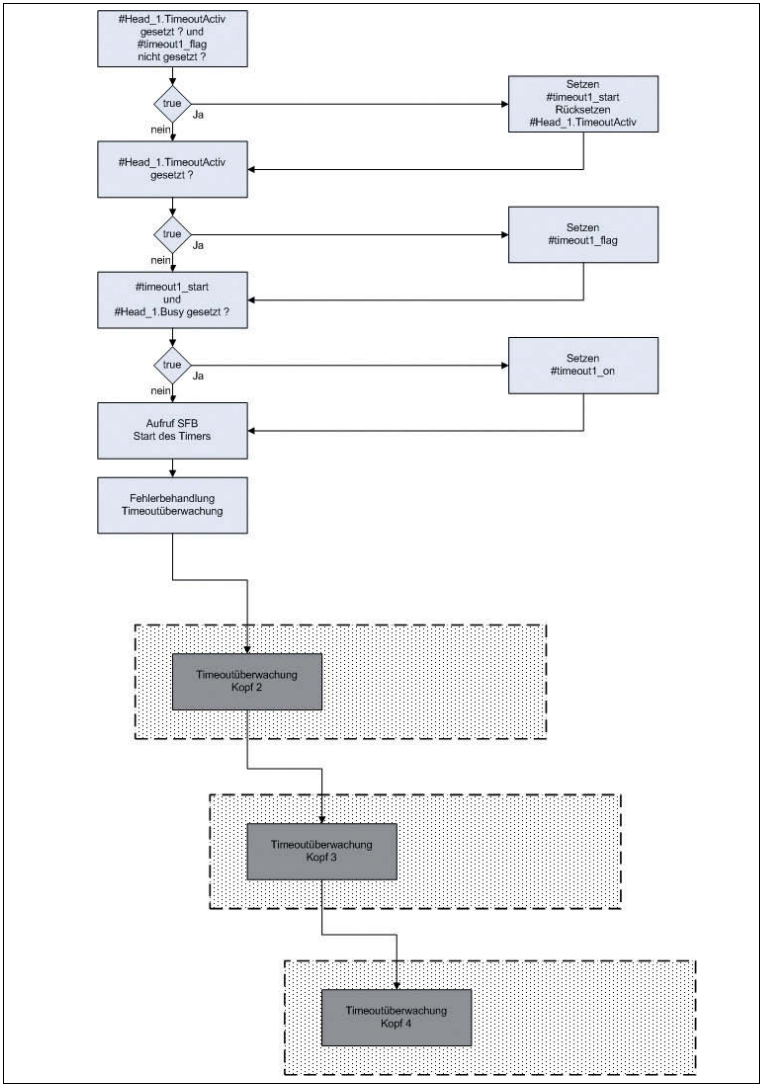


Figure 6.3: Program flow chart Timeout Control

The Timeout Control has the task to check the maximal response time of the IDENTControl. Another task is to display an error notice if the timeout exhausted. In the following describes the Timeout Control for head 1. The other heads are analogue.

2011-01

Firstly the bit #Head_1.TimeoutActiv is checked whether it is set and the bit #timeout1_ flag whether it is not set. The assignment result is transferred to the variable #timeout1_start. If the condition is fulfilled, so the variable #Head_1.TimeoutActiv is reset. Otherwise only the setting of #Head_1.TimeoutActiv is checked. The assignment result transferred to the variable #timeout1_flag. Afterwards the bits #timeout1_ start and #Head_1.Busy are checked whether both bits are set. The result transferred to the variable #timeout1_on.

In the next step the system function block SFB 5 is called. This function creates a falling delay on output Q. The variable effects a rising edge on input IN thereby is a rising edge on output Q (#timeout1_running). If a falling edge on input IN so exist a falling edge on output Q after elapse the time PT #Timeout.

Next the bit #Head_1.Busy whether it is set and the bits #Head_1.ReceivedOK, #timeout1_running and #timeout1_start whether they are not set. If the check is successful, the timeout is exhausted and an error notice followed. Then the bits #Head_1.TimeoutOccured, #Head_1.Error and #Head_1.Done are set. The bits #Head_1.SglCommandActiv, #TransfToHead1, #Head_1.Busy and #Head_1.TimeoutActiv are reset.

This network realizes a falling delay. The delay condition is defined whether the bits #Head_1.TimeoutActiv and #timeout1_flag are not set. Another condition is that the bit #Head_1.Busy has to be set. Both Head_1 bits are set when the function FC 50 is called. The variable #timeout1_flag signalize that the timeout is running at the moment and could not be extended.

