Implementation of the Identification System IC-KP-B6 to a SIMATIC S-7 400 PLC

PEPPERL+FUCHS
IDENT

Contents

1. Hardware specifications

1.1 Equipment and Devices

In the following chart are all components listed used to connect the identification system IDENTControl IC-KP-B6 to SIMATIC S7-400 PLC with Profibus interface.

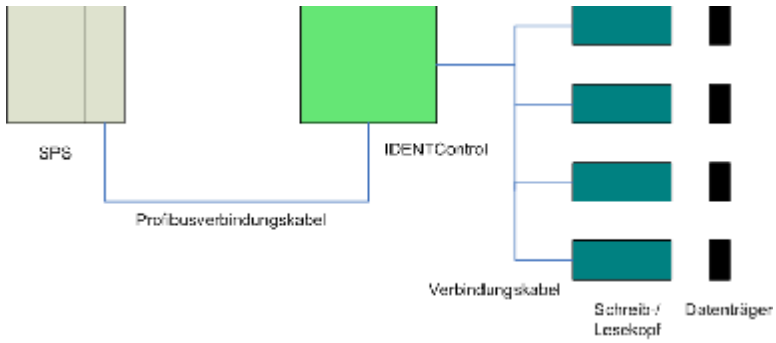
EQUIPMENT	NAME OF MANUFACTURERS
Simatic S-7 400 Power Supply	PS 407 4A
Simatic S-7 400 CPU	CPU-412-2
IDENTControl	IDENTControl IC-KP-B6-SUBD/-V15B
4 x Read-/Write Heads	IQH-18GM-V1
4 x Connection Cable Read-/Write Heads	V1-G-0,6M-PUR-V1-W
Profibus Trunk Cable	-
Data Carrier	IQC21-58
1 x Connection Cable IDENTControl	V1-G-2M-PUR

Table 1: List of used Hardware-Equipment

The equipment is extract of the test composition explain in this manual. The commissioning of the IDENTControl IC-KP-B6 cans also execute by other PLC devices with the same functionality. The information of the installation of the PLC devices you get out of the manuals of the PLC.

1.2 Configuration and Installation

Following graphic shows the build up of the Hardware-Equipment.



Picture 1: Build up of Hardware Equipment

Firstly the connection cables of the Read / Write Heads connect to the associated females of the IDENTControl. Here you have to regard to the distance between two Heads. Because the Heads are influences each other. The necessary distance you get out of the data sheet of the Head.

The data carriers have to be in front of the Read / Write Heads. But it has to regard that only one carrier is inside the field of the head.

Afterwards the IDENTControl has to connect to the power supply 20…30 V DC by the connection cable.

Finally build up the communication access between IDENTControl and PLS by the Profibus connection cable. If the IDENTControl is the last device in the Profibus trunk the cable ending have to be terminated by a cable terminator. In this place you do not get more information about the buildup of the different PLC equipment. For more information see the manual “PLC S7-400 Installation guide”. For more information about the installation of the IDENTControl look inside the manual „IDENTControl IC-KP-B6“(www.pepperl-fuchs.com).

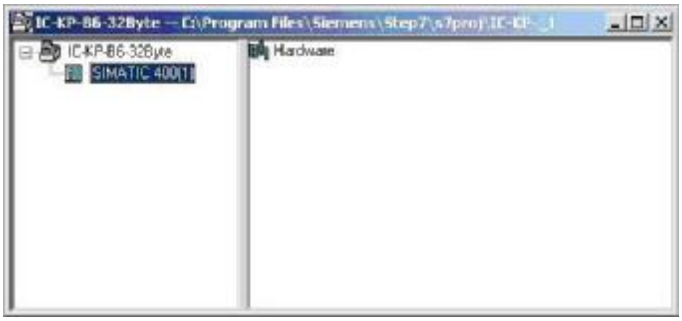
1.3 Configuration of the PLC

In this part is a description of the Hardware configuration inside the Software “SIMATIC Manager“. The configuration is shown with an example. The Parameter of the configuration has to match to the operating conditions of the customer. The Parameter can directly prompt on the display. To parameterize the IDENTControl see the manual “IDENTControl IC-KP-B6“(www.pepperl-fuchs.com). In the delivery status of the IDENTControl default parameter adjusted. The default parameterization you get out of the manual “IDENTControl IC-KP-B6”. In the following chart the parameter of the example program listed.

PARAMETER	VALUE
Profibus Address IDENTControl	3
Transfer rate	1,5 MBit/s
Tag Type	„21“ **à** IQC21

Table 2: Parameter of Hardware Configuration

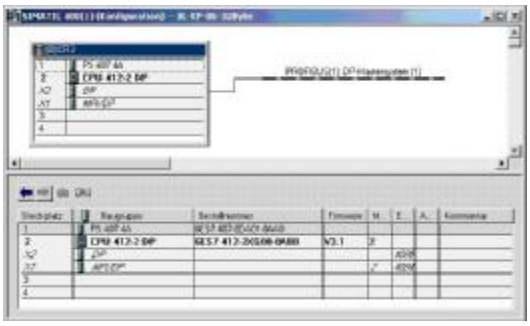
For commissioning the IDENTControl you have to start a new project. After the start of the new project the components of the PLC have to involve into the Hardware Configuration. Firstly the SIMATIK 400 Station has to be implementing.



Picture 2: Involvement SIMATIC400-Station

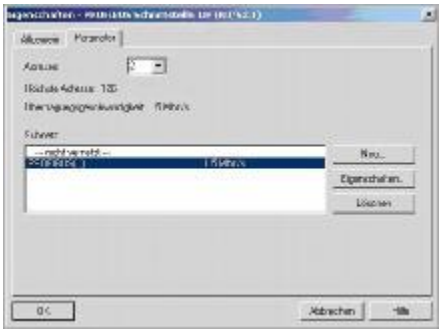
Afterwards the Hardware Configuration is called by double click on the symbol “Hardware”. Inside the Hardware Configuration the required components rack, power supply as soon as CPU. In the next step the DP Master system have to be imple-

mented. The DP Master system realized the communication. The following picture shows the involvement of the Hardware components.



Picture 3: Hardware-Configuration

Next the attributes of the DP communication interface of the PLC has to be defined. For this you have to open the properties window of the DP interface. By double click on the symbol in the picture above the properties window will be open. In the window the actual properties were shown. To change the properties the field “Properties“has to be clicked. Thereby the following window opens.



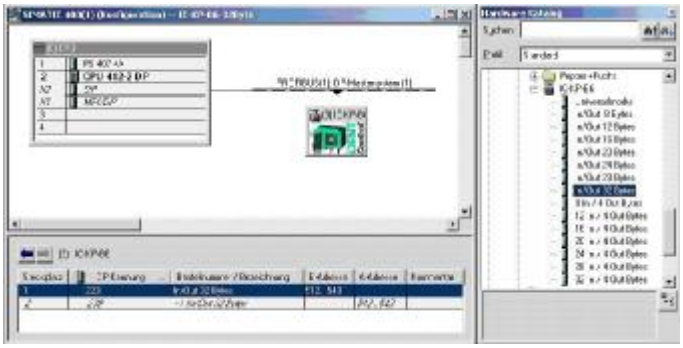
Picture 4: Setting DP-Communication interface of PLC

In this window the parameter of the DP interface of the PLC define. Firstly the Profibus address of the PLC is to define. In the last part the subnet has to be linked new. The properties of the communication network have to be defining in a separate window. The new defined parameters are activated by the field “OK”. The following picture shows the window to parameterize the Profibus communication.

