Commissioning IDENTControl Compact
IC-KP2-2HB17-2V1D Profinet Protocol
with a Siemens S7 PLC

Index of contents

1. Adjustment of communication parameter with web interface

In the delivery status the IDENTControl do not have a Profinet-IO name. The name of the Profinet-IO partner can be assigned with the help of the device web interface. The IP-Address of the delivery status is 169.254.10.12.

With the option “Network” you are able to define the communication parameter and the Profinet-IO name.

The new parameterization of the device will adopt with “Save”. Afterwards the device starts automatically a restart. The new configuration will activated by turning the switch on the back side to position “0”.

Alternatively the Profinet-IO name can define with the help of the Hardware configuration tool of the PLC. In this case you choose following in the hardware configuration: PLC **à** Ethernet **à** Edit Ethernet Node.

In this window you can with “Browse” search for all connected Profinet-IO nodes. Afterwards you can execute a new parameterization for the IP-Address and the Profinet-IO name.

2. Installation GSDML-File

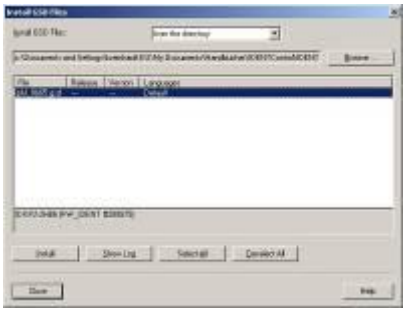
Before starting the first commissioning the identification systems IDENTControl you need to install the GSDML file. You can find the GSDML file you on the CD “Identification Systems” which is enclosed the product. Alternatively you can download the file on the Pepperl+Fuchs Group website.

www.pepperl-fuchs.com

(Product search **à** IC-KP2-2HB17-2V1D **à** 18XXXXX.zip)

To install the GSDML file you need to open the menu point “Options“ **à** ”Install GSD file” in the Simatic hardware configuration.

Afterwards you need to choose the GSD file out of the source list.



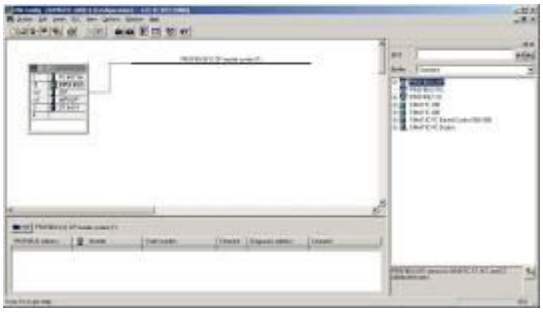
The GSDML file will be transferred into the hardware catalog with the menu point “Options” **à** “Update Catalog”

3. Installation User Program

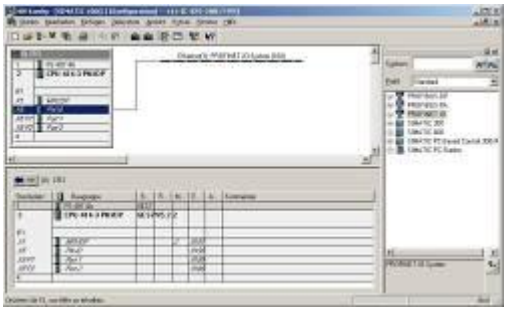
To install the user program you need to unpack the file “IC-KP2-2HB17.zip“. For this you need to select the menu item “File” **à** “Unpack” in the Simatic Manager. Afterwards you can open the file by marking it and click the “Open”-button. After defining the storage space and the successful installation you can see the user program in the Simatic Manager window.

4. Hardware configuration

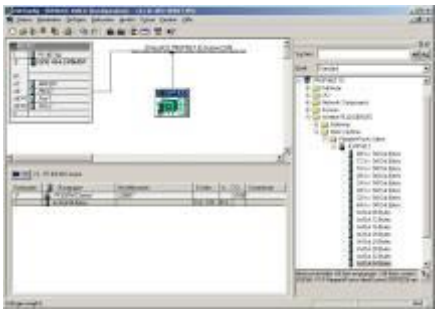
In the hardware configuration you have to parameterize the actual used modules of the PLC.



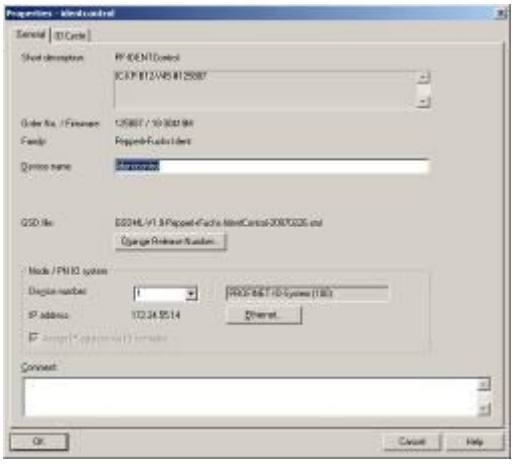
Afterwards you can implement the Profinet-IO master system by right click on the Profinet-IO port.