6.4 Procedure Variable Transformation IN- to STAT-Variables

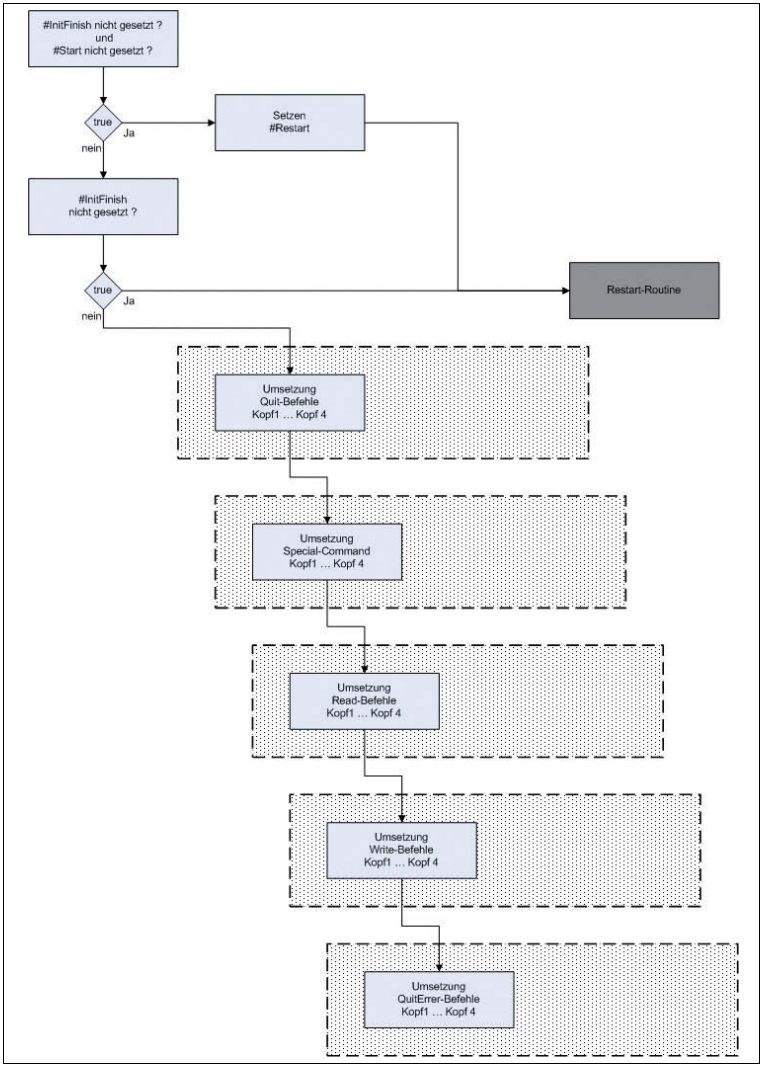


Figure 6.4: Program flow chart Transformation of variables

This part has the task to change the IN variables into static variables. This change is necessary because the state of the IN variables can oscillate. A consequence of it can be a start of a new command while the old command still executed. Another disadvantage of the IN variables is, that they can not manipulated by the function block.

Thus the command which started by the IN variable executed for the holy time the IN variable is set. An answer of this problem is a flank control.

If a positive edge of the IN variable is detected the change into the corresponding STAT variable following. The STAT variable is a local variable inside the function block. The STAT variables have the advantage that they can manipulate inside the function block. After the execution of a command the STAT variable is reset and a new command can be started.

At first the variables #InitFinish and #Start are checked whether they are not set. This condition is fulfilled when the initialisation of the heads is not finished and the Restart routine was not implemented yet. Consequently the bit #Restart is set and the program jump to the point end1 to the end of network 16. There the Restart routine is started.

Otherwise the bit #InitFinish is checked. The user can initiate the restart of the function block by reset this bit. If the initialization is not finished a jump to the restart routine followed and the initialisation is added in the next program cycle.

If the initialisation already successful finished the change of the variables followed. Subsequently the quit commands which are defined external changed. Firstly the state of the cycle before is memorized into the value #SaveQuitHead1. This variable is compared to the variable #Head1Quit. If the value of #Head1Quit = 1 and #SaveQuitHead1 so a positive edge is detected and the bit #QuitHead1 is set. The change of the other heads is analogue.

6.5 Procedure Read Data

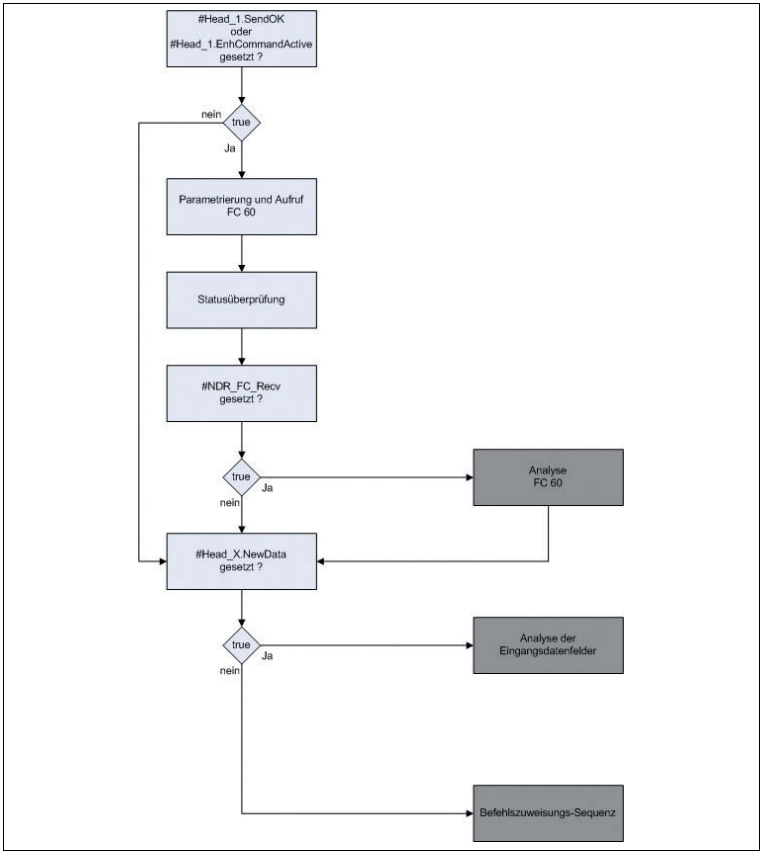


Figure 6.5: Program flow chart Data Import

This network has the task to import data from the IDENTControl by using the function FC 60. After a check whether new data could be import, the function will be execute. Afterwards the data will be analyses.

At the beginning of the network the finish of a command send to the IDENTControl is checked. When a command was sending to the IDENTControl the PLC can read the command response. Also the execution of an enhanced command is checked. If a enhanced command is active on one of the heads, the PLC can always import new data from the IDENTcontrol. The import of the command response will not be executed if an enhanced command is not active or the command sending to the IDENTControl was not successfully. Afterwards the program jumps to the point F605 at the end of this network.

After the analysis whether new data could be import the execution of the function FC 60 follows.

Firstly the value 4 is transferred to the variable #FC_Recv_Call. With the help of this variable the return out of the analysis in this network is realised. Afterwards the required parameters transferred to an ANY parameter type. The import data will be memorised in the memory data field. The beginning of the memory data field is at address 642.0. The start address of the memory data field has to be considering in the configuration of the ANY parameter type. After the configuration of the ANY parameter type the execution of the FC 60 follows.

The variable #Memory_RetVal_FC_Recv contains a status value. At first the variable is checked whether the TCP connection is interrupted. If the connection is interrupted the value of #Memory_RetVal_FC_Recv is (8304)h. In this case the bit #DynErrorTCPConnection will be set.

Afterwards the variable is checked whether the value is 0. The variable has got the value 0 if the data import by the function. In this case the execution was free of failure and the bit #DynErrorTCPConnection is reset.

By the execution of the function there is the possibility that no data exist at the parameterised source address. This is signalised with the failure code (8180)h. In this case new data could not import but the communication between PLC and IDENTControl already existed and the bit #DynErrorTCPConnection is reset.

The end of the execution of the function FC 60 is signalising by the bit #NDR_FC_Recv. The bit is set, if the execution of the function was successfully. Afterwards the program jumps to point aus5 inside the network 21. At this point the import data will be analyses. After the analysis the program returns to the point F605.