Picture 5: Setting Profibus Communication

In the next step the IDENTControl involved into the Profibus network. For this the symbol „IC-KP-B6“has to be dropped out of the Hardware catalogue and put into the

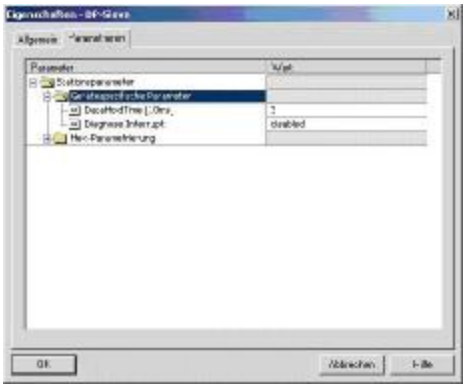
DP Master system. Afterwards the telegram length has to be defined. For this the attribute “IN/OUT X Bytes” drop out of the Hardware catalogue and put inside the connection chart of the lower picture part.



Picture 6: Setting Communication interface IDENTControl

In the example of the commissioning of the IDENTControl the telegram length is defined of 32 Bytes. The maximal telegram length depends on the CPU used. More information you get out of the manual of the used CPU.

To define the Hardware configuration the parameterization of the Data Hold Time is necessary. The Data Hold Time is a device specific parameter and defines the time the data of the IDENTControl are inside the Out data field of the IDENTControl. The defined value of the Data Hold Time has to be greater than the cycle time of the OB1.



Picture 7: Setting Data Hold Time

Now the involvement of the IDENTControl is finished in the next step the integration of the Software of the identification system followed.

2. Software

2.1 Commands of the PLC Program

In this part the different command types of the IDENTControl describe which can execute by the PLC program. Afterwards a description of the execution of a command follows. In the last part the structure of an example command is shown.

2.1.1 Types of Commands

The commands executed by the IDENTControl with the help of the PLC program can separated in two types. You can differentiate a Single and an Enhanced Command.
The Single command executed once if a Code / Data carrier is inside the fields of the read write head. If the program recognize that no carrier is inside the field the PLC program repeat the command execution to the command is executed completed or the maximal numbers of command repetitions is exceeded. The number of maximal command repetitions is defined by the User. It is necessary to prevent the blocking of the head by durable command execution.
For different applications a durable command execution is necessary. The durable command execution is realized by the Enhanced command. The Enhanced command is so long executed as a Quit command or an Error terminates the command.

2.1.2 Command Cycle

At first the command parameters which are necessary to execute a command transmit by the PLC program to the IDENTControl. The IDENTControl sent with the acceptance of the command a status (Status = (FF)h) response back to the PLC. The status response signalize that the IDENTControl accept and execute the command. Afterwards the IDENTControl transmit the command to the specific head and wait for response of the head. The response of the head passes to the PLC by the IDENTControl. Inside the PLC the import data analyse and made available to the User. If an Enhanced command is execute the responses of the heads passes to the PLC until the command is interrupted by a Quit command or an Error.

2.1.3 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.CommandCode	0	0	0	1	0	0	0	0
1	#Head_X.OutData.Channel	Number data words				Channel number			T

2	#Head_X.OutData.Wortadr_High	Start address high Byte
3	#Head_X.OutData.Wortadr_Low	Start address low Byte
4..63	#Head_X.OutData.DW1 ... DW15	unused

Table 3: Structure telegram Single Read Data command

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 definition “IN/OUT X Bytes” inside the Hardware configuration. By the definition of “IN/OUT 64 Bytes” you can maximal transmit 15 data blocks. In this example the definition “IN/OUT 32 Byte” is used. So you can maximal transmit 7 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)
3	6	(0110)
4	8	(1000)

Table 4: channel coding of the Read / Write heads

By the coding of the channel you consider that the LSB is the Toggle bit. The Toggle bit is used to differentiate following identical commands. To differentiate the commands the bit has to be toggled.

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.CommandCode	0	0	0	1	0	0	0	0
1	#Head_X. InData.Channel	Numberdatawords				Channel number			T
2	#Head_X. InData. Status	Status							
3	#Head_X. InData. ExecutionCounter	Event counter							
4..(31)63	#Head_X. InData.DW1 ... .DW(7)15	00.. FF							

Table 5: Structure response telegram Single Read Data command

The response telegram send back the first two 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.

2.1.4 Execution of the Initialization

In the first step of the treatment of the function block the read / write heads have to be initialized. The initialisation is executing successfully for every head. By the initialisation recognize on which IDENT channel a read / write head is connected.

The initialization routine sent a change tag command to the IDENTControl. With this command communicates to the head which tag he is talked to. On the basis of the status massage the function block recognise whether a head is connected on the appropriate channel.

If the initialization of one head is finished the Bit #Head_X.ExistTC or #Head_X.NotExist is set. The Bit #Head_X.ExistTC is set, if a head is connected to the appropriate channel. Otherwise the Bit #Head_X.ExistTC is set.

By the execution of the initialization must be considered that the correct Tag ID has to be assigned. 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-B6“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 hexa-

decimal 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.

2.1.5 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-B6”.

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.

2.1.6 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.

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.

2.1.7 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-B6”.

The execution of a special command is started by setting the IN-Variable #HeadX-SpecialCommand. 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 IN-Variables #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.

2.1.8 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.

PARAMETER	MEANING
#Head_X.Error	Error command execution
#Head_X.InvalidResponse	Invalid response of IDENTControl
#Head_X.TimeoutOccured	Timeout exhausted
#Head_X.Error_SFC_14	Error execution SFC 14
#Head_X.Error_SFC_15	Error execution SFC 15

#Head_X.RetVal_SFC14	Error code SFC 14
#Head_X.RetVal_SFC15	Error code SFC 15
#Memory.Error_SFC14	Error execution SFC 14
#Memory.RetVal_SFC14	Error code SFC 14
#RetVal_SFC20	Error code SFC 20

Table 6: 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.

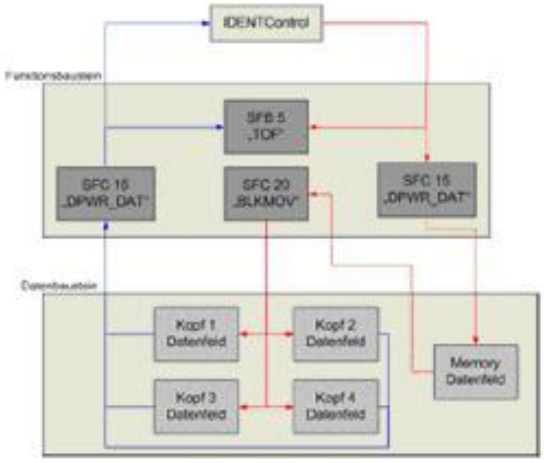
2.2 Used Modules and Functionality

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.
SFC 14 „DPRD_DAT"	System Function	Reading data of DP-Slaves.
SFC 15 „DPWR_DAT"	System Function	Writing data to DP-Slave.
SFC 20 „BLKMOV“	System Function	Coping of memory area.
SFB 5 „TOF“	System Function Block	Realize release delay.

Table 7: Used Blocks and Functionality

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



Picture 8: 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 from the PLC to IDENTControl is realised by the system function SFC 15. The function sends data via a planned Profibus connection to a parameterized DP slave. 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 SFC 14. 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.