Than the connection of the IDENTControl to the Profinet-IO system follows. For commissioning the IDENTControl you need drag the symbol “IC-KP2-2HB6” from the hardware catalog and drop it on the Master system. Then you need to define the Profibus telegram length with the help of the communication modules (i.e. In/Out 64 Byte). You also need to assign a slot to the module.



Per double click on the symbol of the IDENTControl a window with the properties of the Profinet-IO node will open. Now you are able to reconfigure the name of the Profinet-IO node (here: identcontrol).

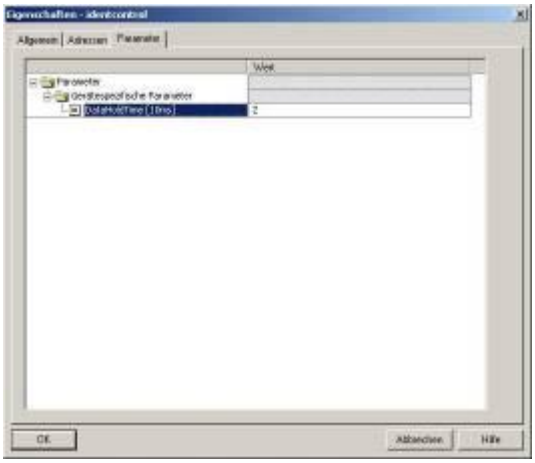


With the choose “Ethernet” you can reconfigure the IP-Address of the Profinet-IO node.

If you want to copy the function block into another application program it is useful also to copy the symbol table. This way you will be able to work with symbol information.

5. Adjustment of device parameters

You can open the device parameters by double-cliching the slot 0 of the I/O table. .



The device specific parameter “Data Hold Time” (DHT) defines the time in which the data in the data output cannot be overwritten. The value of the DHT should be the double time of the maximum cycle time of the PLC.

6. Function block „IDENTControl“

You can open the function block and the instance data block by executing the Call-command:

Call „IDENTControl“, „InstDB“(symbolic description)

Or

Call FB32, DB32

If you would like to implement more than one IDENTControl to the PLC you can generate the necessary instance data blocks with the Call-command.

Call FB32, DB33

Call FB32, DB34 and so one.

The following picture shows how the function is opened and the configuration of the variables.

```
CALL  "IDENTControl" , "InstDB"
 IC_INPUT_Address    :=W#16#200
 IC_OUTPUT_Address   :=W#16#200
 Length_IN           :=64
 Length_OUT          :=64
 Timeout             :=T#2S
 Head1DataFixcode    :="Head1DataFixcode"
 Head2DataFixcode    :="Head2DataFixcode"
 Head1SingleEnhanced:="Head1SingleEnhanced"
 Head2SingleEnhanced:="Head2SingleEnhanced"
 Head1SpecialCommand:="Head1SpecialCommand"
 Head2SpecialCommand:="Head2SpecialCommand"
 Head1Read           :="Head1Read"
 Head2Read           :="Head2Read"
 Head1Write          :="Head1Write"
 Head2Write          :="Head2Write"
 Head1Quit           :="Head1Quit"
 Head2Quit           :="Head2Quit"
 QuitErrorHead1      :="QuitErrorHead1"
 QuitErrorHead2      :="QuitErrorHead2"
 IC_Command_on_Head1:="IC_Command"
 Head1WordNum        :=15
 Head2WordNum        :=15
 Head1WordAddress    :=W#16#0
 Head2WordAddress    :=W#16#0
 Head1TagType        :=W#16#3231
 Head2TagType        :=W#16#3231
 Head1Done           :="Head1Done"
 Head2Done           :="Head2Done"
 Head1NoDataCarrier :="Head1NoDataCarrier"
 Head2NoDataCarrier :="Head2NoDataCarrier"
 Head1Error          :="Head1Error"
 Head2Error          :="Head2Error"
 Head1Busy           :="Head1Busy"
 Head2Busy           :="Head2Busy"
 Head1Status         :="Head1Status"
 Head2Status         :="Head2Status"
 Head1ReplyCounter   :="Head1ReplyCounter"
 Head2ReplyCounter   :="Head2ReplyCounter"
 InitFinish          :="InitFinish"
 SetRestart          :="SetRestart"
```

Name	Datentyp	Beschreibung
IC_INPUT_Address	WORD	Start address of the controller in the process area of the inputs (Input Address)
IC_OUTPUT_Address	WORD	Start address of the controller in the process area of the outputs (Output Address)
Length_IN	INT	Length of the Input telegram (length of the received Profibus telegram)

