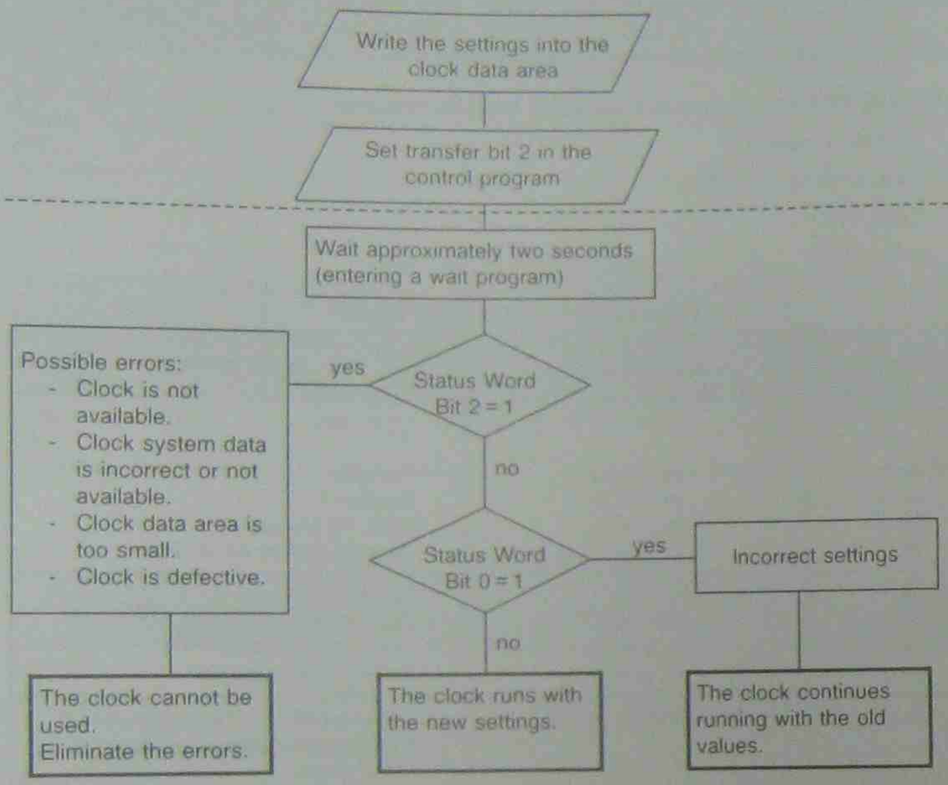


* The lower portion of the flowchart has only a diagnostic function. There is nothing you must perform. You can also implement the upper portion of the flowchart using the FORCE VAR programmer function (programmable controller in the RUN mode) or using the FORCE function (the programmable controller is in the STOP mode and bit 4 = 1 in the status word).

Figure 13-7. Flowchart - Transferring Time and Date Settings to the Clock

If you do not want a value (for example the minutes) in the settings to be transferred, enter the value for relevant byte as either 255_D or FF_H. When you set the clock, the old value present in the clock is retained.

Incorrect settings are displayed by a set bit 0 in the status word. The clock continues to run with the old values.

In a similar manner, you can program new settings for the time prompt function and the operating hours counter. However, the settings are located in other data words in the clock data area. See section 13.4. You must set the respective bit to 1 in the status word so that the clock can accept the new settings. See section 13.5.

13.7 Programming the Integral Real-Time Clock in the User Program

The programming of the clock in the user program should be performed only by users with extensive knowledge of the system. For all other users, use of DB1 is recommended (see sections 13.2 and 13.3).

The following section provides you with information on how to access the clock through the user program.

13.7.1 Reading and Setting the Clock

Example: Program for setting the time and date

Transfer of the settings for the time and date is triggered by input I 32.0. Before you set input I 32.0 (see OB1), you must transfer these settings to flag bytes FY120 to FY127. Values that you do not want to change must be preset with "FF_H".
You can define the clock mode with input I 33.0 (1 = 12-hour mode). Input I 32.1 is the AM/PM bit that you use for setting the 12-hour mode.

The clock data area is in DB2 beginning with DW0, and the status word is FW10.

OB1 STL	Explanation
: : : : : :A I 32.0 :S F 20.0 :JU FB 10 NAME :UHR-STEL WDAY : FY 121 DAY : FY 122 MON : FY 123 YEAR : FY 124 HOUR : FY 125 AMPM : I 32.1 MIN : FY 126 SEC : FY 127 ERR : F 12.1 MODE : I 33.0 :BE	========== SETTING THE TIME AND DATE ========== FIRST TRANSFER TIME AND DATE VALUES INTO FB120 TO FB127. CLOCK SETTING TRIGGERED BY SETTING F 20.0 (RESET IN FB10) (SETTING THE TIME AND DATE) WEEKDAY DAY MONTH YEAR HOUR AMPM-BIT (ONLY IMPORTANT IN 12-HOUR MODE) MINUTES SECONDS ERROR BIT 12 HOUR-MODE: I 33.0 = 1

13

Each parameter block begins with a block ID followed by a colon. This colon must be followed by at least one filler (such as a blank space or a comma). A semicolon must be at the end of each parameter block with at least one filler between the semicolon and the next block ID.

The following parameter blocks are used for the S5-90U (see Table 9-1).

Table 9-1. Parameter Blocks and Their IDs (S5-90U)

Block ID	Explanation/Default Setting
'DB1 ';	Start ID
'OBI: ';	**Onboard interrupt**: Parameter block for enabling or disabling interrupt input / **disabled** (see chapter 9)
'OBC: ';	**Onboard counter**: Parameter block for enabling or disabling counter input / **disabled** (see chapter 9)
'SLI: ';	**SINEC L1**: Parameter block for SINEC L1 configuration / (see chapter 14)
'ERT: ';	**Error return**: Address for parameter error code / no default setting
'END ';	**End** Block ID for DB1

The following parameter blocks are used for the S5-95U (see Table 9-2).

Table 9-2. Parameter Blocks and Their IDs (S5-95U)

Block ID	Explanation/Default Setting
'DB1 ';	Start ID
'OBA: ';	**Onboard analog inputs**: Parameter block for enabling or disabling analog inputs / **disabled** (see chapter 12)
'OBI: ';	**Onboard interrupt**: Parameter block for enabling or disabling interrupt inputs / **disabled** (see chapter 10)
'OBC: ';	**Onboard counter**: Parameter block for enabling or disabling counter inputs / **disabled** (see chapter 11)
'SLI: ';	**SINEC L1**: Parameter block for SINEC L1 configuration / (see chapter 14)
'SDP: ';	**System-dependent parameter**: Parameter block for system specifications / - 128 internal counters are processed - no external I/O bus is available (see section 9.1.8)
'TFB: ';	**Timer function blocks**: Parameter block for time-controlled program processing: OB13 is called up every 100 ms. (see chapter 7)
'CLP: ';	**Clock-Parameters**: Parameter block for integral time clock **no** clock function activated (see chapter 13)
'ERT: ';	**Error return**: Address for parameter error code / no default setting (see section 9.1.2)
'END ';	**End** block ID for DB1

The sequence of the parameters in DB1 is not fixed. A semicolon must be at the end of each parameter block with at least one filler between the semicolon and the next block ID.

The structure of the following parameter blocks is described here in detail.

- ERT: (Error code position)
- SDP: (System specifications)

The parameter blocks that are not discussed here are explained in other chapters.

9.1.2 Setting the Address for the Parameter Error Code in DB1
(An example of how to set the parameters correctly)

We recommend that you use this example when you start setting your parameters. The following two reasons explain why.

- Parameter block "ERT:" is the only block with no default parameters in DB1. You must therefore enter all the parameters. We will explain the rules for assigning parameters step-by-step, so that you can learn the rules quickly.
- The correctly input "ERT:" parameter block makes it easy for you to correct parameter setting errors; therefore, you should complete this block in DB1 before you change or add other parameters.

The error parameter block is only important during the start-up phase. You should erase it during "normal" operation because it takes up a lot of memory space.
To help find parameter errors more easily and to help correct them, you can ask the programmable controller to output error messages in a coded form. All you have to do is to tell the programmable controller where it should store the error code. Make this input in parameter block "ERT:" of DB1.

You can store error codes in the following words.

- Flag words
- Data words in a data block

The entire error code consists of 20 flag bytes or 10 data words. You only need to indicate the start address for the error code in parameter block "ERT:".

How to Proceed:

- Perform an overall reset on the programmable controller.
- Display DB1 on the programmer.
- Position the cursor on the E of the "END" ID at the end of the default DB1.
- Enter the characters that are highlighted in Figure 9-2.