2.3 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“

2.4 Function Block „IDENTControl“

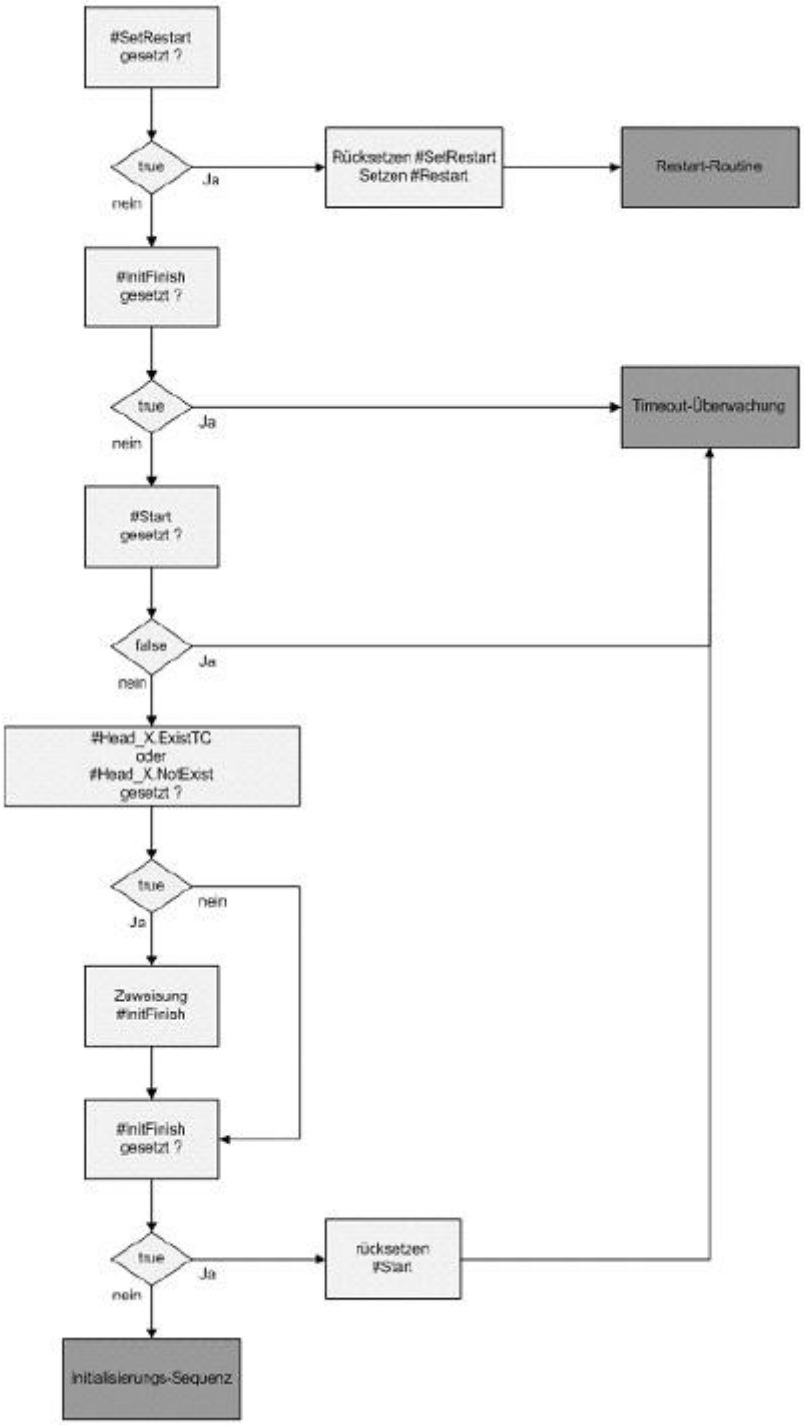
The function block “IDENTControl” has got the task to connect the identification system IDENTControl to the PLC. The function block is user programmable. Thereby the functionality of the function block can adjust to the application of the customer.

The function block is divided in several networks. Following chart shows the several networks and the functionality.

NETZWORK	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	Analysis SFC 15
20	Analysis SFC 14
21	Analysis Input Data Fields

Table 8: Networks of the Function Block “IDENTControl”

2.4.1 Procedure Start-UP-Sequence



Picture 9: 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.

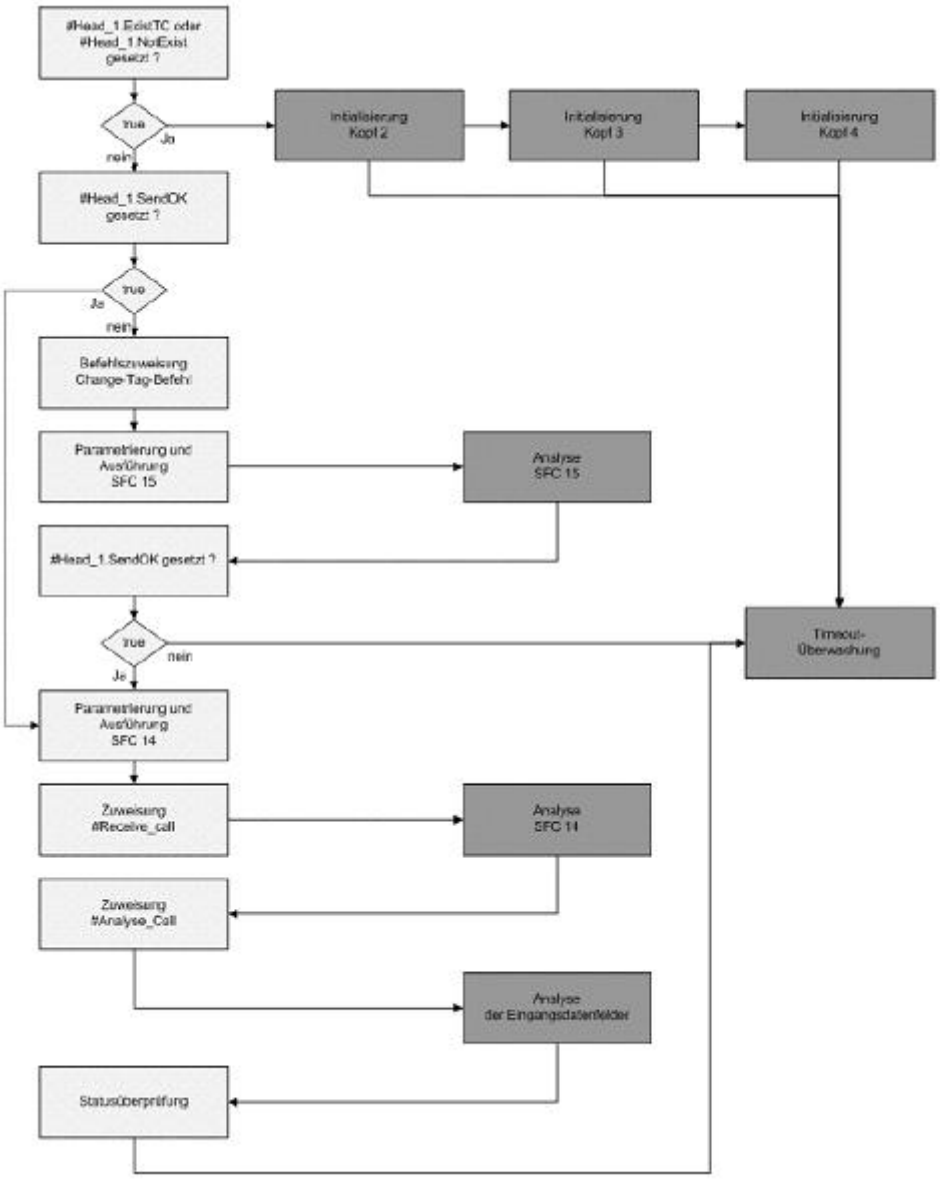
Firstly the bit #SetRestart is checked. The condition of a restart is fulfilled, if the bit is set. Then the program jump to the point sres in the network 17.

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.

2.4.2 Procedure Initialization



Picture 10: 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 Profibus is realized by the system function SFC 15. Before the function is called the function have to parameterize. The parameterization is doing by an ANY parameter. Afterwards the function SFC 15 is called. After a successful execution the program jumps to the point aus1 to the network 19. At this point an error analysis is executed.