Name	Datentyp	Beschreibung
Length_OUT	INT	Length of the Output telegram (length of the sent Profibus telegram)
Timeout	TIME	Timer to control the response time of the system
RetrySingleCommand	INT	Number of repetitions of single commands if no tag was recognized
Head1DataFixcode	BOOL	grasp head 1 to 0:=Fixcode 1:=data area
Head2DataFixcode	BOOL	grasp head 2 to 0:=Fixcode 1:=data area
Head1SingleEnhanced	BOOL	Execution on head 1of 0:=Single 1:=Enhanced command
Head2SingleEnhanced	BOOL	Execution on head 2of 0:=Single 1:=Enhanced command
Head1SpecialCommand	BOOL	Execution of a special command on head 1 (positive edge); the command parameter needs to be defined before in the data structure Head_1.SpecialCommand; the received data is the data structure Head_1.InData
Head2SpecialCommand	BOOL	Execution of a special command on head 2 (positive edge); the command parameter needs to be defined before in the data structure Head_2.SpecialCommand; the received data are inside the data structure Head_2.InData
Head1Read	BOOL	Execution of a read command on head 1 (positive edge); define command parameter Head1WordNum and Head1Wordaddress; received data is in the data structure Head_1.InData
Head2Read	BOOL	Execution of a read command on head 2 (positive edge); define command parameter Head2WordNum and Head2Wordaddress; received data is in the data structure Head_2.InData
Head1Write	BOOL	Execution of a write command on head 1 (positive edge); define command parameter Head1WordNum and Head1Wordaddress; writable data needs to be defined in the data structure Head_1.OutData.DW1...15
Head2Write	BOOL	Execution of a write command on head 2 (positive edge); define command parameter Head2WordNum and Head2Wordaddress; writable data needs to be defined inside the data structure Head_2.OutData.DW1...15
Head1Quit	BOOL	Execution of the quit command on head 1 to abort the enhance command (positive edge)
Head2Quit	BOOL	Execution of the quit command on head 2 to abort the enhance command (positive edge)
QuitErrorHead1	BOOL	Execution of the error routine on head 1 (positive edge)
QuitErrorHead2	BOOL	Execution of the error routine on head 2 (positive edge)
IC_Command_Head1	BOOL	Execution of a special command of the controller (positive edge); command is send on channel 0; define command parameter inside data structure Head_1.SpecialCommand; received data are inside the data structure Head_1.InData
Head1WordNum	INT	Number of allocated data blocks on head 1
Head2WordNum	INT	Number of allocated data blocks on head 2
Head1WordAddress	WORD	Start address of the memory area of the tag allocated on head 1
Head2WordAddress	WORD	Start address of the memory area of the tag allocated on head 2
Head1TagType	WORD	Tag Type of head 1 (table of data carrier)
Head2TagType	WORD	Tag Type of head 2 (table of data carrier)
Head1Done	BOOL	New data exists (Enhanced) or command finished (Single) on head 1 (positive edge)
Head2Done	BOOL	New data exists (Enhanced) or command finished (Single) on head 2 (positive edge)
Head1NoDataCarrier	BOOL	No tag was in front of the head 1 during command execution
Head2NoDataCarrier	BOOL	No tag was in front of the head 2 during command execution
Head1Error	BOOL	Error occurred on head 1 (positive edge)
Head2Error	BOOL	Error occurred on head 2 (positive edge)
Head1Busy	BOOL	Command execution on head 1
Head2Busy	BOOL	Command execution on head 2
Head1Status	BYTE	Status value of channel 1
Head2Status	BYTE	Status value of channel 2
Head1ReplyCounter	BYTE	Value of the reply counter channel 1
Head2ReplyCounter	BYTE	Value of the reply counter channel 2
InitFinish	BOOL	Execution of controller initialization is finished (positive edge)
SetRestart	BOOL	Start of initialization (positive edge)

Example of the parameterization of the communication parameter:

In the hardware configuration the communication module “64In / 64 Out Bytes” is chosen. The process area of the Inputs (Input Address) starts at address 512 and finished at address 575 and has a length of 64 Bytes. The process area of the outputs (Output Address) starts at 512 finishes at address 575 and has a length of 64 Bytes. In this case you have the following parameterizations of the function block:

IC_INPUT_Address	:=W#16#200
IC_OUTPUT_Address	:=W#16#200
Length_IN	:=64
Length_OUT	:=64

Annotation:

If you choose a communication module for read-only mode (f.e. “64 In / 4 Out Bytes”) the output address (“IC_OUTPUT_Address”) has to be completely in the process output area. When using a CPU of the S7-300 line the process output area ranges from 0 to 256. The S7-400 line has a length of the process output area with a length of 0 to 512.

Please check the maximal telegram length of your CPU. The CPU S7-315-2DP has got a telegram length of 32 Bytes. The CPUs of the S7-400 line are able to communicate with a telegram length of 64 Bytes.