DB1	Explanation
0: KS ='DB1 OBA: AI 0 ; OBI: '	
12: KS =' ; OBC: CAP N CBP '	
24: KS ='N ;#SLI: SLN 1 SF '	
36: KS ='DB2 DW0 EF DB3 DW0 '	
48: KS =' KBE MB100 KBS MB1'	
60: KS ='01 PGN 1 ;# SDP: N'	
72: KS ='T 128 PBUS N ; TFB: OB13'	
84: KS =' 100 ; #CLP: STW MW10'	
96: KS ='2 CLK DB5 DW0 '	
108: KS =' SET 3 01.10.91 12:00:'	
120: KS ='00 OHS 000000:00:00 '	
132: KS =' TIS 3 01.10. 12:00:00 '	
144: KS =' STP Y SAV Y CF 00 '	
156: KS =' ; #ERT: ERR MW1 ; END ';	The parameter error code is stored in flag word MW1 after start-up.

Figure 9-2. Inputting the Address for the Parameter Error Code

- Use the following check list to make sure your entries are correct.
  - Is the block ID "ERT:" terminated by a colon?
  - Is at least 1 filler (a blank space in Figure 9-2) added after the colon?
  - Is the parameter name (ERR) entered correctly?
  - Does at least 1 filler (a blank space) follow the parameter name?
  - Is the argument (MW1) entered correctly?
  - Does at least 1 filler (a blank space) follow the argument?
  - Does a semicolon (;) indicate the block end?
  - Does the end ID "END" followed by a space conclude DB1?

- Transfer the changed DB1 to the programmable controller.

- Now you can switch the programmable controller from STOP to RUN. The programmable controller now accepts the changed DB1.

If you did not store the parameter block "ERT:" in DB1, you can localize the error in the ISTACK if there was an incorrect parameter setting. However, you will not know what type of error is present. The same thing applies if you made an error when you input the parameter block "ERT:".

9.1.3 How to Assign Parameters in DB1

As discussed in section 9.1.2, you use the following steps to change or expand the preset values of DB1:

- Display the default DB1, with its parameter block "ERT:" on the programmer.
- Position the cursor on the desired parameter block.
- Change or expand the parameters.
  (for an explanation and possible parameter values see section 9.1.7)
- Transfer the changed DB1 to the programmable controller.
- Switch the programmable controller from STOP to RUN.

Changed DB1 parameters are accepted.

> **Note**
>
> If the programmable controller recognizes a parameter error in DB1, then it remains in the "STOP" mode even after it has been switched from STOP to RUN (red LED lights up).

9.1.4 Rules for Setting Parameters in DB1

DB1 consists of the folowing:

- A start block ID DB1
- One or more parameter blocks ..... e.g.: OBC: CAP 500

  A parameter block consists of:
  - A block ID ..... e.g.: OBC:
  - One or more parameters ..... e.g.: CAP 500

    A parameter consists of:
    - A parameter name ..... e.g.: CAP
    - One or more arguments ..... e.g.: 500
  - A block end ID ..... : ; (Semicolon)
- A block end symbol ..... : END

In the following section are all of the rules that have to be followed if you want to change or expand entire parameter blocks. Follow these steps or the programmable controller will not understand what you have entered.

1. Start ID "DB1"
   DB1 must begin with the start ID "DB1". Do not separate the three characters from each other. After the start ID, there must be at least one filler. The permitted fillers are: a blank space or a comma.

2. The start ID and filler are followed by the block ID for the parameter block. The sequence of the parameter blocks in DB1 is random. The block ID identifies a block and its corresponding parameter. The block ID "SL1", for example, stands for the SINEC-L1 parameter. You must enter a colon immediately after the block ID. If the colon is missing, then the programmable controller skips this block and displays an error message. You must add at least one filler after the colon of a block ID.

3. The parameter name comes next. Parameter names are names for single parameters within a parameter block. Within a block, the first four characters of a parameter name must be different from each other. After the parameter name, you must add at least one filler.

4. At least one argument is attached to each parameter name. An argument is either a number or a STEP 5 operand that you must enter. If several arguments belong to a parameter name, then every argument must be followed by at least one filler (even the last one).
5. Use a semicolon (;) to identify a block end. After the semicolon, you must enter at least one filler. Leaving out the semicolon leads to misinterpretation in the programmable controller.

6. After the semicolon, additional parameter blocks can follow. (Use steps 2 through 5 to create additional parameter blocks.)

7. After the end of the last parameter block, you must enter the end ID "END". This identifies the end of DB1. If you forget to enter an end ID, this leads to errors in the programmable controller.

Steps 1 through 7 present the minimal requirements for setting the parameters. Beyond that, there are additional rules that make it easier for you to assign parameters.

For example:

- You have the ability to add comments.
- You can expand the mnemonics used as parameter names by using plain text.

Comments can be added anywhere a filler is allowed. The comment symbol is the pound (#) sign. The comment symbol must be placed at the beginning and at the end of your comment. The text between two comment symbols may not contain an additional #.

Example: #Comment# . At least one filler must follow the # sign.

In order to make it easier to read parameter names, you can add as many characters as you wish if you add an underscore (_) after the abbreviated parameter name.
Example: SF becomes SF_SENDMAILBOX .
At the end of the input, you must add at least one filler.

There is a rule of thumb that will help you check DB1. You should include at least one filler in the following instances.

- After the start ID
- Before and after the block ID, parameter name, argument, and semicolon

9.1.5 How to Recognize and Correct Parameter Errors

Should an error occur while assigning parameters and the programmable controller does not go to the "RUN" mode, you have two possibilities for recognizing errors.

- By using a parameter error code
- By using the analysis function "ISTACK"

Both possibilities are described below.

Scanning the Parameter Error Code

If you have entered a start address for the parameter error code in parameter block "ERT:" of DB1 (see section 9.1.2), then you can retrieve the cause of the error, and the error location information at this address.

The entire error code occupies 10 data words or 20 flag bytes. In the following examples and tables, we assume that the error code is stored in a data block starting with data word 0. The error code occupies DW0 through DW9. In the "Flag" operand area, this corresponds to FW0 through FW19.

Example: You entered the start address DB3 DW0 in parameter block "ERT:". The parameters set in DB1 have already been transferred to the programmable controller. Then you continue to set parameters in DB1. While attempting to transfer the changed DB1 parameters to the programmable controller, you find out that the programmable controller remains in the "STOP" mode. You suspect that the reason for this is a parameter error. To find the error, display DB3 on the programmer. The entire contents of DB3 appear on the screen. DW0 through DW9 contain the code for the parameter error. In the following figure, you see how your screen could look. Below the screen display is a complete list of parameter error codes and their meanings.

```
 0:        KH=          0 61 0 3
 1:        KH=          0 0 0 0
 2:        KH=          0 0 0 0
 3:        KH=          0 0 0 0
 4:        KH=          0 0 0 0
 5:        KH=          0 0 0 0
 6:        KH=          0 0 0 0
 7:        KH=          0 0 0 0
 8:        KH=          0 0 0 0
 9:        KH=          0 0 0 0
10:
```

Screen display with parameter error codes

CODE

Cause of the error (which error occurred?)	DL (Data Word left)	DR (Data Word right)	Location of error (in which parameter block did the error occur?)
No error	00	00	OBI: Onboard Interrupt
Start or end ID is missing	01	01	OBC: Onboard Counter
Comment not closed off correctly	02	02	OBA: Onboard analog connections
Before END: semicolon missing in front of END		03	SL1: SINEC L1
Syntax error - block ID	03		
Syntax error - parameter	04	06	CLP: Clock parameter
Syntax error - argument	05		
Range exceeded in an argument	06	09	TFB: Timer function block
Parameter combination is not allowed	07	11	SDP: System data parameter
Not defined	08		
Not defined	09		
DB is not present	10		
Not enough space in DB	11	99	ERT: Error return
Error inputting weekday	12	FB	Error can not be assigned to any block
Error in the date	13		
Error in inputting time	14		
Irregular time format in parameter blocks (24h:12h-Mode)	15	FF	

Figure 9-3. Parameter Error Codes and Their Meaning

Locating Parameter Errors in "ISTACK"

If the programmable controller recognizes an error in DB1 in the initial start-up, then the programmable controller remains in the "STOP" mode and stores a message in "ISTACK" describing where the error happened. The "ISTACK" contains the absolute error address as well as the relative error address.

The STEP Address Counter (SAC) in the ISTACK points either **to** the address that contains the incorrect input or **in front of** the address that contains the incorrect input. These are byte addresses.