If the function FC 60 is already in execution the bit #NDR_FC_Recv is not set. In this case no data are imported and the jump to the analysis is not necessary. The program handling continues at point F605.

At point F605 the program checked whether new data are existed to analyse. As soon as new data exist at head X, the bit #Head_X.NewData will be set. The bit is set in the analysis of the function FC 60 if the copying of the memory data field to the specific input data fields of the heads was successfully. If new data exist on one of the heads, the program jump to point lyse into the network 22. At this point the input data of the heads will be analysed. An analysis is necessary if new data are inside the head specific input data fields. The return out of the analysis is realised by the variable #Analyse_Call. The value 4 is transferred to this variable and the program jump to pint ana5 into network 9.

If no new data inside the input data fields of the heads the bit #Head_X.NewData is not set. The analysis of the input data fields is not necessary. The program jumps to point ana5 inside network 9. At this point the command allocation for head 1 followed.

6.6 Procedure Command Allocation of Head 1

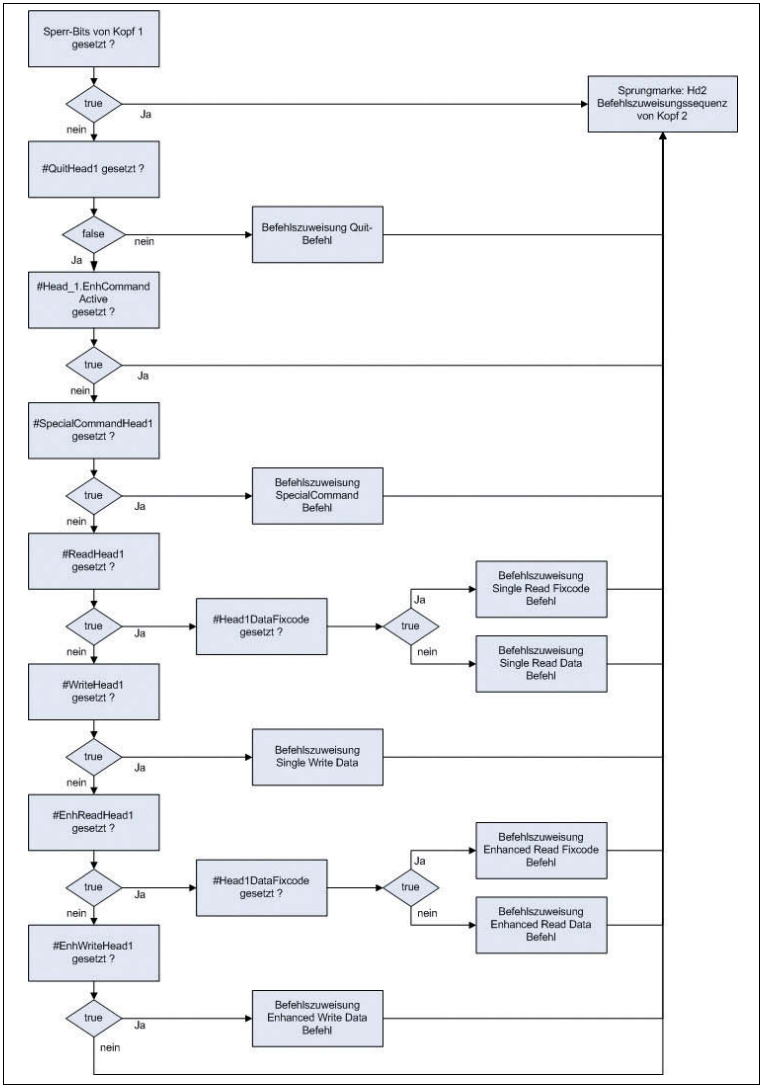


Figure 6.6: Program flow chart of Command Allocation of Head 1

In this part the command allocation for head 1 is exemplarily showed. A command allocation for the other heads is analogue.

At first the head is checked whether head 1 is locked. There are different options which caused a locking of a head. If an error occurs in the last command execution of head 1 the head is locked for further commands until the head is reset by setting the bit #Head_1.QuittError. The command allocation is also blocked if there is no head connected to the corresponding channel. Another possibility of head locking is that data already present in the out data field of the head. This data are the command parameter of the command before but they are not sending to the IDENTControl. Thus the executions of commends is successful the command allocation is blocked until the command execution is finished. The check whether the head is locked is realized by an OR combination. If one of the conditions is complied the command allocation is locked and the program jumps to point Hd2. At this point starts the check for the head 2.

Following the check of the head the command parameter allocate to the out data field. But in the first step the program identify which command to be executed on head 1.

Firstly is checked whether a quit command is executed on head 1. In addition the setting of bit #QuitHead1 is checked. If no quit command is executed on head 1 the bit is not set. Thus a jump to point SCH1 followed. At this point there is a check whether an Enhanced command is active.

If a quit command is to execute the bit #QuitHead1 is set and now the command parameter must transfer into the out data field of head 1. After transferring the command parameter the locking bits of head 1 are set. The bit #TransfToHeadX signalize that the command parameter were loaded into the out data field. But the sending of the command to the IDENTControl did not take place. Thus the parameter of the quit command after the first command execution not transferred in the out data field again, the bit #QuitHead1 is reset. Thus other commands can start on head 1 after the execution of the quit command is finished. With the help of the Reset of the bits #Head_1.EnhCommandActive and #Head_1.Busy the program signalize that the command executed the quit command only for one time and not permanently. After the parameter allocation of the quit command into the out data field a jump to the command allocation of head 2 (Hd2) take place.

If no quit command is to execute on head 1 a jump to point SCH1 followed. At this place firstly checked whether an Enhanced command is executed on head 1 or other commands are in treatment. If one of these conditions is fulfilled the head is locked for other command executions. In the program it is not possible to start an other command on head 1 while an enhanced command is executed. Therefore all start bits for other commands are reset. An exception forms here the quit command. This command can allocate and execute while an enhanced command is active. Thus the enhanced command will be abort. If an Enhanced command is active a jump to command allocation of head 2 (Hd2) followed after reset of the start bits. Afterwards the status of the import data of head 1 is checked. If the value of the status is (FF)h and a single command is active at the same time a command allocation is not necessary and the program jump to point Hd2 (command allocation of head 2). Afterwards checked which command is to execute on head 1.