7. Error and Device Diagnostic

Most of the errors occur during the first implementation (initialization) of the function block. If the start of the initialization (positive edge of SetRestart) the bit InitFinish is not automatically set to true, an error has occurred during the execution of the initialization. The main reason for such an error is a difference in the parameterization of the hardware configuration and the communication parameters of the function block. This can be the parameters of the I/O area or a different parameterization telegram length. Another reason can be a wrong defined Tag Type (Head1(2)TagType). Another option of errors is the execution of commands. In this case the bit Head1(2)Error is set to true. Afterwards it is possible to make a diagnosis with the help of the listed error table. Reasons for such errors could be wrong defined command parameters (especially by executing Special Commands) or a wrong defined Tag Type. A wrong parameterized Tag Type is indicated with the status value 04hex inside the variable Head1(2) Status.

Name	Data Type	Description	Repair
Head1(2)Error	BOOL	Error on head 1(2)	
Head_1(2).InvalidResponse	BOOL	Send and received command telegram are not identical	Check of the Data Hold Time; Check the I/O configuration
Head_1(2).TimeoutOccured	BOOL	Timer to control the response time of the controller is run out. Slave does not answer in the defined time period.	Enlarge the value of the variable Timeout if you have a large cycle time.
Head_1(2).Error_SFC_14	BOOL	Error while reading data from the process area	Check of the variable Head_1 (2).Ret_Val_SFC14. Check of the parameterized I/O Address and telegram length.
Head_1(2).Error_SFC_15	BOOL	Error while writing data to the process area.	Check of the variable Head_1 (2).Ret_Val_SFC15. Check of the parameterized I/O Address and telegram length.

Head_1(2).Ret_Val_SFC14	WORD	Error value while executing the SFC14	W#16#8090 W#16#80B1 Check of the parameterized I/O Address and telegram length (more information inside the system help of the SFC14)
Head_1(2).Ret_Val_SFC15	INT	Error value while executing the SFC15 (convert the value in hex format)	W#16#8090 W#16#80B1 Check of the parameterized I/O Address and telegram length (more information inside the system help of the SFC15)
Head1(2) Status	BYTE	Status value of the last received data of channel1(2)	Check the table status values.
Memory.InData.Status	BYTE	Status value of the last received data of channel1(2)	Check the table status values.

The diagnosis data (Slave Diagnosis) of the IDENTControl will be read with the help of the system function SFC13”DPNRM_DG” out of the diagnosis address. The address will be automatically allocated during the slaves’ configuration. You can see the address by double clicking the IDENTControl symbol.

The length of the diagnosis data is 48 Byte. In the user program the diagnosis data will save inside the DB2. It is recommended to copy this diagnosis address into the application program. This way different device parameters (like software dates) can checked in an easily. Following you can find the diagnosis read from address 4092 (=0xFFC).

```
      U     "RD_SLAVE_DIAG"
      SPBN  DIAG

      CALL  "DPRD_DIAG"
       REQ     :="RD_SLAVE_DIAG"
       LADDR   :=W#16#FFC
       RET_VAL:=MW8
       RECORD  :=P#DB2.DBX0.0 BYTE 48
       BUSY    :=M4.0
//     R       M       4.0
      R     "RD_SLAVE_DIAG"
DIAG: NOP   0
```

8. Examples of command executions

Initialization: (with data carrier IPC03)

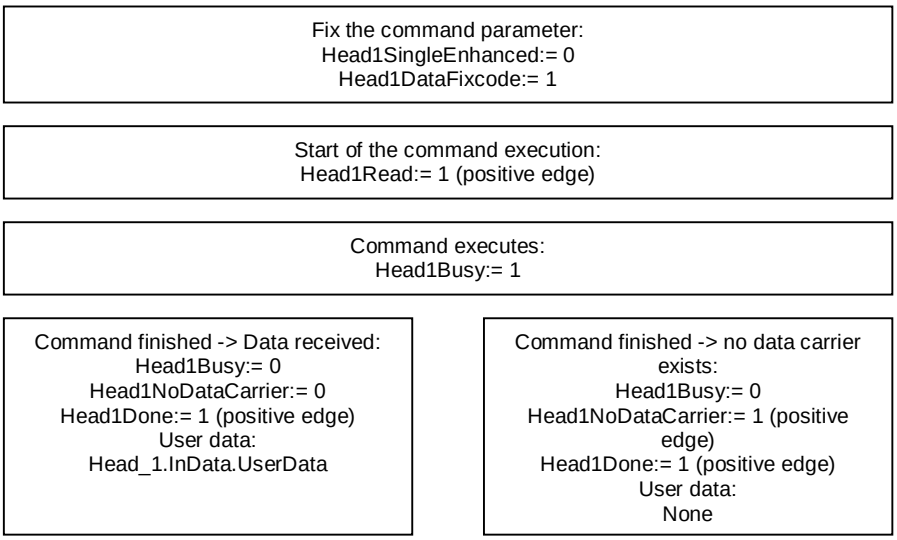
Fix of the data carrier (Tag Type):
Head1TagType:= W#16#3033 (IPC03-..)
Head2TagType:= W#16#3033 (IPC03-..)