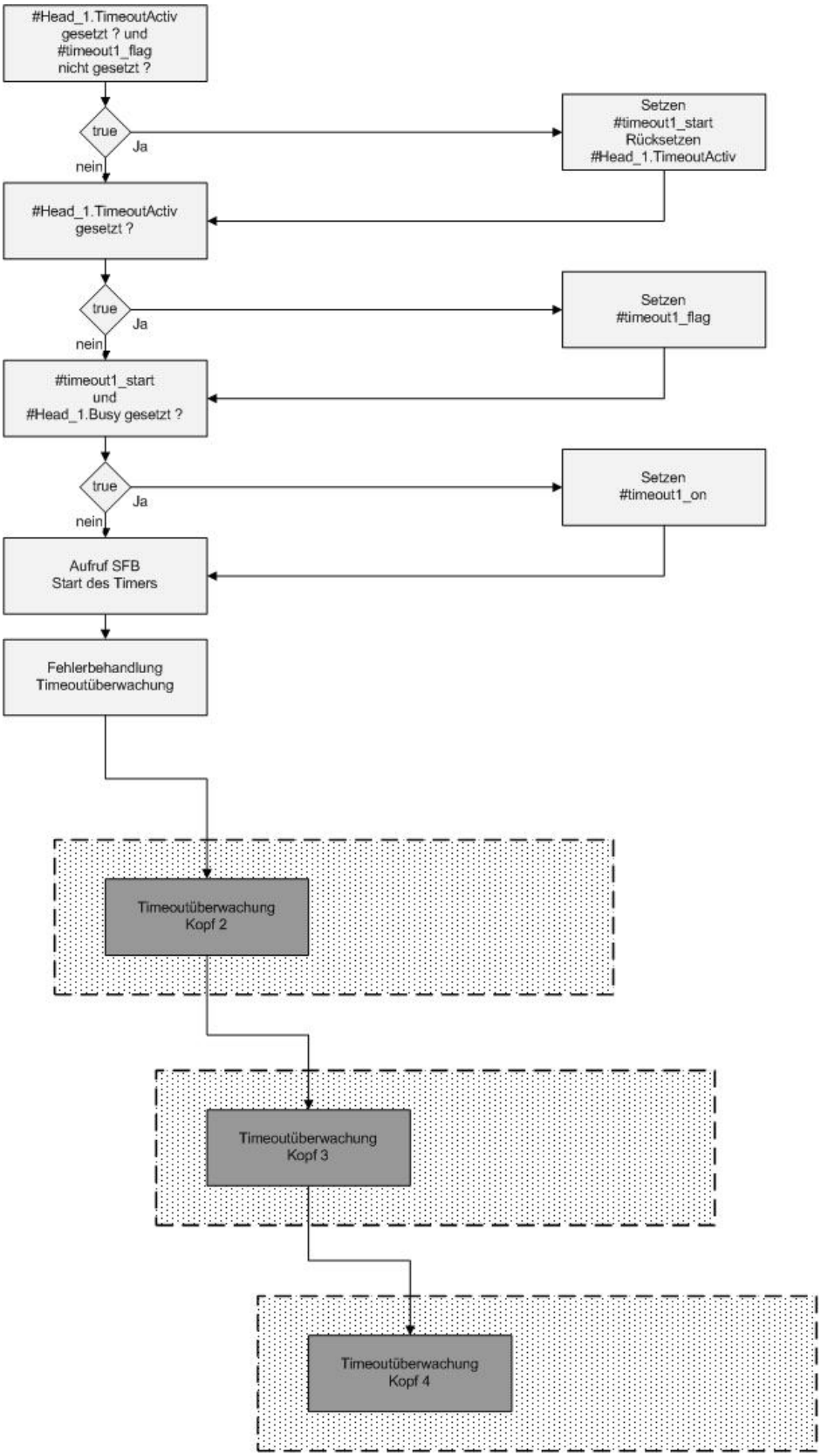
After a successfully execution of the error analysis and no error is recognize the program jump back to the point F151 in network 2.

At the point F501 the bit #Head_1.SendOK is checked. The bit is set, if the function SFC 15 is successful and without errors finished. In this way the program jumps to point rec1. Else, the bit is not set, the program jump to the timeout control into 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 system function SFC 14.

To the point rec1 the necessary parameterization is transferred into the ANY parameter. Afterwards the call of the function SFC 14 follows. Closing the value 0 is transferred to the variable #Receive_call.

2.4.3 Procedure Timeout Control



Picture 11: Program flow chart Timeout Control

The Timeout Control has the task to check the maximal response time of the IDENT-Control. Another task is to display an error notice if the timeout exhausted. In the following describes the Timeout Control for head1. The other heads are analogue.

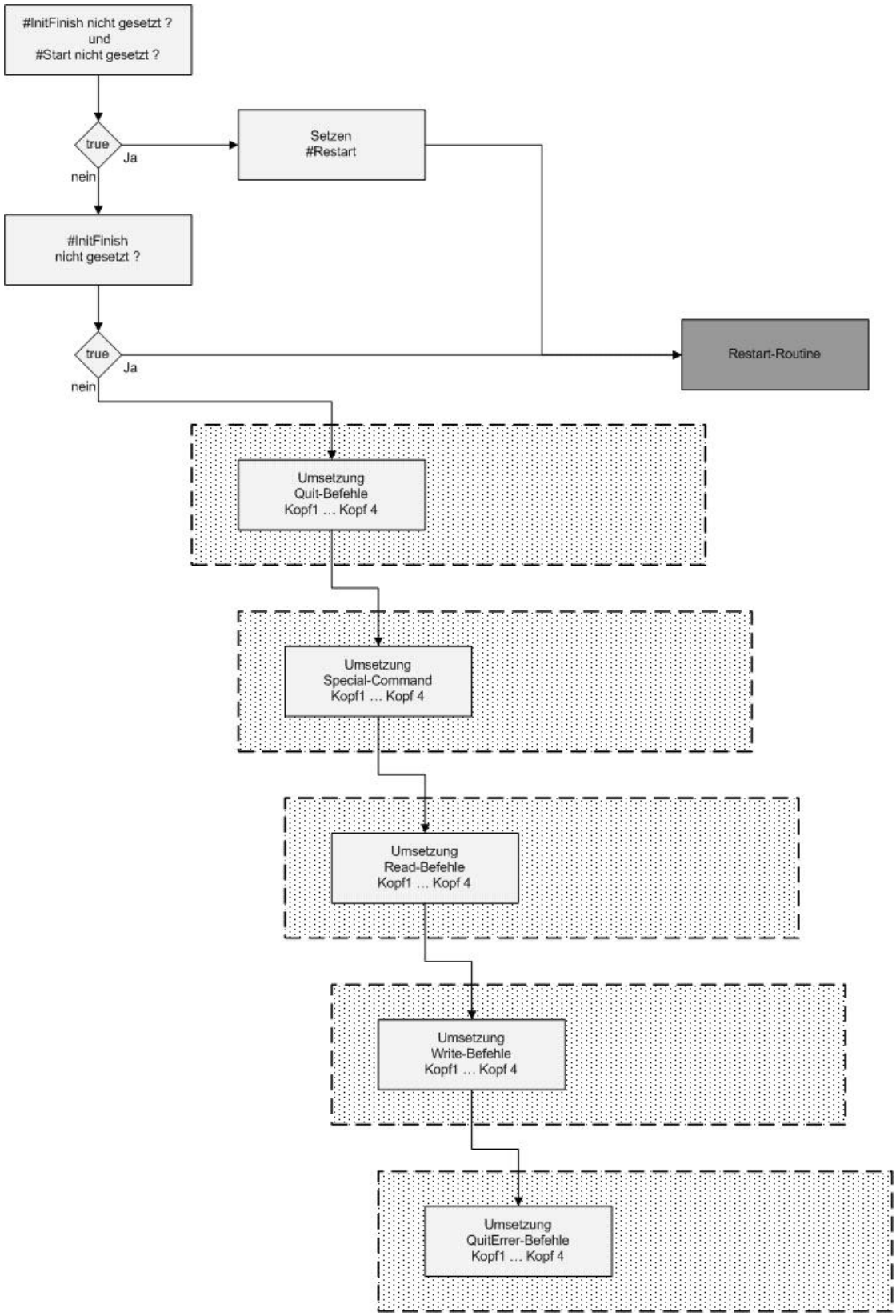
Firstly the bit #Head_1.TimeoutActiv is checked whether it is set and the bit #timeout1_flag whether it is not set. The assignment result is transferred to the variable #timeout1_start. If the condition is fulfilled, so the variable #Head_1.TimeoutActiv is reset. Otherwise only the setting of #Head_1.TimeoutActiv 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 system function SFC 15 is called. The variable #timeout1_flag signalize that the timeout is running at the moment and could not be extended.