If the execution of a special command on head 1 is intend this is signalised by the bit #SpecialCommandHead1. If the bit is set, the program jumps to point SH1. At this point the command parameters, which defined by the user, transfer into the out data field of head 1.

If a read command will be started on head 1 the bit #ReadHead1 is set and the program jumps to point SRH1. At this point the command parameter of the single read command transferred into the out data field of head 1.

During an intended execution of a write command the bit #WriteHead1 is set. In this case, the program jumps to point SWH1. At this point according to the other commands the parameters of the single write command transfer into the out data field of head 1.

During the execution of an enhanced command it applies to note that it is to differentiate between enhanced read and enhanced write command. If an enhanced read command is to execute the bit #EnhReadHead1 is checked. If the bit is set, the program jumps to point ERH1. Else the bit #EnhWriteHead1 is checked. The bit is set, if an enhanced write command is to execute and the program jump to point EWH1. With the help of this graduated inquiry it is possible to check all commands and jump to the associated command parameter allocation. But it is also possible to execute no command on channel 1 of the IDENTControl. In this case all checked bits are not set and the program jump to the command allocation of head 2 (Hd2).

6.7 Procedure Command Execution of Head 1

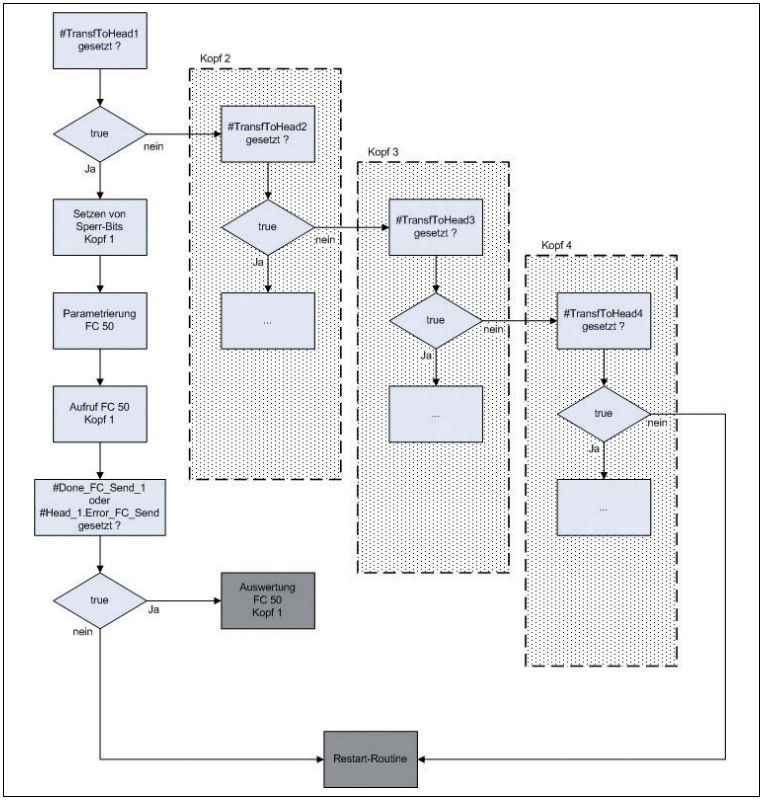


Figure 6.7: Program flow chart Command Execution Head 1

In this part the command execution of head 1 is describing exemplarily. The procedure of the command execution of the heads 2, 3 and 4 is analogue. Therefore in this part is no description of the command execution of the other heads.

By a command execution understands the sending of the command parameters which are allocated in the out data fields to the IDENTControl. The allocation of the command parameters was described in the part before. The sending of the command parameter is realised by the function FC 50. After the successful transfer of the out data field the command is acknowledged by a status response. Then the command is executed by the IDENTControl.

At the beginning of the command execution the out data fields of the heads are checked whether new command parameters are inside. If new command parameters inside the out data fields which are not send to the IDENTControl in the cycle before the bit #TransfToHeadX is set. That is why the program checks all these bits at the beginning of the network. If the bit #TransfToHead1 is set, the program jumps to point exe1. At this point the command parameter transferred to the IDENTControl and will be executed. Otherwise the bit #TransfToHead2 is checked. The same procedure is doing with the other bits #TransfToHeadX. If none of these bits are set, the program jumps to point res. In this case no new command starts on the heads. At point res a restart routine will be started.

At point exe1 the destined command parameters of head 1 send to the IDENTControl. Firstly different locking bits are set. The bit #Head_1.Busy signalises that a command is executed at head 1 at the moment. With the help of bit #Head_1.TimeoutActiv the timeout control is initialise. With the help of the variable #Head1_exe you can differentiate the point of calling the FC 50. There are two different points in the program where the function FC 50 is called for head 1. One point is the initialisation routine of head 1 and the other point is the command execution of head 1. If the FC 50 is called inside the initialisation routine the bit #Head1_exe is reset. But if the FC 50 is called inside the command execution the bit #Head1_exe is set. With the help of this bit the program realise the return to the point where the function was called. Afterwards different status bits are reset at point exe1. Thereby the sending of the command parameter to head 1 is enabled.

Afterwards the parameterization of the FC 50 followed. The parameterization is realised by an ANY parameter type. With the help of the parameter the source data field is defined. The source data field for the call of the FC 50 is the out data field of head 1 #Head_1.OutData. The start address inside the data block of the data field is 92.0. The allocation of the correct data field is necessary otherwise the out data field with the command parameter are not sending to the IDENTControl.

After the parameterization the call of the function FC 50 followed. Afterwards the sending of the data is checked whether the execution is finished. Thus the bits #Done_FC_Send_1 and #Head_1.Error_FC_Send are checked whether they are set. If the bit #Done_FC_Send_1 is set the sending of the command parameter to the IDENTControl was successful. But if an error occurred by the execution of the function the bit #Head_1.Error_FC_Send is set. If one of the bits is set the execution of the execution of the function is finished and the program jumps to point aus1. At this point the analysis of the error followed or it if necessary a jump to the data import routine followed. In this routine the response of the sending command is import from the IDENTControl. If none of the status bits are set the function is already in execution. That means that the out data field is transferred to the IDENTControl at the moment. In this case the analysis of the execution of the function is not necessary. Instead the program jumps to the Restart routine. At this point a check followed whether a restart is to be executed and the function block is finished. In the next program cycle of the function block the execution of the function is checked again. Afterwards the program jumps to the correspondent point inside the function block.