Start of the Initialization:
SetRestart: = 1 (positive edge)

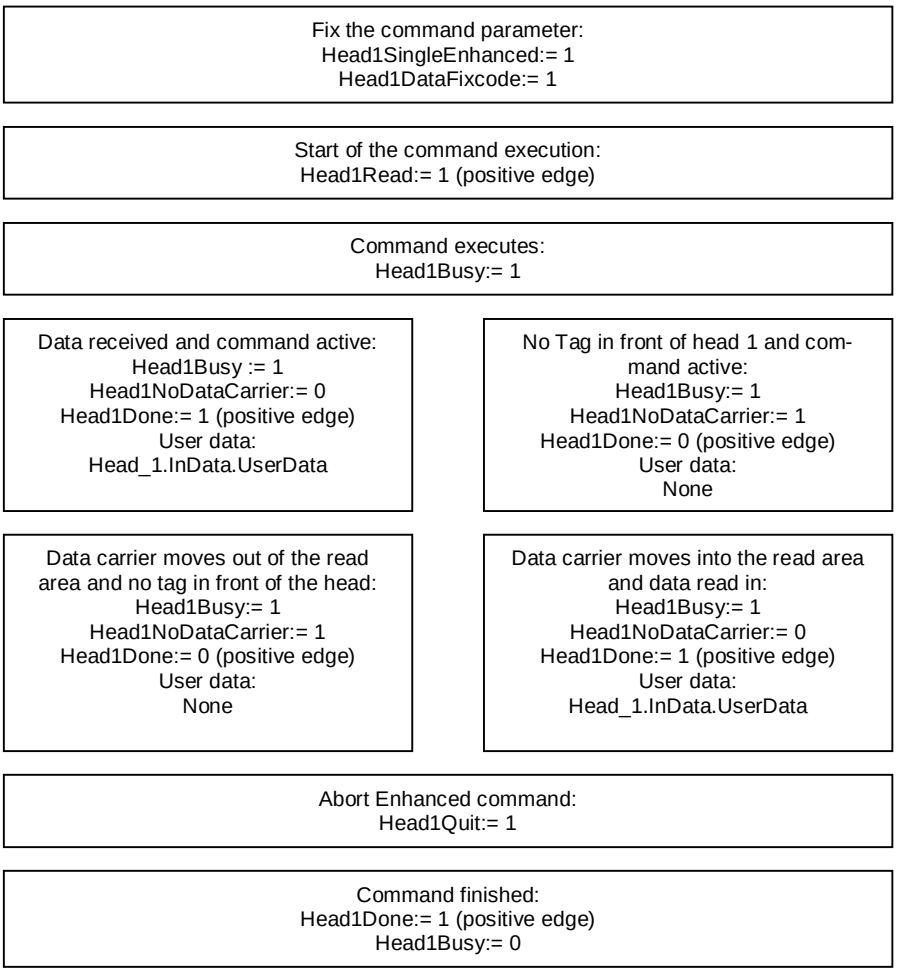
Initialization executes:
Head1Busy:= 1
Head2Busy:= 1

Initialization finished:
InitFinish: = 1 (positive edge)
Head1Done:= 1
Head2Done:= 1

Single Read Fixcode: (head 1)



Enhanced Read Fixcode: (head 1)



Single Read Words: (head 1; 15 data blocks starting at address 0)

Fix the command parameter:
Head1WordNum:= 15 (number of user data blocks)
Head1WordAddress:= 0 (memory address)
Head1SingleEnhanced:= 0
Head1DataFixcode:= 0

Start of the command execution:
Head1Read:= 1 (positive edge)

Command executes:
Head1Busy:= 1

Command finished -> Data received:
Head1Busy:= 0
Head1NoDataCarrier:= 0
Head1Done:= 1 (positive edge)
User data:
Head_1.InData.UserData

Command finished -> no data carrier exists:
Head1Busy:= 0
Head1NoDataCarrier:= 1
Head1Done:= 1 (positive edge)
User data:
None

Enhanced Read Words: (head 1; 15 data blocks starting at address 0)

Fix the command parameter:
Head1WordNum:= 15 (number of user data blocks)
Head1WordAddress:= 0 (memory address)
Head1SingleEnhanced:= 1
Head1DataFixcode:= 0

Start of the command execution:
Head1Read:= 1 (positive edge)

Command executes:
Head1Busy:= 1

Data received and command active:
Head1Busy:= 1
Head1NoDataCarrier:= 0
Head1Done:= 1 (positive edge)
User data:
Head_1.InData.UserData

No Tag in front of head 1 and command active:
Head1Busy:= 1
Head1NoDataCarrier:= 1
Head1Done:= 0 (positive edge)
User data:
None

Data carrier moves out of the read area and no tag in front of the head:
Head1Busy:= 1
Head1NoDataCarrier:= 1
Head1Done:= 0 (positive edge)
User data:
None

Data carrier moves into the read area and data read in:
Head1Busy:= 1
Head1NoDataCarrier:= 0
Head1Done:= 1 (positive edge)
User data:
Head_1.InData.UserData

Abort Enhanced command:
Head1Quit:= 1

Command finished:
Head1Done:= 1 (positive edge)
Head1Busy:= 0

Single Write Words: (head 1; 15 data blocks starting at address 0)