Example: Your inputs into DB1 are as follows. The position shaded contains an error.

```
 0:    KS  ='DB1 OBA: AI 0 ; OBI:     '
12:    KS  ='  ; OBC: CAP N      CBP '
24:    KS  ='N       ; SL1: SLN 40 ST '
36:    KS  ='DB2  DW0   EF DB3  DW0  '
48:    KS  ='  KBE MB100      KBS MB1'
60:    KS  ='01      PGN 1   ; SDP: N'
72:    KS  ='T 128 PBUS N ; TFB: OB13'
84:    KS  ='  100    ; #CLP: STW MW10'
96:    KS  ='2        CLK DB5  DW0   '
108:   KS  ='  SET 3  01.10.91 12:00:'
120:   KS  ='00    DHS 000000:00:00  '
132:   KS  ='  TIS 3 01.10. 12:00:00 '
144:   KS  ='  STP Y SAV Y  CF 00    '
156:   KS  =' ; #END ';
```

The decimal numbers in front of each input line represent the word address for the first character that can be entered for that respective line. Each word consists of two characters (2 bytes).

Figure 9-4. Erroneous Parameter Assignment in DB1

The error causes ISTACK to display the following addresses:

- The absolute (error) address: $014C_H$ (absolute SAC)
- The relative (error) address: 0042_H (relative SAC)

So that you can locate the error in DB1 exactly, you must convert the relative byte address that is displayed in hexadecimal format into a decimal word address. Decimal format is required because the programmer displays a DB in words.

0042_H	=	66_D	66_D	:	2_D	= 33_D
Hexadecimal byte address		Decimal byte address				Decimal word address

The information displayed in the chart above shows that the error occurred after address 24 and before address 36. In Figure 9-4, argument 40 occupies address 33; the "40" is an incorrect entry. The error is due to a range violation.

9.1.6 Transferring DB1 Parameters to the Programmable Controller

Unlike other data blocks, DB1 is processed only one time. This occurs when a cold restart is performed on the programmable controller. This was done so that DB1 could handle certain special functions.

One such special function is the assignment of parameters in the programmable controller with the help of DB1. Setting parameters means that you enter parameters in DB1 for those internal functions that your programmable controller should work with.

The programmable controller's operating system only accepts these inputs into DB1 when there is a cold restart. You must perform a cold restart anytime you make changes to DB1. You can perform a cold restart by switching from POWER OFF to POWER ON or from STOP to RUN.

The programmable controller accepts the parameters from DB1 and stores them in the system data area.

Note

The programmable controller remains in the "STOP" mode if a parameter assignment error is found during start-up. The red LED lights up on the operator panel and ISTACK displays a DB1 addressing error.

9.1.7 Reference Guide for Setting Parameters in DB1

Parameter	Argument	Explanation
Block ID: OBA:		**Onboard Analog Inputs**
AI	p	Number of analog inputs read in cyclically
p = 0 to 8		0 = no analog channel read in
Block ID: OBI:		**Onboard Interrupt**
IP	p	Interrupt, positive edge, channel p
IN	p	Interrupt, negative edge, channel p
IPN	p	Interrupt, positive and negative edge, channel p
INP	p	Interrupt, negative and positive edge, channel p
p = 0 to 3		
Block ID: OBC:		**Onboard Counter**
CAP	p/N	Counter A, positive edge, comparison value p
CBP	p/N	Counter B, positive edge, comparison value p
CAN	p/N	Counter A, negative edge, comparison value p
CBN	p/N	Counter B, negative edge, comparison value p
CCP	q/N	Cascaded counter, positive edge , Comparison value q
CCN	q/N	Cascaded counter, negative edge, Comparison value q
p = 0 to 65536	q = 0 to 4294967196	N/n = not activated
Block ID: SL1:		**SINEC L1**
SLN	p	Slave number
SF	DBx DWy	Location of Send Mailbox
EF	DBxDWy	Location of Receive Mailbox
KBE	MBy	Location of Coordination Byte "Receive"
KBS	MBy	Location of Coordination Byte "Send"
PGN	p	Programmer bus number
p = 1 to 30		x = 2 to 255 y = 0 to 255
Block ID:SDP:		**System-Dependent Parameter**
NT	p	Number of timers being processed
PBUS	J/Y/N	Start-up only via connected bus units
p = 0 to 128	n/N = no	J/Y = yes
Block ID: TFB:		**Timer Function Block**
OB13	p	Intervals (ms) at which OB13 is called up and is processed
p = 0 to 655350 (State in 10 ms steps)		

S5-95U

Parameter	Argument	Meaning
Block ID: CLP:		**Clock Parameters**
STW	DBxDWy, MWz,EWv or AWv	Location of the status word (STatus Word)
CLK	DBxDWy, MWz,EWv or AWv	Location of the clock data (CLocK Data)
SET	wd dd.mm.yy hh:mm:ss[1] AM/PM[2]	Setting the clock time and date
OHS	hhhhhh:mm:ss[1]	Setting the operating hours counter (Operating Hours counter Set)
OHE	J/Y/N	Enabling the operating hours counter (Operating Hours counter Enable)
TIS	wd dd.mm hh:mm:ss[1] AM/PM[2]	Setting the prompting time (Timer Set)
STP	J/Y/N	Updating the clock during "STOP" (STоP)
SAV	J/Y/N	Saving the clock time after the last change from "RUN" to "STOP" or "POWER OFF" (SAVe)
CF	p	Inputting the correction factor (Correction Factor)

wd = 1 to 7 (weekday = Sun..Sat)
dd = 01 to 31 (day)
mm = 01 to 12 (month)
yy = 0 to 99 (year)
hh = 00 to 23 (hours)
mm = 00 to 59 (minutes)
ss = 00 to 59 (seconds)
hhhhhh = 0 to 999999 (hours)

p = – 400 to 400
v = 0 to 126
x = 2 to 255
y = 0 to 255
z = 0 to 254
J/J = ja/yes)
Y/Y = yes
N/N = no

1 If an argument such as seconds, for example, is not to be entered, input XX. The clock continues to run with the updated data. The TIS parameter block does not acknowledge this argument.

2 If you input AM or PM after the clock time, the clock runs in the 12-hour mode. If you omit this argument, the clock runs in the 24-hour mode. You must use the same time mode in the SET and TIS parameter blocks.

S5-90U

Parameter	Argument	Explanation
Block ID: OBit:		**Onboard Interrupt**
IP	0	Interrupt, positive edge I 33.0
Block ID: OBC:		**Onboard Counter**
C4P	p/N	Counter, positive edge, comparison value p
p = 0 to 65.536	N/n = not activated	
Block ID: SIL1:		**SINEC L1**
SLN	p	Slave number
SF	DBx DWy	Location of Send Mailbox
EF	DBxDWy	Location of Receive Mailbox
KBE	MBz	Location of Coordination Byte "Receive"
KBS	MBz	Location of Coordination Byte "Send"
PGN	p	Programmer bus number
p = 1 to 30	x = 2 to 63	y = 0 to 255 z = 0 to 127

9.1.8 Defining System Characteristics in DB1 (only for the S5-95U)

You can set some system characteristics of the S5-95U in parameter block "SDP:" (System Dependent Parameter).

- You can set how many of the 128 internal timers the processor is constantly processing (Parameter INT)
- You can select whether the programmable controller should take into consideration the connected S5-100U modules during start-up (Parameter PBUS).

Before giving an example for assigning parameters for parameter block "SDP:", we will explain the meaning of the two system characteristics you can assign with the parameters.

The parameter block for the internal timer "INT" is preset so that all 128 timers are constantly being processed. You can shorten the required processing time if you set only the internal timers that you really need.

The parameter "PBUS" is preset so that the programmable controller even starts up if the error message PEU (external I/O modules not ready) appears during start-up. If you want to be certain that the programmable controller starts up only if the "PEU" message is not displayed, then you have to change the PBUS parameter.

The dependency relationship between "PEU" and "PBUS" is shown in Table 9.3

Table 9-3. Programmable Controller Start-Up Procedure, Dependent on PEU and PBUS

PEU \ PBUS	No	Yes
No	Programmable Controller starts up and goes into the "RUN" mode. I/Os are addressed via the bus module's serial bus.	Programmable Controller starts up and goes into the "RUN" mode. I/Os are addressed via the bus module's serial bus.
Yes = No bus is connected or there is a fault	Programmable Controller recognizes "PEU" during start-up. Because the "no bus present" parameter was set (PBUS N), the programmable controller goes into the "RUN" mode and the I/O modules are not addressed.	Programmable Controller recognizes "PEU" during start-up. Because the "bus present" parameter was set, there is an error on the bus. The "PEU" message in ISTACK causes the programmable controller to go into STOP.

Example: The user program only requires 25 internal timers and you want to be certain that the programmable controller starts up only if the external I/O modules are ready.