2011-01

Now the command execution of head 1 is finished. After a successful finish of the command execution and a successful finish of the analysis of the response the execution of head 2 followed.

6.8 Procedure Restart Routine

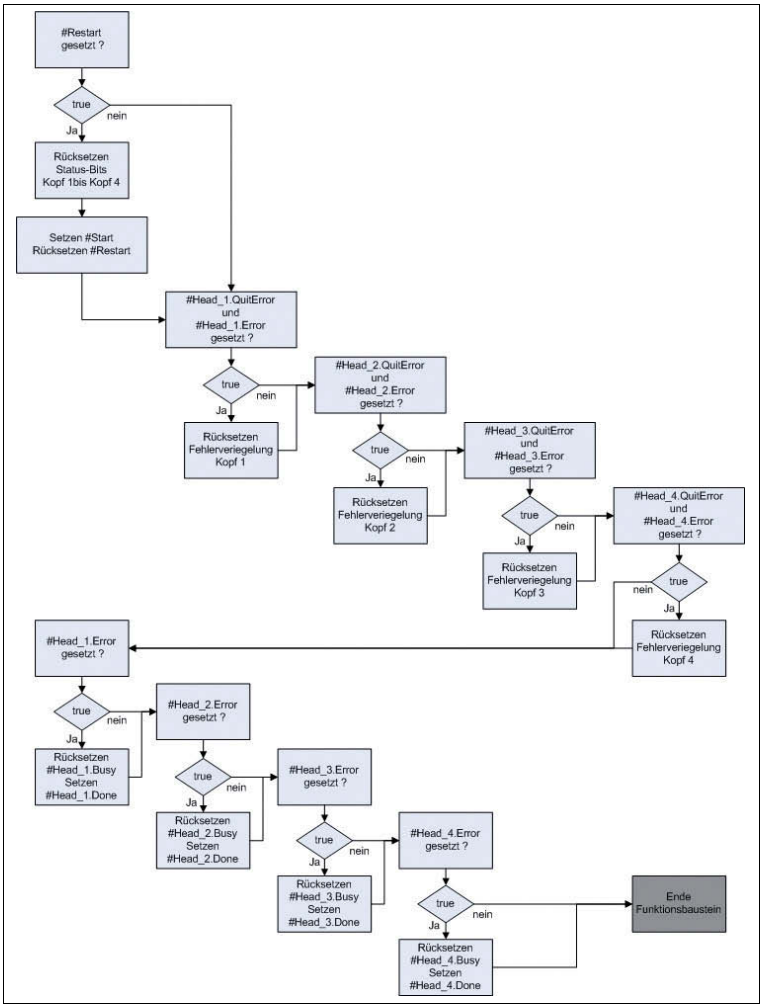


Figure 6.8: Program flow chart of the Restart routine

The task of the Restart routine is to put back the function block into a defined basic state and to finish the function block. Subsequent the Restart routine is described for head 1. The Restart routine for the other heads is analogue.

Firstly the bit #Restart is checked whether it is set. The bit is set, if a restart of the function block is intended. If the bit is not set, the program jumps to point end2. The point end2 is at the beginning of network 19. At this point all bits are reset, which signalize an error status.

If the bit #Restart is set, all necessary status bits of head 1 are reset. The reset of the status bits of the heads 2, 3 and 4 follows directly.

After the reset of the status bits of all heads the bit #Start is set. By setting this bit the program checked the initialisation of the all heads inside the Start-Up-Sequence in the next cycle. The reset of #Restart signalise that the function block is in a defined basic state.

One part of the Restart routine is the acknowledgement of the error bits. This part goes through if the bit #Restart is not set at the beginning of the Restart routine. If the bit is not set the program directly jumps to the acknowledgement of the error bits. The acknowledgement is also passing through if a restart was happen. As soon as an error in the command execution of the heads is detected the specific error bit #Head_X.Error is set. Thus the command execution on this head is locked. The function block gives the user the possibility to unlock the error state of the head. For this the user have to set the IN variable #QuitErrorHeadX. In the part of the transformation of the variables these variables change to the STAT variable #Head_X.QuitError. At the beginning of the acknowledgement part the program checked the bits #Head_1.QuitError and #Head_1.Error. If both bits are set an error in the command execution arose and the user will unlock the error state of the head. After that the specific error and status bits are reset. The acknowledgement of the error state follows directly.

Afterwards a part of the program follows which handle the error notices. The acknowledgement sequence is passing through if the user will unlock the error state of a head. If the error state is not unlocking the specific status bits have to set. The status bits have the task to signalize the abort of a command by an error.

At first the bit #Head_1.Error is checked whether the bit is set. If the bit is set, the bit #Head_1.Busy is reset by the program. This bit signalizes a current execution of a command. Afterwards the bit #Head_1.Done is set. This bit signalizes the end of a command execution on head 1. The reset of the status bits of the other heads follows directly. Then the program jump to point endG. At this point the function block is finished.

6.9 Procedure Analysis of Function FC 50

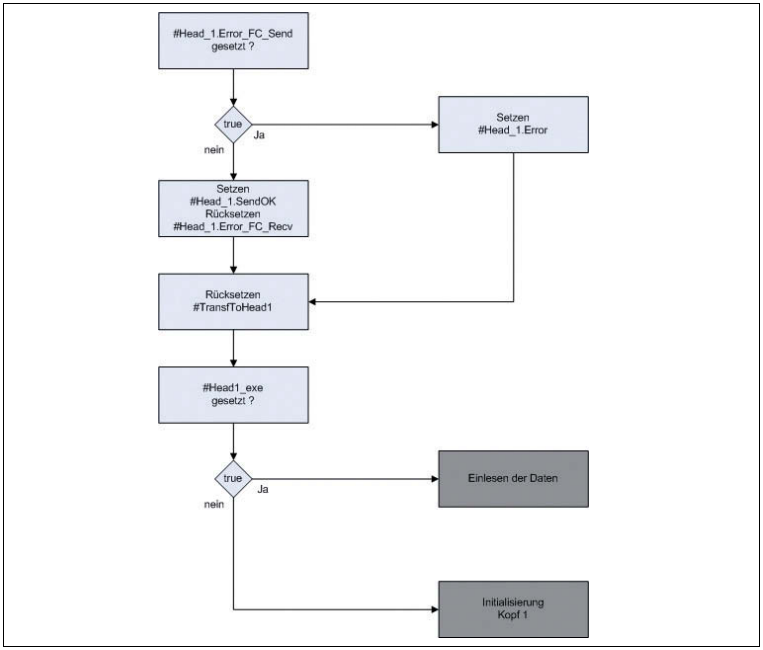


Figure 6.9: Program flow chart analysis FC 50 of head 1

The task of this program part is to analyse the execution of the FC 50. The FC 50 is called inside the function block on two different points. One point is the initialisation and the other point is the command execution. Following the execution of the FC 50 the program jumps to the analysis of the FC 50. With the help of the analysis the program detect a failure in the execution of the FC 50. In this case the out data field of the function block is not transferred to the IDENTControl.