Fix the command parameter:
Head1WordNum:= 15 (number of user data blocks)
Head1WordAddress:= 0 (memory address)
Head1SingleEnhanced:= 0
Head1DataFixcode:= 0

Define user data inside out data field:
Head_1.OutData.UserData := XXXX

Start of the command execution:
Head1Write:= 1 (positive edge)

Command executes:
Head1Busy:= 1

Command finished -> Data written
Head1Busy:= 0
Head1NoDataCarrier:= 0
Head1Done:= 1 (positive edge)

Command finished -> no data carrier exists:
Head1Busy:= 0
Head1NoDataCarrier:= 1
Head1Done:= 1 (positive edge)

Enhanced Write Words: (head 1; 15 data blocks starting at address 0)

Fix the command parameter:
Head1WordNum:= 15 (number of user data blocks)
Head1WordAddress:= 0 (memory address)
Head1SingleEnhanced:= 1
Head1DataFixcode:= 0

Define user data inside out data field:
Head_1.OutData.UserData:= XXXX

Start of the command execution:
Head1Write:= 1 (positive edge)

Command executes:
Head1Busy:= 1

Data written and command active:
Head1Busy:= 1
Head1NoDataCarrier:= 0
Head1Done:= 1

No Tag in front of head 1 and command active:
Head1Busy:= 1
Head1NoDataCarrier:= 1
Head1Done:= 0

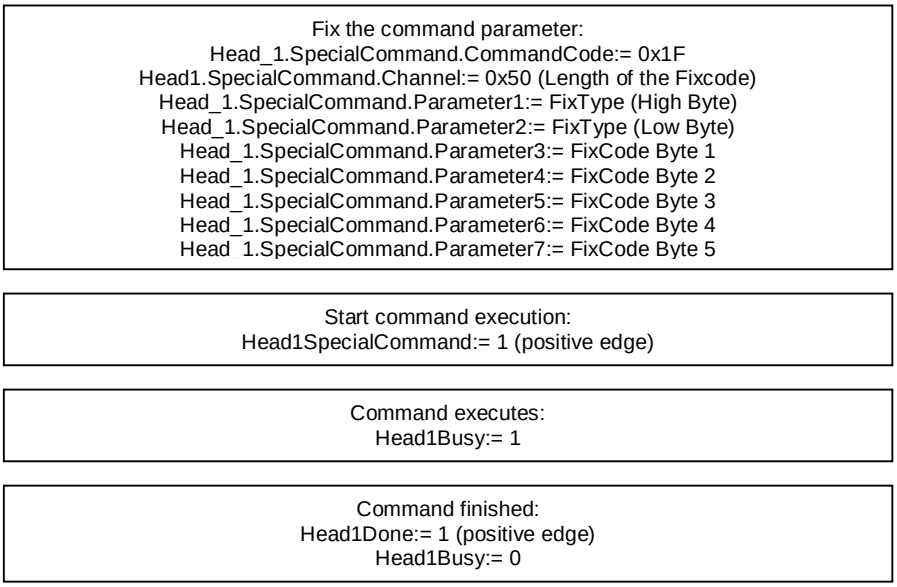
Data carrier moves out of the read area and no tag in front of the head:
Head1Busy:= 1
Head1NoDataCarrier:= 1
Head1Done:= 0 (positive edge)

Data carrier moves into the read area and data written:
Head1Busy:= 1
Head1NoDataCarrier:= 0
Head1Done:= 1 (positive edge)

Abort Enhanced command:
Head1Quit:= 1

Command finished:
Head1Done:= 1 (positive edge)
Head1Busy:= 0

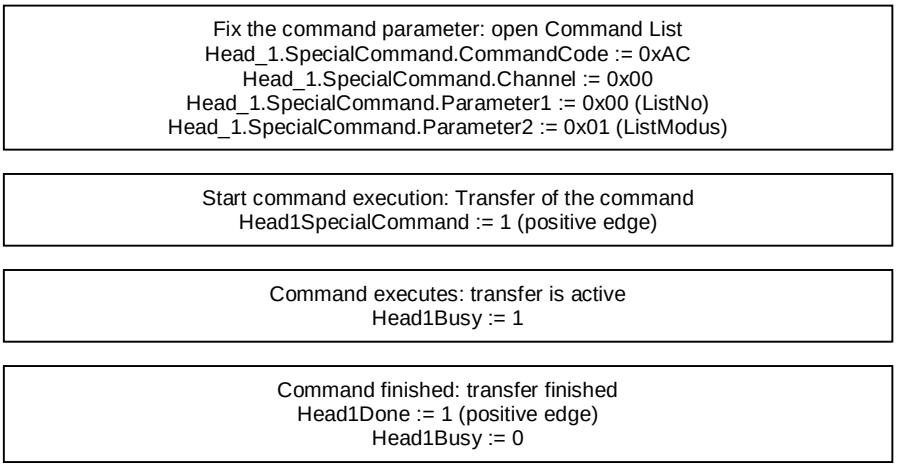
Single Write Fixcode: (head 1; IPC11)



Command List (Prefetch): (head 1)

With the help of the command list it is possible to execute different commands successively. At first you need to open the command list. Afterwards you need to transfer the commands which have to be performed and then close the command list. The list will be executed when you activate the list in Single or Enhanced mode. The command list will be stored volatile. The opening, closing and the activation as well as the transfer of the performed commands will be executed with the help of the SpecialCommand.