2.4.4 Procedure Variable Transformation IN- to STAT-Variables



Picture 12: 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.

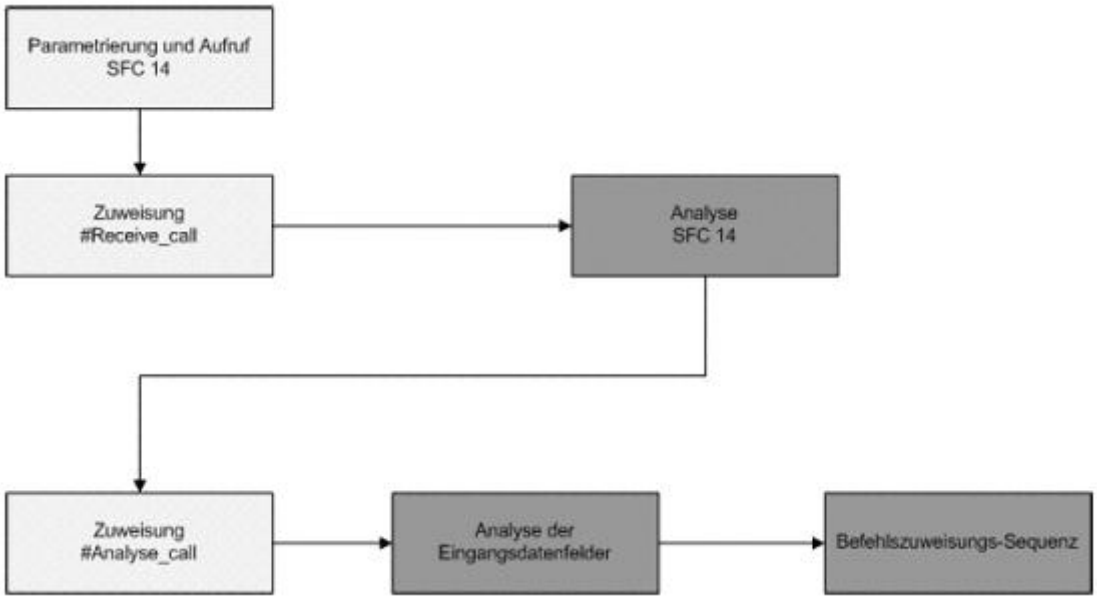
Otherwise the bit #InitFinish is checked. The user can initiate the restart of the function block by reset this bit. If the initialization is not finished a jump to the restart routine followed and the initialisation is added in the next program cycle.

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.

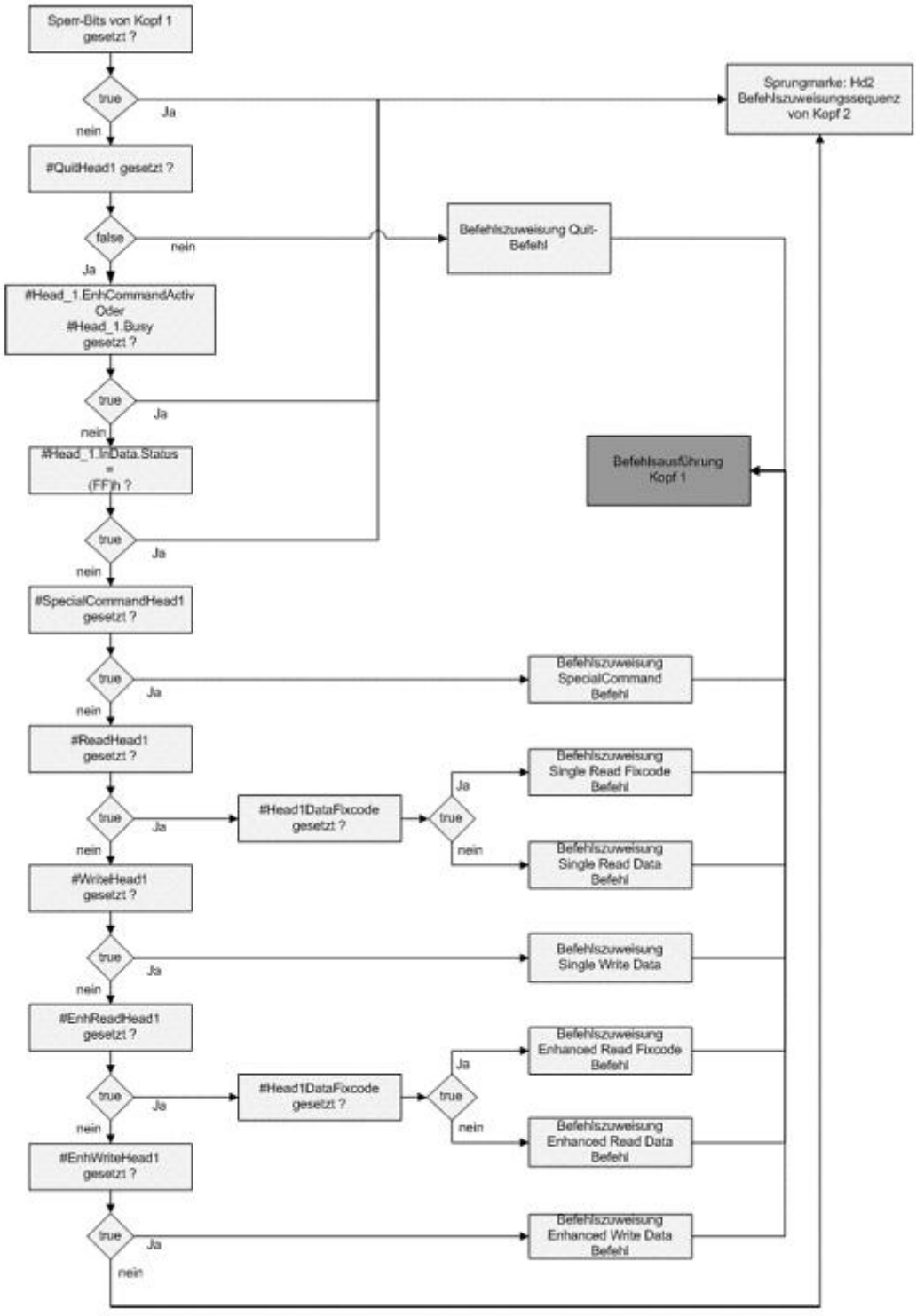
2.4.5 Procedure Read Data



Picture 13: Program flow chart Data Import

This network hast the task to import data from the IDENTControl by using the function SFC 14. At the beginning the required parameters transferred to an ANY parameter type. Afterwards the function is called and the data imported from the parameterized slave address. Next the value 4 is transferred to the variable #Receive_call and the program jump to the point aus5 into the network 20. At this point the import data are analyzed and copied in the head specific input data fields. With the help of the variable #Receive_call the jump back to the point F145 into network 8 is realized. At this point the value 4 is transferred to the variable #Analyse_call. Then the program jump to the point lyse into the network 21. At this point the head specific data fields analyzed. After analysis of the input data fields the program jumps to the point ana5 to the start of network 9. Than begins the declaration of command parameters.

2.4.6 Procedure Command Allocation of Head 1



Picture 14: Program flow chart of Command Allocation 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 execution of head 1 (exe1) 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.