How to proceed:

- Display DB1 on the programmer.
- Change the parameter block "SDP:" as described in Figure 9-5.
  - Position the cursor below the parameter arguments
  - Overwrite the arguments
- Transfer the changed DB1 to the programmable controller.
- Switch the programmable controller from STOP to RUN: the programmable controller accepts the changed parameters.

```
 0:     KS  ='DB1 OBA: AI 0 ; OBI:     '
12:     KS  ='  ; OBC: CAP N       CBP '
24:     KS  ='N       ;#SL1: SLN 1 SF '
36:     KS  ='DB2  DW0   EF DB3  DW0   '
48:     KS  ='  KBE MB100       KBS MB1'
60:     KS  ='01       PGN 1  ;# SDP: N'
72:     KS  ='T 025 PBUS N ; TFB: OB13'
84:     KS  =' 100     ; #CLP: STW MW10'
96:     KS  ='2         CLK DB5  DW0   '
108:    KS  ='  SET 3  01.10.91 12:00:'
120:    KS  ='00     OHS 000000:00:00  '
132:    KS  ='  TIS 3 01.10. 12:00:00 '
144:    KS  ='   STP Y SAV Y  CF 00    '
156:    KS  =' ; #END ';
```

Figure 9-5. Entering the Address for "System Data Parameters"

9.2 Integrated Function Blocks (only for the S5-95U)

Some standard function blocks are integrated in the S5-95U. You can call up these blocks in your control program with the commands "JU FB x" or "JC FB x". The character "x" stands for block number.

Overview:

Block No.	FB240	FB241	FB242	FB243	FB250	FB251
Block name	COD:B4	COD:16	MUL:16	DIV:16	RLG:AI	RLG:AO
Call length (in words)	5	6	7	10	10	9
Processing time (in ms)	< 0.6	< 1.0	< 0.9	< 2.1	≤ 2.4	≤ 4.8

9.2.1 Code Converter : B4 - FB240 -

Use function block FB240 to convert a number in BCD (4 tetrads) with sign to a fixed-point binary number (16 bits).
You must change a two-tetrad number to a four-tetrad number before you convert it.

- If a tetrad is not in the BCD defined range, then FB240 displays the value "0". An error bit message does not follow.

Table 9-4. Call and Parameter Assignments of FB240

Parameter	Meaning	Type	Assignment	STL
BCD	BCD number	I W	0 to 9999	: JU FB240 NAME : COD:B4 BCD : SBCD : DUAL :
SBCD	Sign of the BCD number	I BI	"1" for "-" "0" for "+"	
DUAL	Fixed-point number (KF)	Q W	16 bits "0" or "1"	

9.2.2 Code Converter : 16 - FB241-

Use function block FB 241 to convert a fixed-point binary number (16 bits) to a number in BCD code with additional consideration of the sign. An eight-bit binary number must be transferred to a 16-bit word before conversion.

Table 9-5. Call and Parameter Assignments of FB241

Parameter	Meaning	Type	Assignment	STL
DUAL	Binary Number	I W	-32768 to +32767	: JU FB241 NAME : COD:16 DUAL : SBCD : BCD2 : BCD1 :
SBCD	Sign of the BCD Number	I BI	"1" for "-" "0" for "+"	
BCD2	BCD number 4th and 5th tetrads	Q BY	2 tetrads	
BCD1	BCD number tetrads 0 to 3	Q W	4 tetrads	

9.2.3 Multiplier : 16 - FB242 -

Use function block FB 242 to multiply one fixed-point binary number (16 bits) by another. The product is represented by two fixed-point binary numbers (16 bits each). The result is also scanned for zero. An eight-bit number must be transferred to a 16-bit word prior to multiplication.

Table 9-6. Call and Parameter Assignments of FB242

Parameter	Meaning	Type	Assignment	STL
Z1	Multiplier	I W	-32768 to +32767	: JU FB242 NAME : MUL:16 Z1 : Z2 : Z3=0 : Z32 : Z31 :
Z2	Multiplicand	I W	-32768 to +32767	
Z3=0	Scan for zero	Q BI	"1" if the product is zero	
Z32	Product high-word	Q W	16 Bits	
Z31	Product low-word	Q W	16 Bits	

9.2.4 Divider : 16 - FB243 -

Use function block FB 243 to divide one fixed-point binary number (16 bits) by another. The result (quotient and remainder) is represented by two fixed-point binary numbers (16 bits each).

The divisor and the result are also scanned for zero. An eight-bit number must be transferred to a 16-bit word prior to division.

Table 9-7. Call and Parameter Assignments of FB243

Parameter	Explanation	Type	Assignment	STL
Z1	Dividend	I W	-32768 to +32767	: JU FB243 NAME : DIV:16 Z1 : Z2 : OV : FEH : Z3=0 : Z4=0 : Z3 : Z4 :
Z2	Divisor	I W	-32768 to +32767	
OV	Overflow bit	Q BI	"1" if overflow	
FEH		Q BI	"1" for division by zero	
Z3=0	Scan for zero	Q BI	"1": quotient is zero	
Z4=0	Scan for zero	Q BI	"1": remainder is zero	
Z3	Quotient	Q W	16 bits	
Z4	Remainder	Q W	16 bits	

9.2.5 Analog Value Conditioning Modules FB250 and FB251

Function block FB250 reads in an analog value from the onboard analog input module or from an analog input module and outputs a value XA in the scale range specified by the user.

Function block FB251 allows you to output analog values to analog output modules. Values from the range between the "UGR" (lower limit) parameters and the "OGR" (upper limit) parameters are converted to the nominal range of the selected module.

You will find more information on the following topics in section 12.6.

- Calling up and setting parameters in FB250
- Calling up and setting parameters in FB251
- Analog value processing with FB250 and FB251, an example

9.3 Integrated Organization Blocks (in the S5-95U only)

9.3.1 Scan Time Triggering OB31

A scan time monitor monitors the program scan time. If program scanning takes longer than the specified scan monitoring time of 300 ms, the CPU goes into the "STOP" mode. This situation can happen when one of the following errors occurs.

- The control program is too long
- The program enters a continuous loop

You can retrigger the scan time monitor at any point in the control program by calling up OB31. Calling up this block restarts the scan time monitor.

Call up OB31

- Prerequisite: SYSTEM COMMANDS "YES" has been specified on the programmer (this option is set automatically from Stage IV software on and in the LAD 90 software)
- JU OB31 can be programmed at any point in the control program.

Programming

One statement is sufficient, e.g. "BE", so that the retriggering is effective. You can input additional instructions.

9.3.2 Battery Failure OB34

The S5-95U constantly checks the status of the battery in the power supply. If a battery fails (BAU), OB34 is processed before every cycle until the battery is replaced. You can program the reaction of the programmable controller to battery failure in OB34. If OB34 is not programmed, there is no reaction.

9.3.3 OB251 PID Algorithm

A PID algorithm is integrated in the operating system of the S5-95U. OB251 helps you use this algorithm to meet your needs.

Before calling up OB251, you must first open a data block called the controller DB. It contains the controller parameters and other controller specific data. The PID algorithm must be invoked periodically and generates the manipulated variable. The more closely the scan time is maintained, the more accurately the controller fulfills its task. The control parameters specified in the controller DB must be adapted to the scan time.

You should always call OB251 from the time OB (OB13). You can set time OBs at a call up interval ranging between 10 ms and 10 minutes. The PID algorithm requires no more than 1.7 ms to process.

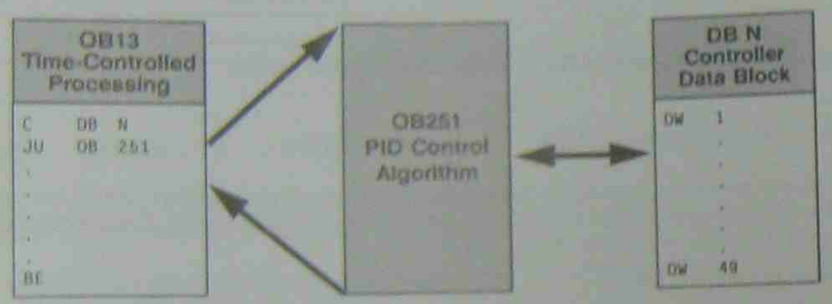


Figure 9-6. Calling Up the OB251 PID Algorithm

The continuous action controller is designed for controlled systems such as those present in process engineering for controlling pressure, temperature, or flow rate.

The "R" variable sets the proportional component of the PID controller. If proportional action is required, most controller designs use the value R = 1 (a 1000 entered in OB251 equals a value of 1 for R).

The individual Proportional action, Integral action, and Derivative action components can be deactivated via their parameters (R, TI, and TD) by presetting the pertinent data words to zero. This enables you to implement all required controller structures without difficulty, e.g., PI, PD, or PID controllers.

You can forward the system deviation XW or, using the XZ input, any disturbance variable or the inverted actual value X to the derivative action element. Specify a negative K value for a reverse acting controller.
When the correction information (dY or Y) is at a limit, the integral action component is automatically deactivated in order not to impair the dynamic response of the controller.