Open the Command List:



After the opening of the command list you need to transfer the performed commands with the help of the SpecialCommand to the controller. Afterwards first command Enhanced Read Fixcode and second command Enhanced Read Words are parameterized.

Command 1: Enhanced Read Fixcode head 1

Fix the command parameter: Enhance Read Fixcode Head_1.SpecialCommand.CommandCode := 0x01 Head_1.SpecialCommand.Channel := 0x00
Start command execution: Transfer of the command Head1SpecialCommand := 1 (positive edge)
Command executes: transfer is active Head1Busy := 1
Command finished: transfer finished Head1Done := 1 (positive edge) Head1Busy := 0

Command 2: Enhanced Read Words

Fix the command parameter: Enhanced Read Words Head_1.SpecialCommand.CommandCode := 0x19 Head_1.SpecialCommand.Channel := 0xF0 (number of data blocks) Head_1.SpecialCommand.Parameter1 := 0x00 (WordAddress High Byte) Head_1.SpecialCommand.Parameter2 := 0x00 (WordAddress Low Byte)
Start command execution: Transfer of the command Head1SpecialCommand := 1 (positive edge)
Command executes: transfer is active Head1Busy := 1
Command finished: transfer finished Head1Done := 1 (positive edge) Head1Busy := 0

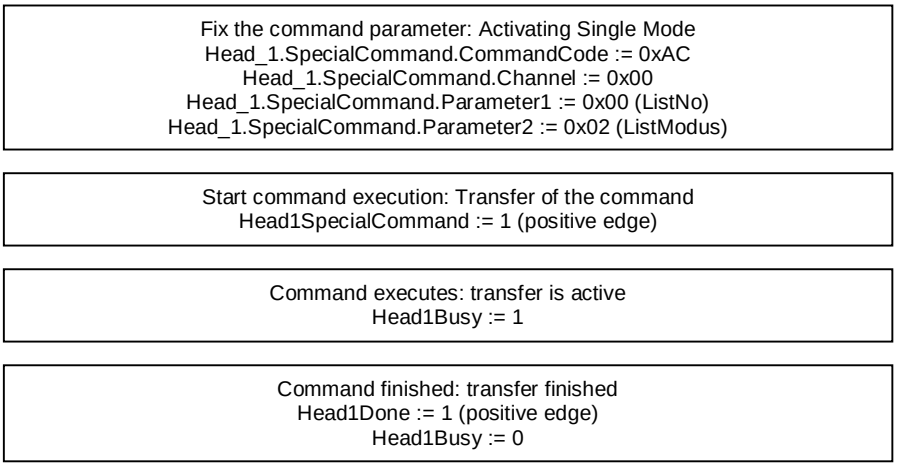
The number of the performed commands inside the command list is 10. The parameterization of the command list is finished by closing of the command list.

Closing command list:

Fix the command parameter: closing command list Head_1.SpecialCommand.CommandCode := 0xAC Head_1.SpecialCommand.Channel := 0x00 Head_1.SpecialCommand.Parameter1 := 0x00 (ListNo) Head_1.SpecialCommand.Parameter2 := 0x00 (ListModus)
Start command execution: Transfer of the command Head1SpecialCommand := 1 (positive edge)
Command executes: transfer is active Head1Busy := 1
Command finished: transfer finished Head1Done := 1 (positive edge) Head1Busy := 0

The execution of the command list will be started by activating in Single or Enhanced Mode.

Activating with Single Mode:



After the activation of the command list the commands will be executed. The read in data will be available inside the data field Head_1.InData.UserData. If a tag is in the reading range the status (Head1Status) of every response telegram has the value 0x00. To verify the different responses you need to check the ReplyCounter (Head1ReplyCounter). By receiving a new telegram the value of this variable changes based on the previous telegram. After finishing the cycle of the command list you get a response with the Status (Head1Status) value of 0x0F. If you activate the command list in Single mode, the commands will be executed once. After the last telegram (Head1Status = 0x0F) the execution is automatically stopped. By activating in Enhanced Mode the execution of the command list starts again with the first command after receiving the last telegram.