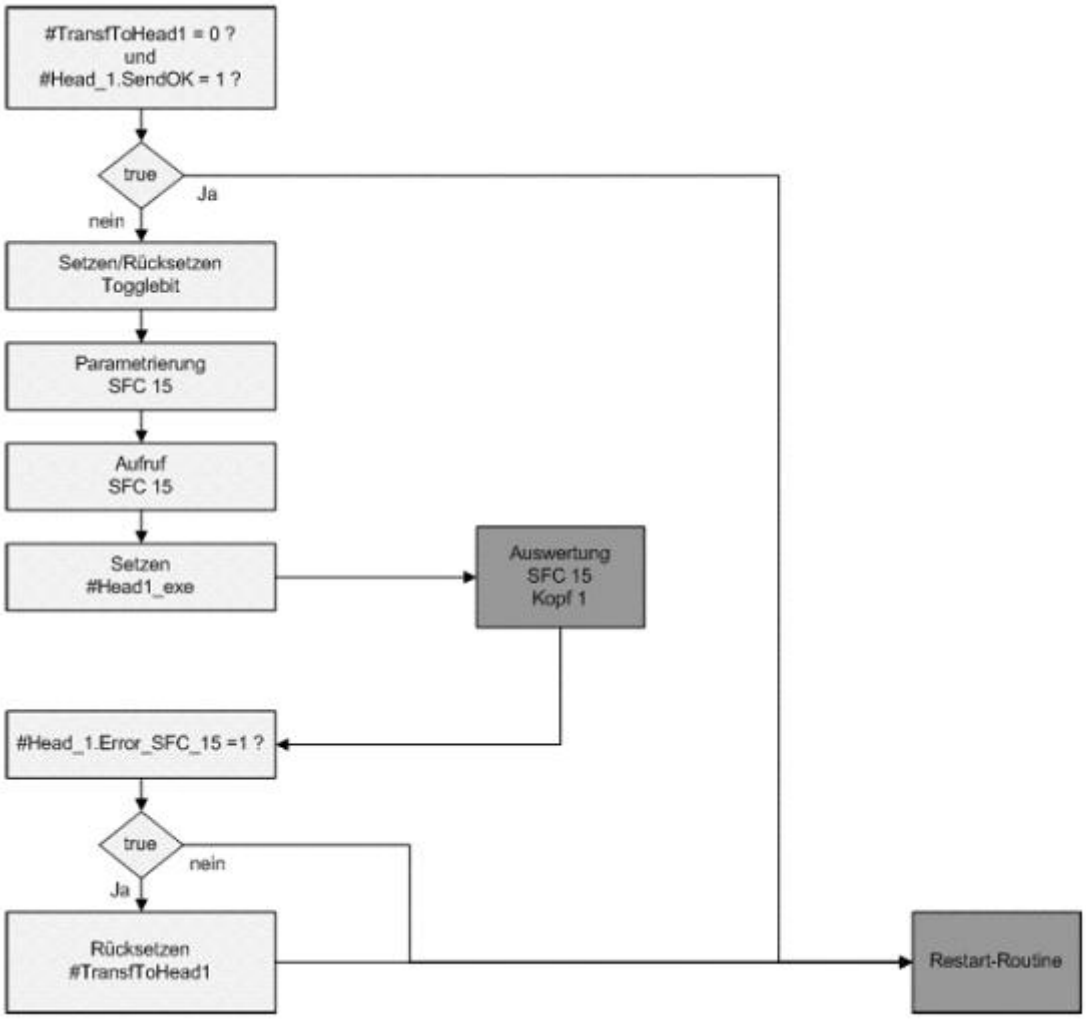
If the execution of a special command on head 1 is intend this is signalised by the bit #SpecialCommandHead1. If the bit is set, the program jumps to point SH1. At this point the command parameters, which defined by the user, transfer into the out data field of head 1.

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). After the command parameter allocation of head 1 the program jumps to point exe1. At this point the function SFC15 is called and the out data field of head 1 will be transmitted to the IDENTControl.

2.4.7 Procedure Command Execution of Head 1



Picture 15: 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 system function SFC 15. 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 point exe1 is firstly examines the bits #TransfToHead1 and #Head_1.SendOK. If the bit #TransfToHead1 is not set and the bit #Head_1.SendOK is set then a com-

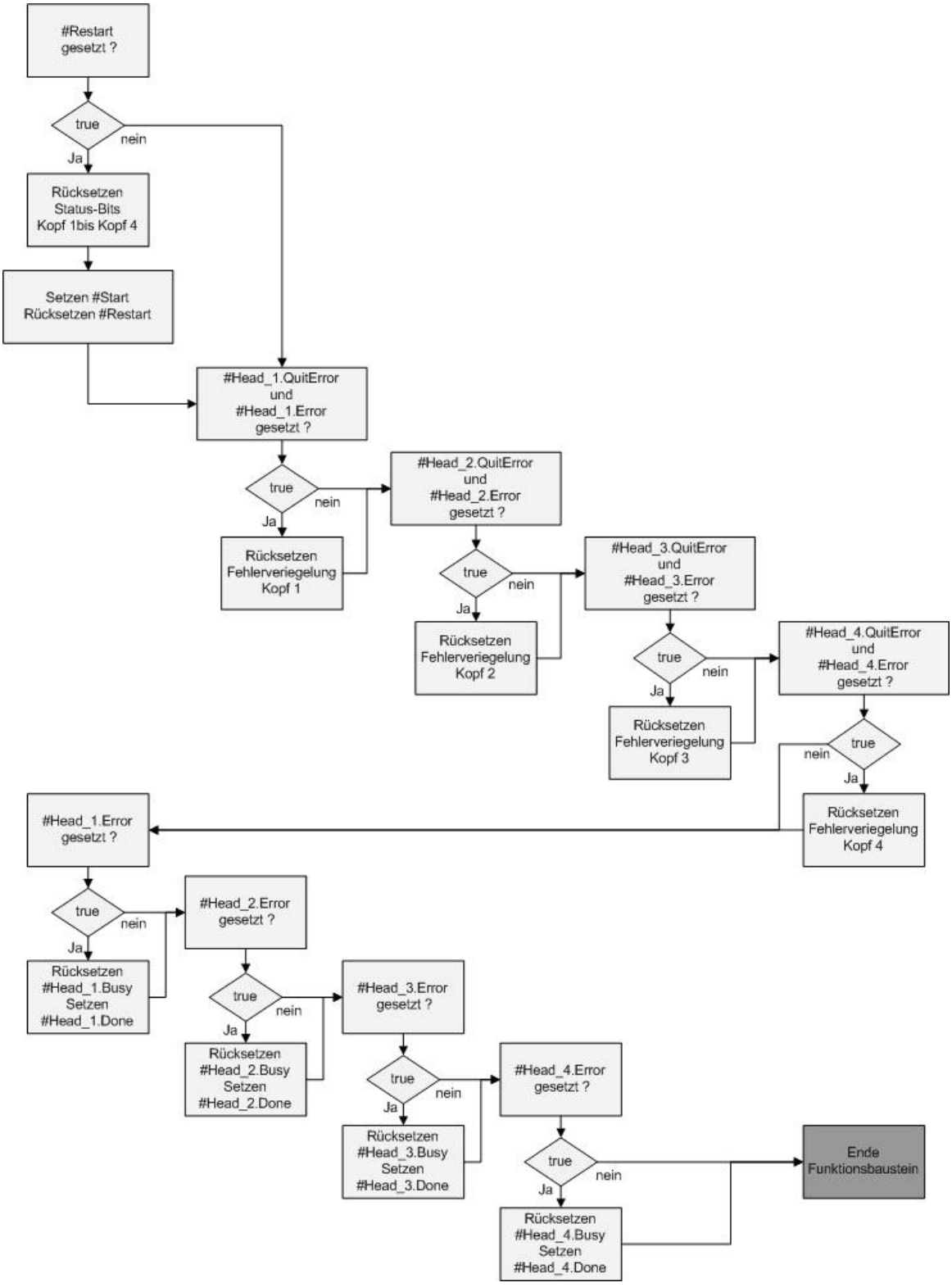
mand was sending to the IDENTControl in the cycle before and now the program jump to the Restart routine.

Afterwards the Toggle bit is set or reset. This is necessary to differentiate sequential identical commands.