Following the analysis is exemplarily described for head 1. The analysis of the other heads is analogue.

To the beginning of the analysis the variable #Head_1.Error_FC_Send is checked whether it is set. If an error occurred in the execution of the FC 50 the program jumps to point err1. At this point the failure bit #Head_1.Error_FC_Send is set. If the execution of the FC 50 was successful the out data field of head 1 was transferred to the IDENTControl. Thus the bit #Head_1.SendOK is set. Afterwards the bit #Head_1.Error_FC_Recv is reset. The bit is reset at this place because the import of the response followed after the sending of the out data field. Thus the program avoid that the execution of the FC 50 is understand incorrect. Afterwards the program jumps to point BE1. The point BE1 is passing through by the program and not depends of a successful execution of the function FC 50. In this program part the bit #TransfToHead1 is reset at first. Thus it is possible to allocate new

command parameter if no other locking bits are set. Afterwards the bit #Head1_exe is checked. With the help of this bit the return out of the analysis is realised. The bit is set, if the analysis is execute in consequence of the command execution. Afterwards the program jumps to point ReDa into network 8. At this point the import routine of the data is passing through. The bit #Head1_exe is not set, if the analysis is execute in consequence of the initialisation. Thus the program returns to point F501 inside the initialisation.

The analysis of the other heads followed directly.

6.10 Procedure Analysis of Function FC 60

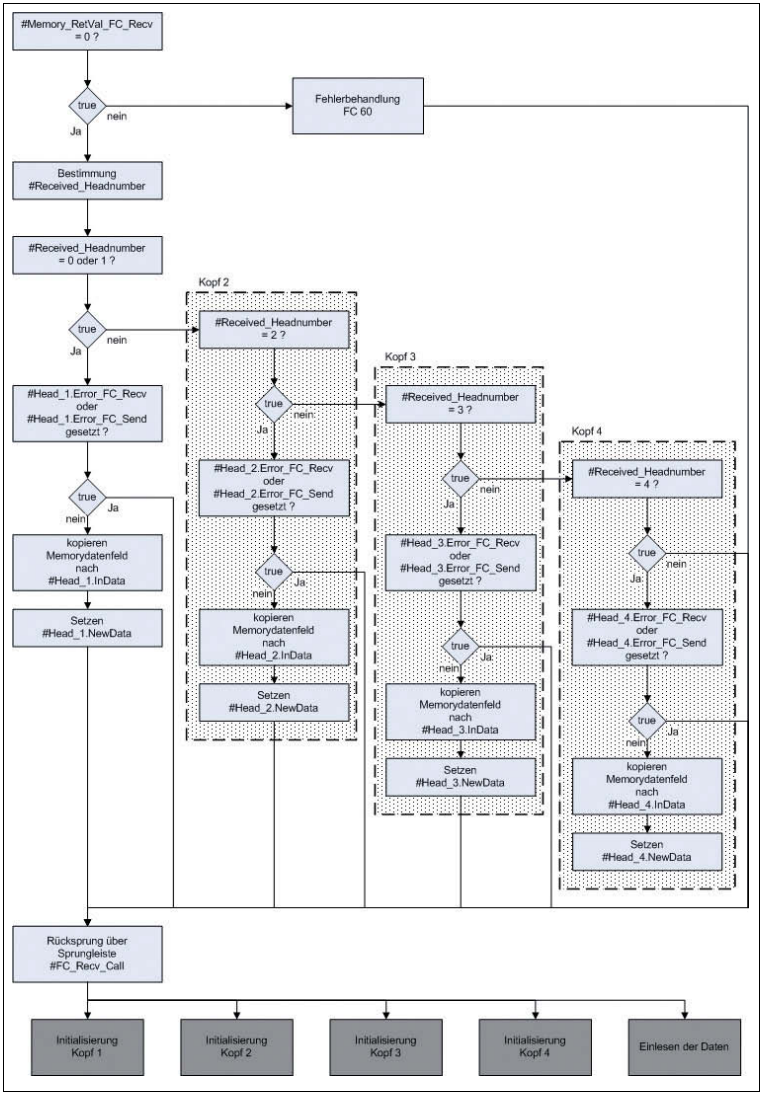


Figure 6.10: Program flow chart analysis FC 60

The task of this program part is to analyse the execution of the FC 60. The FC 60 is called inside the program on two different points. One point is the initialisation and the other is procedure where the data from the IDENTControl import. After the execution of the FC 60 in the different program parts the program jump to the analysis of the FC 60 into network 21. Firstly the execution of the FC 60 is checked whether a error occurred in the execution. Therefore the variable #Memory_RetVal_FC_Recv is checked whether it has a value. The program jumps to er60 in this case. At point er60 the program executes an error handling. But if the execution of the FC 60 was free of failure the head number of the import data will be isolated. The head number of the import data will be allocated to the variable #Received_Headnumber.

Afterwards on the base of the head number the memory data field copied into the head specific in data field. Then the bit #Head_X.NewData is set. This bit signalise that new data are inside the in data fields. The last step is the return to the program part, where the analysis was called.