The switch settings in the block diagram are implemented by setting the respective bits in control word "STEU".

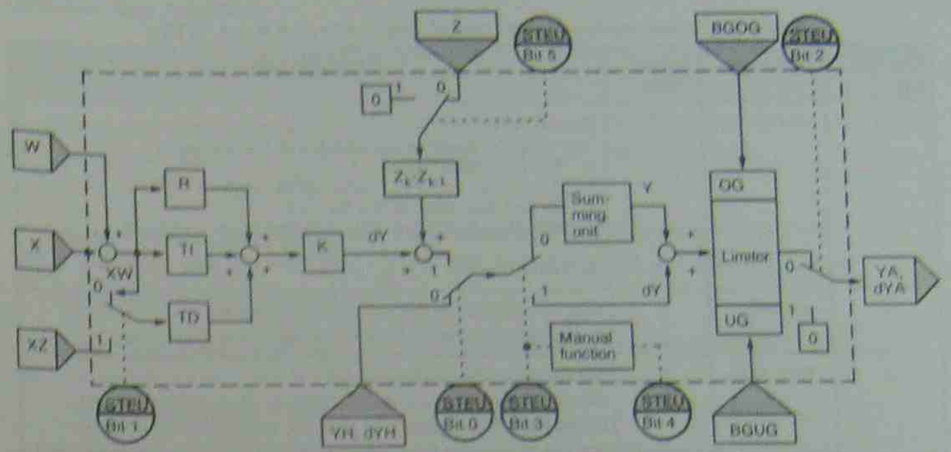


Figure 9-7. Block Diagram of the PID Controller

Table 9-8. Legend for the Block Diagram of the PID Controller

Designation	Explanation
K	Proportional coefficient K > 0 direct acting; K < 0 reverse acting
R	R parameter (proportionality component) (usually 1000)
TA	Scan time
TN	Integral-action time
TV	Derivative action time
TI	Constant TI TI = Scan time TA/Integral action time TN
TD	Constant TD TD = Derivative action time TV/Scan time TA
W	Setpoint
STEU	Control word
YH, dYH	Output value YH → Control Word Bit 3 = 0; dYH → Control Word Bit 3 = 1
Z	Disturbance variable
XW	System error
X	Actual value
XZ	Substitute value for system error
Y, dY	Manipulated variable, Manipulated increments
BGOG	Upper limit of the manipulated variable
BGUG	Lower limit of the manipulated variable
YA, dYA	Output word YA → Control Word Bit 3 = 1; dYA → Control Word Bit 3 = 0

Table 9-9. Description of the Control Bits in Control Word "STEU"

Control Bit	Name	Signal State	Description
0	AUTO	0	Manual mode The following variables are updated in Manual mode: 1) X_k, XW_{k-1} and PW_{k-1} 2) XZ_k, XZ_{k-1} and PZ_{k-1}, when STEU bit 1 = 1 3) Z_k and Z_{k-1}, when STEU bit 5 = 0 Variable dD_{k-1} is set to 0. The algorithm is not computed.
		1	Automatic mode
1	XZ EIN	0	XW_k is forwarded to the differentiator. The XZ input is ignored.
		1	A variable other than XW_k is forwarded to the differentiator
2	REG AUS	0	Normal controller processing
		1	When the controller is invoked (OB251), all variables (DW 18 to DW48) with the exception of K, R, TI, TD, BGOG, BGUG, YH_k and W_k are reset in the controller DB. The controller is deactivated.
3	GESCHW	0	Correction algorithm
		1	Velocity algorithm
4	HANDART	0	When GESCHW = 0: Following the transfer to Manual mode, the specified manipulated variable value YA is adjusted exponentially to the manual value in four sampling steps. Additional manual values are then forwarded immediately to the controller output. When GESCHW = 1: The manual values are forwarded immediately to the controller output. The limiting values are in force in Manual mode.
		1	When GESCHW = 0: The manipulated variable last output is retained. When GESCHW = 1: Correction increment dY_k is set to zero.
5	NO Z	0	With feedforward control
		1	No feedforward control
6 and 7	-		These bits are not assigned.
8 to 15	-		The PID algorithm uses these bits as auxiliary flags.

The control program can be supplied with fixed values or parameters. Parameters are input via the assigned data words. The controller is based on a PID algorithm. Its output signal can be either a manipulated variable (positioning algorithm) or a manipulated variable modification (correction rate algorithm).

Correction Rate Algorithm

The relevant correction increment dY_k is computed at instant $t = k \cdot TA$ according to the following formula:

- Without feedforward control (D11.5 = 1); XW is forwarded to the differentiator (D11.1 = 0)

$$dY_k = K[(XW_k - XW_{k-1})\,R + TI \cdot XW_k + \tfrac{1}{2}\,(TD\,(XW_k - 2XW_{k-1} + XW_{k-2}) + dD_{k-1})]$$
$$= K\,(dPW_kR + dI_k + dD_k)$$

- With feedforward control (D11.5 = 0); XW is forwarded to the differentiator (D11.1 = 0)

$$dY_k = K[(XW_k - XW_{k-1})\,R + TI \cdot XW_k + \tfrac{1}{2}\,(TD\,(XW_k - 2XW_{k-1} + XW_{k-2}) + dD_{k-1})] + (Z_k - Z_{k-1})$$
$$= K\,(dPW_kR + dI_k + dD_k) + dZ_k$$

- Without feedforward control (D11.5 = 1); XZ is forwarded to the differentiator (D11.1 = 1)

$$dY_k = K[(XW_k - XW_{k-1})\,R + TI \cdot XW_k + \tfrac{1}{2}\,(TD\,(XZ_k - 2XZ_{k-1} + XZ_{k-2}) + dD_{k-1})]$$
$$= K\,(dPW_kR + dI_k + dD_k)$$

- With feedforward control (D11.5 = 0); XZ is forwarded to the differentiator (D11.1 = 1)

$$dY_k = K[(XW_k - XW_{k-1})\,R + TI \cdot XW_k + \tfrac{1}{2}\,(TD\,(XZ_k - 2XZ_{k-1} + XZ_{k-2}) + dD_{k-1})] + (Z_k - Z_{k-1})$$
$$= K\,(dPW_kR + dI_k + dD_k) + dZ_k$$

P component I component D component Z component k: k^{th} sample

When XW_k is applied:	XW_k	=	$W_k - X_k$
	PW_k	=	$XW_k - XW_{k-1}$
	QW_k	=	$PW_k - PW_{k-1}$
		=	$XW_k - 2XW_{k-1} + XW_{k-2}$
When XZ is applied:	PZ_k	=	$XZ_k - XZ_{k-1}$
	QZ_k	=	$PZ_k - PZ_{k-1}$
		=	$XZ_k - 2XZ_{k-1} + XZ_{k-2}$
The result is:	dPW_k	=	$(XW_k - XW_{k-1})R$
	dI_k	=	$TI \cdot XW_k$
	dD_k	=	$\frac{1}{2}(TD \cdot QW_k + dD_{k-1})$ when XW is applied
		=	$\frac{1}{2}(TD \cdot QZ_k + dD_{k-1})$ when XZ is applied
	dZ_k	=	$Z_k - Z_{k-1}$

Positioning Algorithm

The formula used to compute the correction rate algorithm is also used to compute the positioning algorithm.

In contrast to the correction rate algorithm, however, the sum of all correction increments computed (in DW 48), rather than the correction increment dY_k is output at sampling instant t_k.

At instant t_k, manipulated variable Y_k is computed as follows:

$$Y_k = \sum_{m=0}^{m=k} dY_m$$

Initializing the PID Algorithm

OB251's interface to its environment is the controller DB.
All data needed to compute the next manipulated variable value is stored in this DB. Each controller must have its own controller data block.

The controller-specific data are initialized in a data block that must comprise at least 49 data words. The CPU goes to STOP with a transfer error (TRAF) if no DB has been opened or if the DB is too short.

Caution

Make sure that the right controller DB has been opened before calling control algorithm OB251.

Table 9-10. Format of the Controller DB

Data Word	Name	Comments
1	K	Proportional gain (-32 768 to + 32 767) for controllers without a differential component Proportional gain (- 1500 to +1500) for controllers with a differential component[1] K is greater than zero when the control is direct acting, and less than zero when the control is reverse acting; the specified value is multiplied by a factor of 0.001.
3	R	R parameter (- 32 768 to + 32 767) for controllers without a differential component R parameter (- 1500 to + 1500) for controllers with a differential component[1] Normally 1 for controllers with P component; the specified value is multiplied with a factor of 0.0001
5	TI	Constant TI (0 to 9999) $TI = \frac{\text{Sampling interval TA}}{\text{Integral-action time}}$ The specified value is multiplied with a factor of 0.001.
7	TD	Constant TD (0 to 999) $TD = \frac{\text{Derivative-action time TV}}{\text{Sampling interval TA}}$
9	W	Setpoint (- 2047 to + 2047)

[1] It is possible to have larger gains, if sudden incremental changes to the system deviation are small enough. This is the reason you have to divide larger deviations into smaller ones such as adding the setpoint via a ramp function.