In the next step different locking bits are set and the parameterization of the SFC 15 accomplished. The source data field of the SFC 15 is the outdata field of head 1 #Head_1.OutData. After the parametrisation the system function SFC 15 is called. With the help of the variable #Head1_exe you can differentiate the point of calling the SFC 15. There are two different points in the program where the function SFC 15 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 SFC 15 is called inside the initialisation routine the bit #Head1_exe is reset. But if the SFC 15 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.

After the setting of the bit #Head1_exe the program jumps to point aus1 into the network 19. At this point the program checked the execution of the SFC 15. With the help of #Head1_exe the return to point F155 is realised. At this point the bit #TransfToHead1 is reset, if no error by the execution of the SFC 15 arose. Finally the program jump to the Restart routine.

2.4.8 Procedure Restart Routine



Picture 16: Program flow chart 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 18. 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.

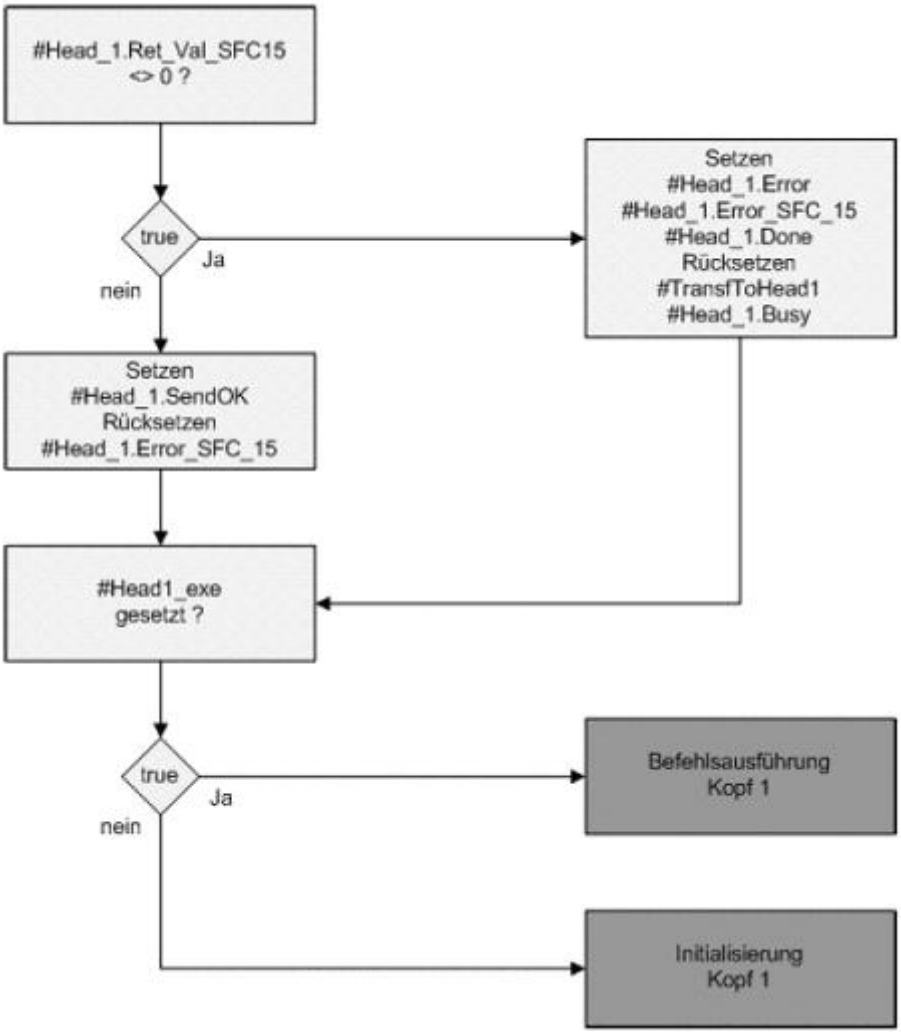
One part of the Restart routine is the acknowledgement of the error bits. This part goes through if the bit #Restart is not set at the beginning of the Restart routine. If the bit is not set the program directly jumps to the acknowledgement of the error bits. The acknowledgement is also passing through 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.

2.4.9 Procedure Analysis of Function SFC 15



Picture 17: Program flow chart Analysis SFC 15

The task of this program part is to analyse the execution of the SFC 15. The SFC 15 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 SFC 15 the program jumps to the analysis of the SFC 15. With the help of the analysis the program detect a failure in the execution of the SFC 15. In this case the out data field of the function block is not transferred to the IDENTControl.

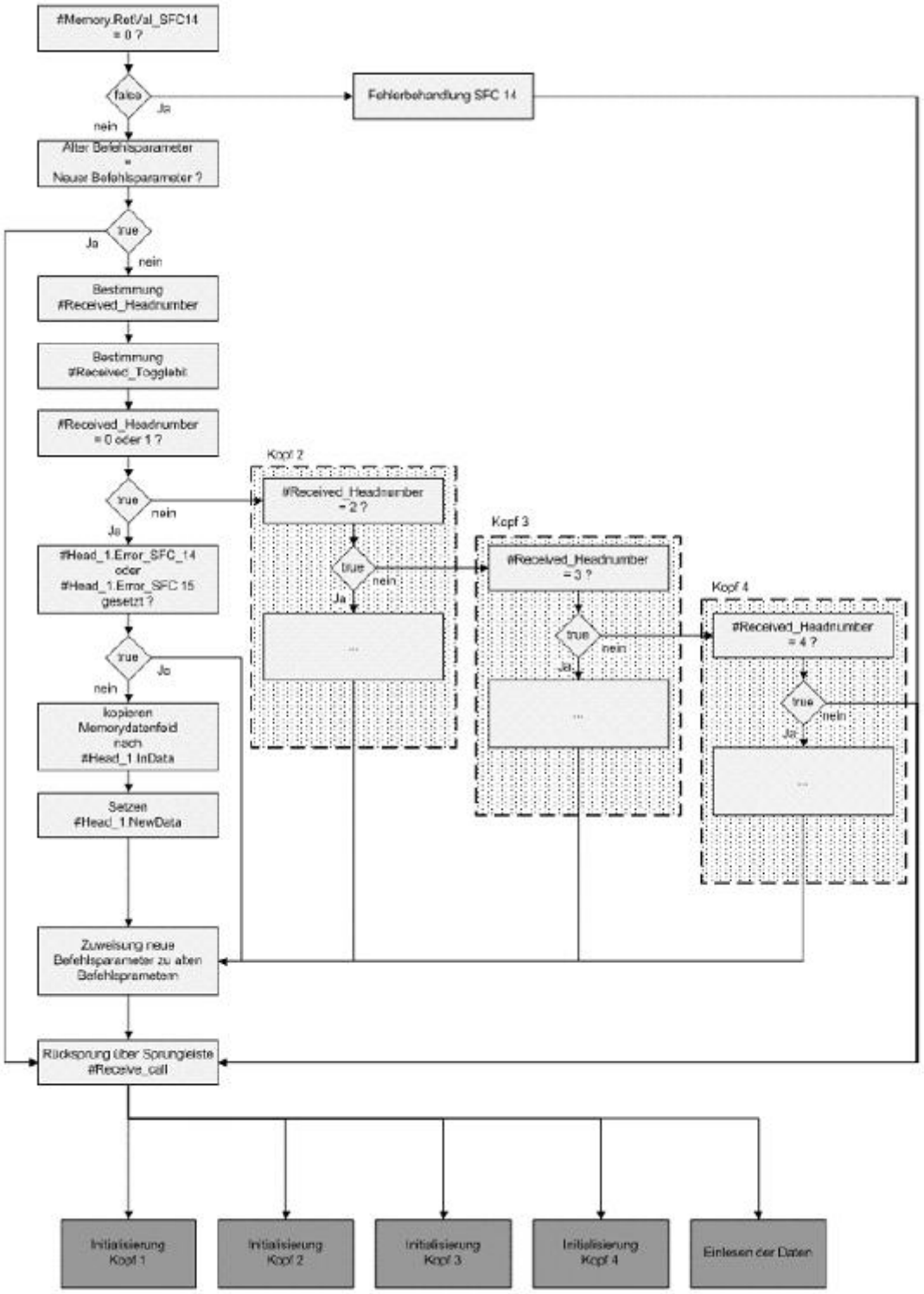
Following the analysis is exemplarily described for head 1. The analysis of the other heads is analogue.