6.11 Analysis of Input Data Fields

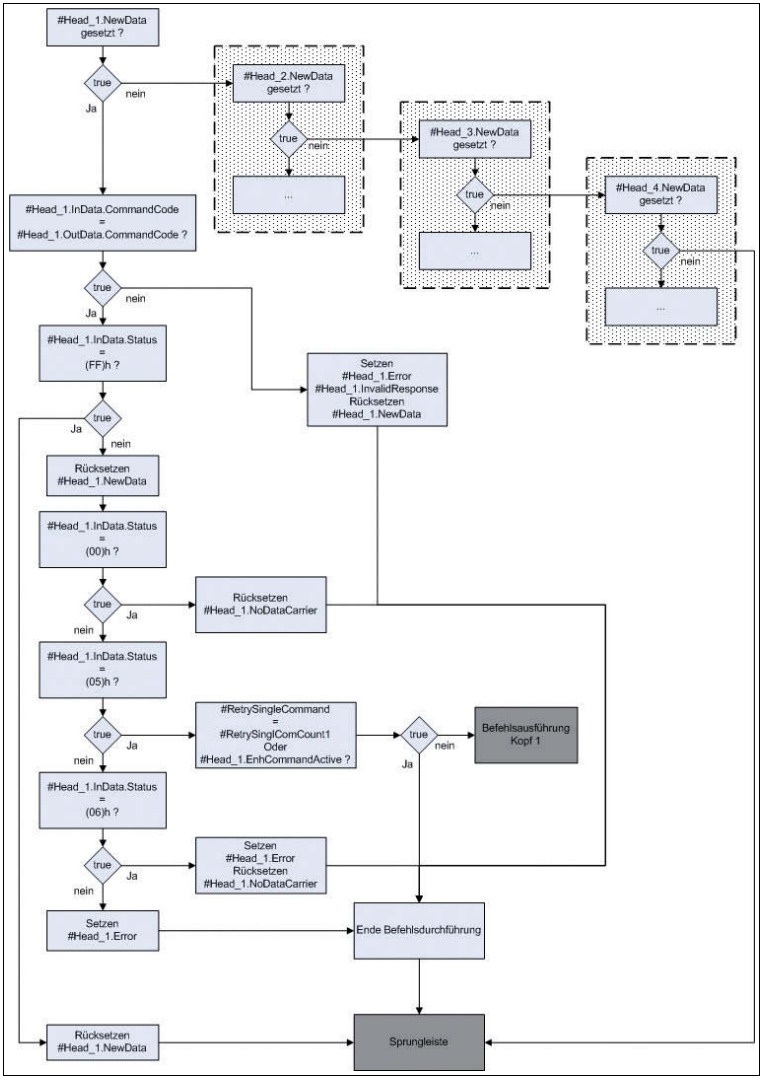


Figure 6.11: Program flow chart Analysis Input Data Fields

This part of the program has the task to analyse the head specific in data fields. The analysis of the in data fields is necessary as soon as new data copied from the memory data field in the specific in data fields.

At the beginning of the analysis the program detect in which in data fields new data exist. If new data exist the bit #Head_1.NewData is set and the program jump to the point where the analysis executes. At this point the analysis starts. Firstly the command code of the out data field is compared to the command parameter of the in data field. If both parameters not identical an error in the execution of the IDENTControl occurred and the program jump to point err1. At this point the error notices #Head_1.Error and #Head_1.InvalidResponse are set.

If both parameters are identical in the next step the status value of the import data is checked. If the status has the value (FF)h, the IDENTControl received the send command but the command is currently executed so that the import data are invalid. That is why the bit #Head_1.NewData reset at point sff1. If the command execution successfully finished by the IDENTControl the status has the value (00)h.

The status of the import data has the value (05)h, if no data carrier was in front of the head whiles the command execution. Thus the executions counter increment and the command execution is repeated. If the maximal number of repetitions is achieving the bit #Head_1.NoDataCarrier is set and the command execution is not repeated. If the import data has got other value an error in the command execution occurred and the failure notice #Head_1.Error is set.

Finally the program return to the point where the analysis of the in data fields was called.

7 Appendix

7.1 Listing of Parameter

7.1.1 Input Parameter (IN-Parameter)

- **IDENT_Control_Address**
  Address of the IDENTControl inside the Ethernet network. The declaration refers to the I/O addresses of the IDENTControl in the Hardware configuration
- **ID**
  Connection number
- **TCP_Telegramlength**
  Telegram length (Port 10000 -> 34 Byte; Port 10001 -> 66 Byte)
- **Timeout**
  Time slot for Timeout control
- **RetrySingleCommand**
  Number of maximal command repetitions
- **HeadXDataFixcode**
  HeadXDataFixcode = 0 -> access to Fixcode
  HeadXDataFixcode = 1 -> execution of enhanced command
- **HeadXSingleEnhanced**
  HeadXSingleEnhanced = 0 -> execution of single command
  HeadXSingleEnhanced = 1 -> execution of enhanced command
- **HeadXQuit**
  Start of a quit command to abort an enhanced command
- **HeadXRead**
  Start of a read command
- **HeadXWrite**
  Start of a write command
- **QuitErrorHeadX**
  Start a quit error command to unlock error locking
- **HeadXSpecialCommand**
  Start of a special command
- **IC_Command_on_Head1**
  Start of an IDENTControl command. An assignment of a channel is not necessary. The command is not executing by the read / writes heads

7.1.2 Pass Parameter (IN-OUT-Parameter)

- **InitFinish**
  End of the initialization of all heads
- **SetRestart**
  Start of the Restart routine

7.1.3 Static Parameter (STAT-Parameter)

- **Head_X.InData**
  Head_X.InData is a structure of data which contained the response of the IDENTControl if a command was sending. The structure consists of different elements of data types.
  - **Head_X.InData.TelegrammLengthHigh**
    High Byte of telegram length
  - **Head_X.InData.TelegrammLengthLow**
    Low Byte of telegram length (contains #TCP_Telegramlength)
  - **Head_X.InData.CommandCode**
    Command code of the response telegram
  - **Head_X.InData.Channel**
    The high nipple contained the number of read/write words. The low nipple contains the channel identification.
  - **Head_X.InData.Status**
    Status value of the command execution
  - **Head_X.InData.ExecutionCounter**
    Event counter
  - **Head_X.InData.DW1 … DW15**
    Data field of user data which are import front IDENTControl (1 data block = 4 Bytes).
- **Head_X.OutData**
  Head_X.OutData is also a structure of data. This structure contains the data, which send to the IDENTControl to execute the defined command. The data field is divided into element with different data types.
  - **Head_X.OutData.TelegrammLengthHigh**
    High Byte of telegram length
  - **Head_X.OutData.TelegrammLengthLow**
    Low Byte of telegram length
  - **Head_X.OutData.CommandCode**
    Command code of execute command
  - **Head_X.OutData.Channel**
    Channel identification
  - **Head_X.OutData.Wordadr_High**
    High Byte of start address, where data read/write in the memory area; if change tag command -> high Byte of Tag identification code
  - **Head_X.OutData.Wordadr_Low**
    Low Byte of start address, where data read/write in the memory area; if change tag command -> low Byte of Tag identification code
  - **Head_X.OutData.DW1 … DW15**
    Data field of user data which are export to the IDENTControl (1 data block = 4 Byte)
- **Head_X.WordAddress**
  Start address for access of memory area
- **Head_X.TimeoutActiv**
  Timeout control is active
- **Head_X.InvalidResponse**
  Invalid response of the IDENTControl