Table 9-10. Format of the Controller DB (Continued)

Data Word	Name	Comments
11	STEU	Control word (bit pattern)
12	YH	Value for Manual operation (- 2047 to + 2047)
14	BGOG	Upper limit value (- 2047 to + 2047)
16	BGUG	Lower limit value (- 2047 to + 2047)
22	X	Actual value (- 2047 to + 2047)
24	Z	Disturbance variable (- 2047 to + 2047)
29	XZ	Derivative time (- 2047 to + 2047)
48	YA	Output variable (- 2047 to + 2047)

All parameters (with the exception of the control word STEU) must be specified as 16-bit fixed point numbers.

Caution

The PID algorithm uses the data words that are not listed in Table 9-10 as auxiliary flags.

Initialization and Call Up of the PID Controller in a STEP 5 Program

Several different PID controllers can be implemented by calling up OB251 repeatedly. A data block must be initialized prior to each OB251 call up. These DBs serve as data interface between the controllers and the user.

Note

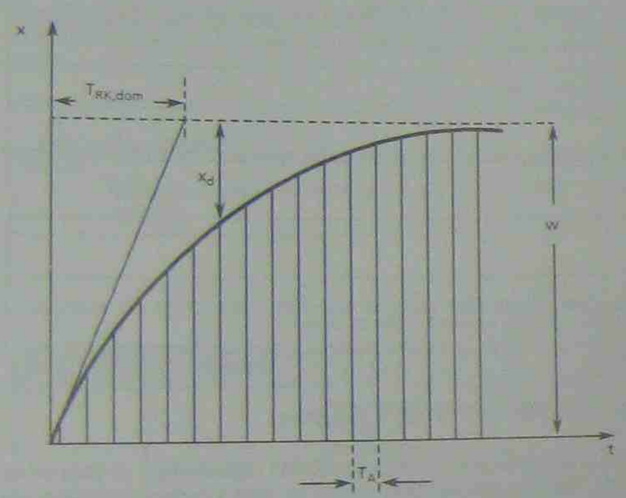
Important controller data are stored in the high-order byte of control word DW11 (DL11). Therefore make sure that only T DR 11/SU D11.0 to D11.7 or RU D 11.0 to D11.7 operations are used to modify user-specific bits in the control word.

Selecting the Sampling Interval

In order to be able to use the known analog method of consideration for digital control loops too, do not select a sampling interval that is too large.
Experience has shown that a TA sampling interval of approximately 1/10 of the time constant $T_{RK,dom}$* produces a control result comparable to the equivalent analog result. Dominant system time constant $T_{RK,dom}$ determines the step response of the closed control loop.

$$TA = 1/10 \cdot T_{RK,dom}$$

In order to ensure the constancy of the sampling interval, OB251 must always be called up in the service routine for time interrupts (OB13).



x = Control variable
t = Time
T_A = Sampling interval
$T_{RK,dom}$ = Dominant system time constant of the closed control loop
w = System error / Setpoint
x_d = Control deviation

Figure 9-8. Principle of Interval Sampling

* $T_{RK,dom}$ = dominant system time constant of the closed control loop

Example for the Use of the PID Controller Algorithm:

A PID controller is supposed to keep an annealing furnace at a constant temperature.
The temperature setpoint is entered via a potentiometer.
The setpoints and actual values are acquired using an analog input module and forwarded to the controller. The computed manipulated variable is then output via an analog output module.

The controller mode is set in input byte 0 (see control word DW 11 in the controller DB).
You must use the well-known controller design procedure to determine how to tune the controller for each controlled system.

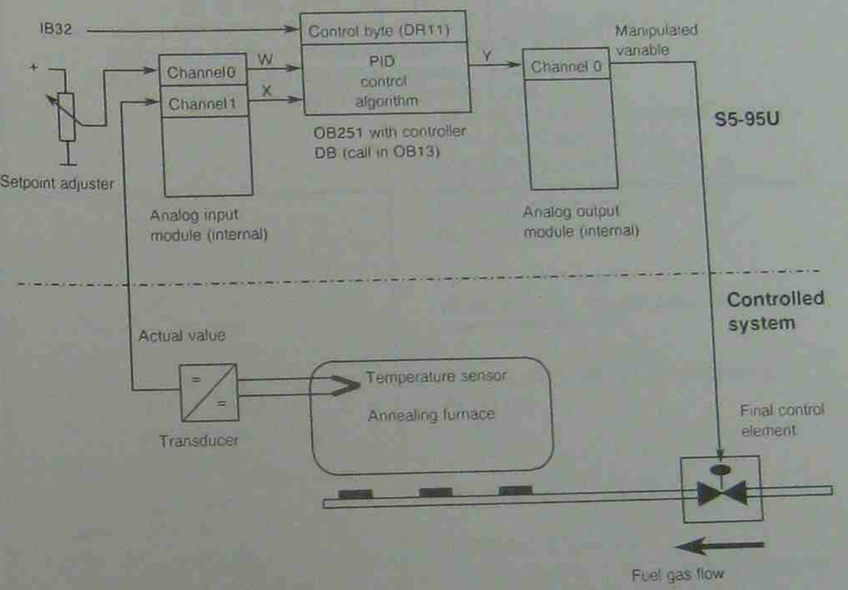


Figure 9-9. Process Schematic

The analog signals of the setpoint and actual values are converted into corresponding digital values in each sampling interval (set in OB13). OB251 uses these values to compute the new digital manipulated variable, from which, in turn, the analog output module generates a corresponding analog signal. This signal is then forwarded to the controlled system.

Calling the Controller in the Program:

OB 13	STL	Description
	:	
	:JU FB 10	
NAME	:CONTROLLER 1	PROCESS CONTROLLER
	:	
	:	THE CONTROLLER'S SAMPLING INTERVAL
	:	DEPENDS ON THE TIME BASE USED
	:	TO CALL OB13 (SET IN DB1).
	:	THE DECODING TIME OF THE ONBOARD
	:	ANALOG INPUTS MUST BE TAKEN
	:	INTO ACCOUNT WHEN SELECTING
		THE SAMPLING INTERVAL.
	:	
	:	
	:BE	

FB10	STL	Description
NAME	:CONTROLLER 1	
	:	
	:C DB 30	SELECT CONTROLLER'S DB
	:	
	:	READ CONTROLLER'S CONTROL BITS
	:	
	:	
	:L PY 32	READ CONTROLLER'S
	:T FY 10	CONTROL BITS
	:T DR 11	AND STORE IN DR11
	:	NOTE CAREFULLY:
	:	DR11 CONTAINS IMPORTANT CONTROL
	:	DATA FOR OB251
	:	THE CONTROL BITS MUST
	:	THEREFORE BE TRANSFERRED WITH
	:	T DR11 TO PREVENT
	:	CORRUPTING DL11
	:	
	:	READ ACTUAL VALUE AND SETPOINT
	:	
	:	
	:	FLAG 0 (FOR UNUSED FUNCTIONS
	:A F 12.0	IN FB 250)
	:R F 12.0	FLAG 1
	:AN F 12.1	
	:S F 12.1	
	:	READ ACTUAL VALUE
	:JU FB250	
NAME	:RLG: AI	MODULE ADDRESS
BG	: KF +8	CHANNEL NO. 0, FIXED-POINT BIPOLAR
KNKT	: KY 0,6	UPPER LIMIT FOR ACTUAL VALUE
OGR	:KF +2047	LOWER LIMIT FOR ACTUAL VALUE
UGR	:KF - 2047	NO SELECTIVE SAMPLING
EINZ	: F 12.0	STORE SCALED ACTUAL VAL. IN CONTR. DB
XA	: DW 22	ERROR BIT
FB	: F 12.2	RANGE VIOLATION
BU	: F 12.3	
	:	