To the beginning of the analysis the variable #Head_1.Ret_Val_SFC15 is compared to the value 0. This variable represents an error value of the execution of the SFC 15. If the value of the variable is not 0 and error is occurred. And the program jump to

point e151. At this point the error bits are set, which signalize an error in the execution of the SFC 15 on head 1.

If no error occurred the value of the variable is 0 and the bit #Head_1.SendOK is set. Then the program jumps to point spr1. At this point the return out of the analysis is realised with the help of the variable #Head1_exe. If the bit is set, the program jumps to point F155 to the command execution of head 1. Otherwise the program jumps to point F151 to the initialisation of head 1.

2.4.10 Execution of Analysis Function SFC 14



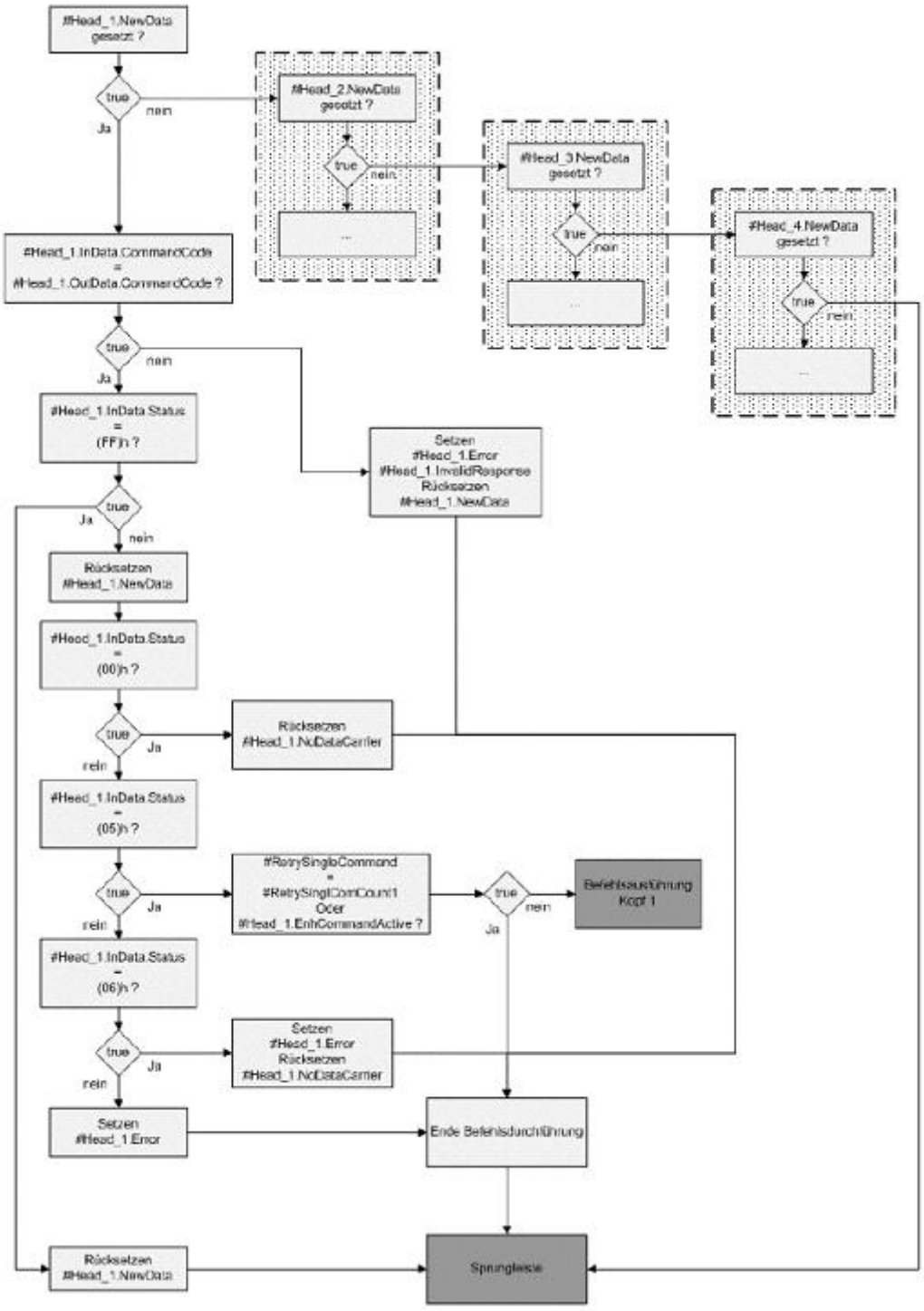
Picture 18: Program flow chart Analysis SFC 14

The task of this program part is to analyse the execution of the SFC 14. The SFC 14 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 SFC 14 in the different program parts the program jump to the analysis of

the SFC 14 into network 20. Firstly the execution of the SFC 14 is checked. If the value of the variable #Memory.ReVal_SFC14 is not 0, an error is occurred in the execution of the SFC 14. In this case the program jumps to point er14. At this point the program performs an error handling. If the execution of the SFC 14 is error free the program compare the new (import) command parameter to the old command parameter. If both parameters are not identical, new data import from the IDENTControl. If new data import from the IDENTControl, firstly the head number of the data isolated from the memory data field. The head number allocated to the variable #Received_Headnumber. In the next step the Togglebit identify and allocate to the variable #Received_Togglebit.

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. This data can now analyse by the program. Following the new import command parameter allocate to the old command parameter. The last step is the return to the program part, where the analysis was called.

2.4.11 Analysis of Input Data Fields



Picture 19: 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 IDENTContro 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.

3. Addendum

3.1 Listing of Parameter

3.1.1 Input Parameter (IN-Parameter)

- IDENT_Control_Address
  Address of the IDENTControl inside the Profibus network. The declaration refers to the I/O addresses of the IDENTControl in the Hardware configuration.
- Timeout
  Time slot for Timeout control
- RetrySingleCommand
  Number of maximal command repetitions.
- HeadXDataFixcode
  HeadXDataFixcode = 0 **à** access to memory area
  HeadXDataFixcode = 1 **à** access to Fixcode
- HeadXSingleEnhanced
  HeadXSingleEnhanced = 0 **à** execution of single command
  HeadXSingleEnhanced = 1 **à** 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.

3.1.2 Pass Parameter (IN-OUT-Parameter)

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

3.1.3 Static Parameter (STAT-Parameter)

- Byte_number
  Number of telegram length (i.e. Byte_number = 32 if „IN/Out 32 Byte“
- 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.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 and the Toggle bit.
  - Head_X.InData.Status
    Status value of the command execution

  - Head_X.InData.ExecutionCounter
    Execution 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 datatypes.
  - 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 **à** 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 **à** 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
- 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 **à** data read / write finish (command already active)
  Single command **à** command execution finished
- Head_X.Busy
  Command is in processing
- Head_X.Error_SFC_14
  Error by the execution of the SFC 14
- Head_X.Error_SFC_15
  Error by the execution of the SFC 15
- 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_SFC14
  Contain an error value by the execution of the SFC 14
- Head_X.RetVal_SFC15
  Contain an error value by the execution of the SFC 15
- 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.

3.2 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-	(??)h	#Head_1.SpecialCommand.Code	#Head1SpecialCommand = 1

Control-Command		#Head_1.SpecialCommand.Parameter	#IC_Command_on_Head1 = 1
QuitError Command	-	-	#QuitErrorHeadX
Quit Command	-	-	#HeadXQuit

3.3 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 Kbytes	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 Kbytes