2011-01

- **Head_X.QuitError**
  Unlock the failure locking on channel X
- **Head_X.NewData**
  New data are available for analysis on channel X
- **Head_X.NotExist**
  No read / write head is connected to the channel X
- **Head_X.ExistTC**
  Read / write head is connected to channel X
- **Head_X.Error**
  Error in the command execution of the IDENTControl
- **Head_X.TimeoutOccured**
  Time slot of the command response is passed
- **Head_X.ReceiveOK**
  Response of the IDENTControl received
- **Head_X.SendOK**
  Data field was send to the IDENTControl
- **Head_X.NoDataCarrier**
  No data carrier in front of the head
- **Head_X.Done**
  Enhanced command -> data read / write finish (command already active)
  Single command -> command execution finished
- **Head_X.Busy**
  Command is in processing
- **Head_X.Error_FC_Recv**
  Error by the execution of the FC 60
- **Head_X.Error_FC_Send**
  Error by the execution of the FC 50
- **Head_X.EnhCommandActive**
  Enhanced command is active
- **Head_X.SglCommandActive**
  Single command is active
- **Head_X.WordNum**
  Number of transmitted user data blocks
- **Head_X.RetVal_FC_Recv**
  Contains an error value by the execution of the FC 60
- **Head_X.RetVal_FC_Send**
  Contains an error value by the execution of the FC 50
- **Head_X.SpecialCommand**
  This data field contains the parameter to execute a special command. With the help of the special command you can execute commands which are no standard commands of the function block. Before starting the execution of the special command the user have to transfer the command parameter in this data field. This data field copied into the out data field and transfer to the IDENTControl.
  - **Head_X.SpecialCommand.Code**
    Command code

- **Head_X.SpecialCommand.Channel**
  Channel identification and possibly the number of transferred user data blocks.
- **Head_X.SpecialCommand.Parameter1 …5**
  Other parameter of the command

- **Head_X.Memory**
  The memory data field contains data which sent from the IDENTControl to the PLC. All data, which import from the IDENTControl stored in this data field, independently of the channel. After the check of the channel identification the data copied into the specific in data fields of the heads. The structure of the memory data field is the same like in data fields of the channels.

7.2 Command List

Command	Code	Parametrisazion	Execution
Single-Read-Fixcode	(01)h	none	#HeadXDataFixcode = 1 #HeadXRead = 1 #HeadXWrite = 0 #HeadXSingleEnhanced = 0
Enhanced-Read-Fixcode	(1D)h	none	#HeadXDataFixcode = 1 #HeadXRead = 1 #HeadXWrite = 0 #HeadXSingleEnhanced = 1
Single-Read-Data	(10)h	#HeadX.WordAddress #HeadX.WordNum	#HeadXDataFixcode = 0 #HeadXRead = 1 #HeadXWrite = 0 #HeadXSingleEnhanced = 0
Enhanced-Read-Data	(19)h	#HeadX.WordAddress #HeadX.WordNum	#HeadXDataFixcode = 0 #HeadXRead = 1 #HeadXWrite = 0 #HeadXSingleEnhanced = 1
Single-Write-Data	(10)h	#HeadX.WordAddress #HeadX.WordNum #HeadX.OutData.DW	#HeadXDataFixcode = 0 #HeadXRead = 0 #HeadXWrite = 1 #HeadXSingleEnhanced = 0
Enhanced-Write-Data	(19)h	#HeadXWordAddress #HeadXWordNum #HeadX.OutData.DW	#HeadXDataFixcode = 0 #HeadXRead = 0 #HeadXWrite = 1 #HeadXSingleEnhanced = 0
Special-Command	(??)h	#Head_X.SpecialCommand.Code #Head_X.SpecialCommand.Channel #Head_X.SpecialCommand.Parameter	#HeadXSpecialCommand = 1 #IC_Command_on_Head1 = 0
IDENT-Control-Command	(??)h	#Head_1.SpecialCommand.Code #Head_1.SpecialCommand.Parameter	#Head1SpecialCommand = 1 #IC_Command_on_Head1 = 1
QuitError-Command	-	-	#QuitErrorHeadX
Quit-Command	-	-	#HeadXQuit

7.3 Code/Data Carrier

Typ	#Head_X.TagType	Access	Memoryarea	Fixcodelength
IPC02-..	W#16#3032	Read Fixcode	-	5 Byte
IPC03-..	W#16#3033	Read Fixcode Read Data Write Data	116 Byte	4 Byte
IPC10-..	W#16#3130	Read Data Write Data	12 Byte	-
IPC11-..	W#16#3131	Read Data Write Data	5 Byte	-
IPC12-..	W#16#3132	Read Fixcode Read Data Write Data	8 kByte	4 Byte
IPC14-..	W#16#3134	Read Data Write Data	5 Byte	-
IQC20-..	W#16#3230	Read Fixcode Read Data Write Data	-	8 Byte
IQC21-..	W#16#3231	Read Fixcode Read Data Write Data	112 Byte	8 Byte
IQC22-..	W#16#3232	Read Fixcode Read Data Write Data	256 Byte	8 Byte
IDC-…1k	W#16#3530	Read Fixcode Read Data Write Data	128 Byte	4 Byte
ICC-..	W#16#3532	Read Fixcode	-	7 Byte
MVC-60	W#16#3630	Read Fixcode Read Data Write Data	-	8 kByte

FACTORY AUTOMATION – SENSING YOUR NEEDS

Worldwide Headquarters
Pepperl+Fuchs GmbH
68307 Mannheim · Germany
Tel. +49 621 776-0
E-mail: info@de.pepperl-fuchs.com

USA Headquarters
Pepperl+Fuchs Inc.
Twinsburg, Ohio 44087 · USA
Tel. +1 330 4253555
E-mail: sales@us.pepperl-fuchs.com

Asia Pacific Headquarters
Pepperl+Fuchs Pte Ltd.
Company Registration No. 199003130E
Singapore 139942
Tel. +65 67799091
E-mail: sales@sg.pepperl-fuchs.com

www.pepperl-fuchs.com

PEPPERL+FUCHS
SENSING YOUR NEEDS

TDOCT1190A_ENG
01/2011