FB10 (continued) STL				Explanation
	:			
	:JU	FB250		READ SETPOINT
NAME	:RLG:	AI		
BG	:	KF	+8	MODULE ADDRESS
KNKT	:	KY	1,6	CHANNEL NO. 1, FIXED-POINT BIPOLAR
OGR	:	KF	+2047	UPPER LIMIT FOR SETPOINT
UGR	:	KF	- 2047	LOWER LIMIT FOR SETPOINT
EINZ	:	F	12.0	NO SELECTIVE SAMPLING
XA	:	DW	9	STORE SCALED SETPOINT IN CONTR. DB
FB	:	F	13.1	ERROR BIT
BU	:	F	13.2	RANGE VIOLATION
	:			
	:A	F	10.0	IN MANUAL MODE, THE SETPOINT IS
	:JC	=WEIT		SET TO THE ACTUAL VALUE TO FORCE
	:L	DW	22	THE CONTROLLER TO REACT
	:T	DW	9	TO A SYSTEM DEVIATION, IF ANY,
	:			WITH A P STEP
	:			ON TRANSFER
	:			TO AUTOMATIC
	:			MODE
WEIT	:			
	:			
	:	JU	OB251	CALL CONTROLLER
	:			
	:			
	:			OUTPUT MANIPULATED CONTROLLER
	:			
	:JU	FB251		
NAME	:RLG:	AQ		
XE	:	DW	48	FORW. MAN. VAR. TO ANAL. OUTPUT MOD.
BG	:	KF	+8	MODULE ADDRESS
KNKT	:	KY	0,1	CHANNEL 0, FIXED-POINT BIPOLAR
OGR	:	KF	+2047	UPPER LIMIT FOR ACTUATING SIGNAL
UGR	:	KF	- 2047	LOWER LIMIT FOR ACTUATING SIGNAL
FEH	:	F	13.5	ERROR BIT WHEN LIMITING VAL. DEFINED
BU	:	F	13.6	RANGE VIOLATION
	:BE			

DB 30 STL				Explanation
0:	KH	=	0000;	
1:	KF	=	+01000;	K PARAMETER (HERE = 1), FACTOR 0.001
2:	KH	=	0000;	(VALUE RANGE: - 32768 TO 32767)
3:	KF	=	+01000;	R PARAMETER (HERE = 1), FACTOR 0.001
4:	KH	=	0000;	(VALUE RANGE: - 32768 TO 32767)
5:	KF	=	+00010;	TI = TA/TN (HERE = 0.01), FACTOR 0.001
6:	KH	=	0000;	(VALUE RANGE: 0 TO 9999)
7:	KF	=	+00010;	TD = TV/TA (HERE = 10), FACTOR 1
8:	KH	=	0000;	(VALUE RANGE: 0 TO 999)
9:	KF	=	+00000;	SETPOINT W, FACTOR 1
10:	KH	=	0000;	(VALUE RANGE: - 2047 TO 2047)
11:	KM	=	00000000 00100000;	CONTROL WORD
12:	KF	=	+00500;	MANUAL VALUE YH, FACTOR 1
13:	KH	=	0000;	(VALUE RANGE: - 2047 TO 2047)
14:	KF	=	+02000;	UPPER CONT. LIMIT BGOG, FACTOR 1
15:	KH	=	0000;	(VALUE RANGE: - 2047 TO 2047)
16:	KF	=	-02000;	LOWER CONT. LIMIT BGUG, FACTOR 1
17:	KH	=	0000;	(VALUE RANGE: - 2047 TO 2047)
18:	KH	=	0000;	
19:	KH	=	0000;	
20:	KH	=	0000;	
21:	KH	=	0000;	
22:	KF	=	+00000;	ACTUAL VALUE X, FACTOR 1
23:	KH	=	0000;	(VALUE RANGE: - 2047 TO 2047)
24:	KF	=	+00000;	DISTURBANCE VARIABLE Z, FACTOR 1
25:	KH	=	0000;	(VALUE RANGE: - 2047 TO 2047)
26:	KH	=	0000;	
27:	KH	=	0000;	
28:	KH	=	0000;	
29:	KF	=	+00000;	FEEDFORWARD XZ FOR DIFF.,
30:	KH	=	0000;	FACTOR 1, (- 2047 TO 2047)
31:	KH	=	0000;	
32:	KH	=	0000;	
33:	KH	=	0000;	
34:	KH	=	0000;	
35:	KH	=	0000;	
36:	KH	=	0000;	
37:	KH	=	0000;	
38:	KH	=	0000;	
39:	KH	=	0000;	
40:	KH	=	0000;	
41:	KH	=	0000;	
42:	KH	=	0000;	
43:	KH	=	0000;	
44:	KH	=	0000;	
45:	KH	=	0000;	
46:	KH	=	0000;	
47:	KH	=	0000;	
48:	KF	=	+00000;	CONTROLLER OUTPUT Y, FACTOR 1
49:	KH	=	0000;	(VALUE RANGE: - 2047 TO 2047)
50:				

EWA 4NEB 812 6065-02a

Figures

Tables

10 Interrupt Processing

This chapter provides you with the following information.

- The number of interrupt inputs available on each of the programmable controllers
- How to connect the interrupt inputs
- How to define properties of the interrupt inputs in DB1
- How the interrups are processed internally
- How to calculate reaction times to a process interrupt

10.1 Using Onboard Interrupt Inputs

The S5-90U has one interrupt input, I 33.0. The S5-95U has four interrupt inputs, I 34.0 through I 34.3.

When your programmable controller is delivered, all of its interrupt inputs are disabled (default value). You enable the interrupt inputs when you set their parameters in DB1. See section 10.1.2 for additional information on setting parameters for interrupt inputs in DB1.

If you have programmed OB3, the programmable controller automatically branches to OB3 if an interrupt is called up. If you have not programmed OB3, then either the cyclical or time-driven program continues immediately after the interrupt.

10.1.1 Connecting Interrupt Inputs

Example S5-90U: Connecting the S1 electronic sensor to the I 33.0 interrupt input.

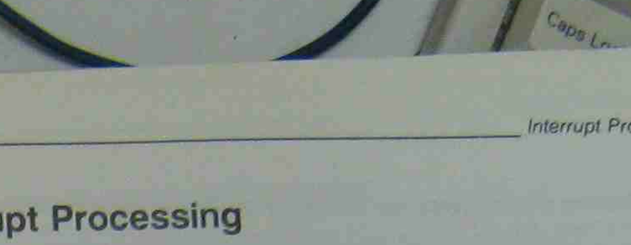


Figure 10-1. Example of a Connection for Interrupt Input (S5-90U)

Example S5-95U: Connecting the S1 electronic sensor to the I 34.0 interrupt input.

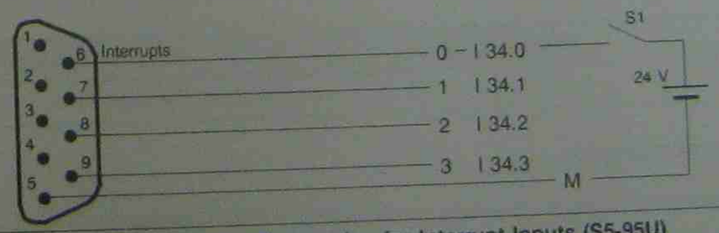


Figure 10-2. Example of a Connection for Interrupt Inputs (S5-95U)

You can connect the other interrupt inputs of the S5-95U (I 34.1 to I 34.3) in a similar way.

10.1.2 Setting Parameters for Interrupt Inputs in DB1

To set interrupt parameters, enter the following items in DB1.

- Which interrupt input should trigger the interrupt-driven program processing
- Whether the interrupt input should react to the positive, the negative, or to both edges of a pulse (S5-95U) (The interrupt input I 33.0 reacts only to the positive edge of a pulse.)

Example: Setting parameters for all four interrupts (S5-95U)

Table 10-1. Setting Parameters for Interrupt Inputs

Setting Parameters for Interrupt Inputs	Explanation
12: KS =' OBI: IP 0 IN 1 IPN 2 I' 24: KS =' PN 3 ;'	Interrupt after positive edge at I 34.0 Interrupt after negative edge at I 34.1 Interrupt after positive as well as negative edge at I 34.2 Interrupt after negative as well as positive edge at I 34.3

S5-95U
All four interrupt inputs are enabled.

S5-90U
The parameters for interrupt input I 33.0 are set just like interrupt I 34.0 as shown in Table 10.1.

Independent from the parameter settings, you can use the interrupt inputs with appropriate shielding as digital inputs (24 V).

Table 10-2. Parameters for the Interrupt Inputs

PLC	Parameter	Argument	Meaning
S5 95	Block ID: OBI:		Onboard Interrupt
	IP IN IPN INP	p p p p	Interrupt, positive edge, channel p Interrupt, negative edge, channel p Interrupt, positive and negative edge, chan. p Interrupt, negative and positive edge,chan . p
	p = 0 ... 3		
S5 90	Block ID: OBI:		Onboard Interrupt
	IP	0	Interrupt, positive edge, I33.0

Possible DB1 Parameters

You will find the procedures for setting the parameters in DB1 in chapter 9.

10.2 Programming Reactions to Interrupts in OB3

Interrupt-driven program processing is possible only after you have met the following prerequisites.