9. Table data carrier

Name	Tag Type	Command	Fixcode	Data	Word Address	frequency
IPC02	W#16#3032	Read Fixcode	5 Byte	-	-	125kHz
IPC03	W#16#3033	Read Fixcode Read Words Write Words	4 Byte	116 Byte	0000…001C	125kHz
IPC11	W#16#3131	Read Fixcode Write Fixcode	5 Byte	-	-	125kHz
IPC12	W#16#3132	Read Fixcode Read Words Write Words	4 Byte	8192 Byte	0000…07FF	125kHz
IDC	W#16#3530	Read Words Write Words Read Fixcode Write Fixcode Read Special Fixcode Write Special Fixcode	7 Byte Fixcode 6 Byte Special Fixcode	128 Byte	0000…001F	250kHz
ICC	W#16#3532	Read Fixcode	7 Byte	-	-	250kHz
IQC21	W#16#3231	Read Fixcode Read Words Write Words	8 Byte	112 Byte	0000…001B	13,56MHz
IQC22	W#16#3232	Read Fixcode Read Words Write Words	8 Byte	256 Byte	0000…003F	13,56MHz
IQC23	W#16#3233	Read Fixcode Read Words Write Words	8 Byte	224 Byte	0000…0037	13,56MHz
IQC24	W#16#3234	Read Fixcode Read Words Write Words	8 Byte	928 Byte	0000…00E7	13,56MHz
IQC31	W#16#3331	Read Fixcode	8 Byte	32 Byte	0000…0007	13,56MHz

		Read Words Write Words				
IQC33	W#16#3333	Read Fixcode Read Words Write Words	8 Byte	2000 Byte	0000…00F9	13,56MHz
IQC35	W#16#3335	Read Fixcode Read Words Write Words	8 Byte	256 Byte	0000…003F	13,56MHz
MVC	W#16#3630	Read Fixcode Read Words Write Words	8 Byte	7552 Byte	0000…075F	2,45GHz

10. Table Status values

Head1(2)Status	Meaning	Repair
0x00	Command executed without errors	None; next command can be sent to this channel
0xFF	Command in process	Command execute from the head in progress; a command can be sent to another channel (not to the same)
0x01	Battery status is low (only MVC Tags)	Data will be sent in the same telegram; change the battery or the whole tag
0x04	Incorrect or incomplete command or parameter not in valid range	Check the command parameters and the defined Tag Type (IQC33 has got even numbered WordNum); check the installation of the head (is the head grounded; shielded read head cable)
0x05	No data carrier in detection range	Check the distance between tag and head; check the installation of the head (is the head grounded; shielded read head cable)
0x06	Hardware error; no head is connected to this channel; head is defective	Check the cable of the head (shielded cable named V1-G-XM-PUR ABG-V1-W); check the LED of the head (switched off: head damaged; blinking: execute Initialization with correct Tag Type; constant: head is OK)
0x07	Internal device error	Internal memory overflow (reduce the Data Hold Time)
0x09	parameterized data carrier type does not match to the connected read head	Check the parameterized Tag Type
0x0F	Last telegram of command list	None;
0x20	Reset was executed	Error sent by the bus interface; signals that the controller is ready for command executions
0x40	Incorrect or incomplete command or parameter not in valid range	Error sent by the bus interface; Check the command parameters and the defined Tag Type (IQC33 has got even numbered WordNum); check the installation of the head (is the head grounded; shielded read head cable)
0x60	Hardware error; no head is connected to this channel; head is defective	Error sent by the bus interface; Check the cable of the head (shielded cable named V1-G-XM-PUR ABG-V1-W); check the LED of the head (switched off: head damaged; blinking: execute Initialization with correct Tag Type; constant: head is OK)
0x70	Internal device error	Internal memory overflow (reduce the Data Hold Time)

11. Table version information

Version	Date	Change Function Block	Change Documentation
2.0	18.11. 2008	- Change from 4-Channel version to 2-Channel Version - Replacement of the IN-Variable “IDENTControlAddress” with the IN-Variables INPUT/OUTPUT_Address and INPUT/OUTPUT_Length - Output data length of 4 Bytes possible by implementation of SFC81 - Entering the OUT-Variables Done; NoDataCarrier; Busy and Error	- Initial edition
2.1	28.02. 2009	- Connecting variable Memory.Error_SFC_14 with variable Head1(2)Error - Change of the check of the variables Head_1(2).ExistTC and Head_1(2).NotExist by parameterisation of the commands -> Triggermode - Implementation of the Status check of value 0x0F -> last telegram of the command list - Change of the identity check of the Input and Output data field -> by execution of the command list the Input and Output datafield are not identical - Reset of the variable Head_1(2).QuitError by execution of the Reset and the Quit-Error routine. - Implementation of new symbolic names of the user data fields of	- Implement new picture with the overview of the variables of the FB (page 5) - Implement new variables in the table with the functionality (page 5-6) - Information about the different telegram length of the CPUs lines (page 7) - Change of the name of the status variable (page 7-8)

		the Input and Output data fields - >Head_1(2).InData/OutData.UserData - Implementation of a check of the value of the variable Head1(2)WordNum -> if value higher than 15 the variable Head1(2)Error will be set - Implementation of the Outdata variable Head1(2)Status and Head1(2)ReplyCounter	
2.2	24.04. 2009	- Text library added for classification of the Status values	- none