- The interrupt input or inputs in DB1 are enabled.
- The programmable controller must be in the POWER ON state and in the RUN mode.
- There is no IA operation in your program that would disable interrupt processing. See also section 8.2.8.
- OB3 is programmed.

- **Triggering an Interrupt**

  You can trigger an interrupt by calling up an interrupt on an enabled interrupt input. You can also trigger an interrupt if the comparison value of a counter has been reached. See chapter 11 for additional information.

  The programmable controller automatically branches to OB3 if an interrupt is triggered. If you have not programmed OB3, either the cyclic or time-driven program continues immediately after the interrupt. You can interrupt the cyclically processed program after every STEP 5 statement.

  Use the IA command to disable interrupt processing. Use the RA command to enable interrupt processing. The default is RA. Interrupts are stored during IA.

Note

Even for interrupt processing, you may not exceed the general block nesting depth of 16 levels.

- **Interrupt Priority**

  S5-95U:
  Interrupts cannot interrupt each other but are executed sequentially. A maximum of eight new causes for interrupts can be stored during the processing of an interrupt. The queue is processed according to the sequence in which interrupts occurred. If there are more causes for interrupts present than can be stored in the queue, then the excess causes are summarized in the last entry.

  S5-90U:
  You cannot stop interrupt processing once it has started. If an interrupt and a counter overflow occur simultaneously, interrupt processing has priority. Only one new cause for an interrupt can be stored while interrupt processing is running. If more causes for interrupts are present than can be stored, then they are ignored.

- **Scanning the Cause for an Interrupt**

S5-95U:
Positive and/or negative edges trigger interrupts on one or more interrupt inputs. OB3 is called up if it is programmed. The appropriate bit in diagnostic byte IB35 is set to "1" even if OB3 is not programmed as follows.

Bit 35.4 for I 34.0
Bit 35.5 for I 34.1
Bit 35.6 for I 34.2
Bit 35.7 for I 34.3

The bits are immediately read into the process image input table (PII), not when the first cyclical input into the PII is made.

S5-90U:
A positive edge at interrupt input I 33.0 triggers an interrupt. OB3 is called up if it is programmed. Bit 35.4 in diagnostic byte IB35 is set to "1" even if OB3 is not programmed.

Table 10-3. Example of Scanning for the Cause of an Interrupt in the Diagnostic Byte

Example	STL	Explanation
The diagnostic byte is to be evaluated within the program.	A I 35.4 = Q 32.0 BE	If bit 4 in the diagnostic byte is set to "1", signal 1 is assigned to Q 32.0.

Note

Reset the interrupt display bits to "0" after interrupt processing. The respective bits are set again when another interrupt is called up.

Table 10-4. Resetting the Diagnostic Byte in the Program after Interrupt Processing

Example	STL	Explanation
You want to reset the appropriate bit in the diagnostic byte after the interrupt.	OB3 A I 35.4 R I 35.4 JC FB3 . . BE	There was an interrupt at I 34.0 Bit 4 in the diagnostic byte is reset to "0". The interrupt reaction program in FB3 is executed.

It is also possible to scan by means of direct I/O access (L PBx) or by reading in the PII (EBx). Such a scan is not a good idea because the PII is not necessarily updated. It is also possible that the signal state has changed since an edge caused an interrupt.

Table 10-5. Scanning for the Causes of an Interrupt - Direct I/O Access / PII

Access Possibilities: \ PLCs	S5-90U	S5-95U
Interrupt Location	EB 33	EB 34
Scan the interrupt inputs by reading in the PII[1] by direct I/O access[2]	L EB 33 L PB 33	L EB 34 L PB 34

[1] The value that is read in corresponds to the last signal state read into the PII.
[2] The value that is read in corresponds to the updated signal state.

- **Addressing the Outputs**

When OB3 is used for interrupt processing, transfer can be made only to the onboard I/O modules. During the T PB/PW command, data is transferred directly to the respective onboard outputs. The normal PIQ is written to simultaneously. There is no interrupt PII and no interrupt PIQ. Internal timers are not updated during interrupt processing.

Table 10-6. Possibilities for Addressing the Outputs

Addressing Possibilities	Instructions for S5-90U	Instructions for S6-95U	Explanation
By direct I/O access	T PB 32/33 T PW 32	T PB 32/33 T PW 32/40	The contents of ACCU 1 are directly written to the onboard outputs and to the normal PII.
By writing to the PIQ	T QB 32/33 T QW 32	T QB 32/33 T QW 32/40	The contents of ACCU 1 are written to the PIQ.

10

Figure 10-3 shows how data transfer between the process image I/O tables and ACCU 1 takes place via the various load and transfer statements in OB3.

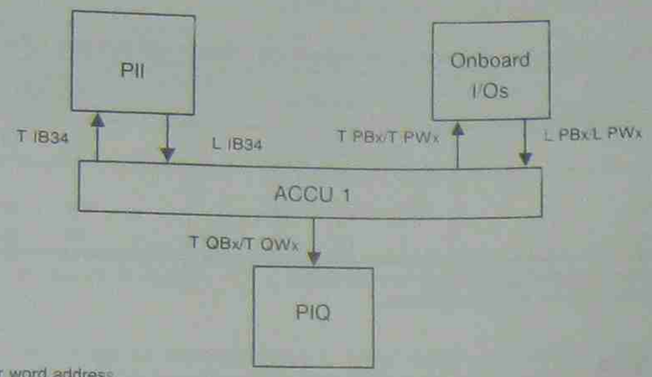


Figure 10-3. Overview of the Ways OB3 Can Access the Process I/O Images in the S5-95U

- **Programming Example for OB3 in the S5-95U**

Table 10-7. OB3 Programming Example

Example	STL	Explanation
An electronic sensor is connected to interrupt input I 34.0.	`A I 35.4` `R I 35.4`	Bit 4 in the diagnostic byte (IB35) is set. Bit 4 in the diagnostic byte (IB35) is being reset.
Branch to FB12 if the sensor triggers an interrupt.	`JC FB12`	Branch to FB12 if a cause for an interrupt has occurred.

Caution

Be sure to save the flags (into a data block for example) if these flags are to be overwritten during interrupt processing and are needed in the cycle again.

10.3 Calculating Interrupt Reaction Times

The interrupt reaction time is the amount of time that passes from the first occurrence of an interrupt until the first statement in OB3 is called up. The interrupt reaction time is dependent upon the delay time of the interrupt input and upon the corresponding operating system run time.

Calculate the programmable controller's interrupt reaction times as follows:

Programmable controller's interrupt reaction time = basic reaction time + additional reaction times

The basic reaction time is 0.6 ms and is valid if the following conditions exist.

- No integrated FBs were used.
- The parameters for the integral clock are not set.
- There are no programmer/OP functions present.
- No SINEC L1 is connected.

The additional reaction times are variable. They are listed in Table 10.8.

Table 10-8. Additional Reaction Times

Additional Running Functions of the Programmable Controller	Variable Reaction Times
Integrated FBs	≤ 0.5 ms
Parameters set for clock	≤ 0.2 ms
SINEC L1-Bus to the programmer interface	≤ 8.0 ms
OP Functions	≤ 0.4 ms
Programmer offline functions: BLOCK STATUS/TRANSFER OUTP ADDR BLOCK COMPRESS - If no blocks are moved - If blocks are moved	≤ 0.5 ms 18 ms per kByte - Depending on the number of blocks present (after overall reset 31 ms) - 600 ms per 1kword instructions in the block to be moved

EWA 4NEB 812 6065-02a

Figures

Tables

11 Onboard Counter Inputs

The S5-90U has one counter input I 33.1, while the S5-95U has two counter inputs. All counter inputs are disabled at the factory (default value). You have to set the relevant parameters in DB1 to use them and thus enable the counter inputs (see section 9.1). The counter inputs are 24-V inputs.

A 16-bit wide counter register is assigned internally to each counter input. The counter registers count the pulses that enter at the enabled counter inputs.

In the following text, we differentiate between counter inputs and counter registers only when necessary. Otherwise, we briefly refer to a counter (S5-90U) or counter A and counter B (S5-95U).

11.1 Connecting Counter Inputs

Example S5-90U: A sensor S1 is connected to counter input I 33.1 (IW36).

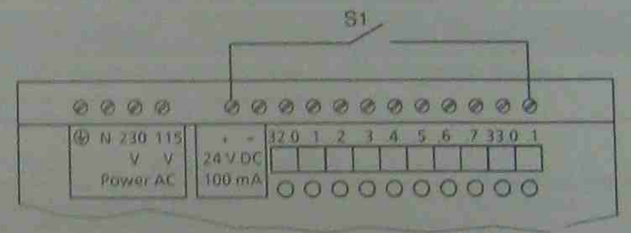


Figure 11-1. Connecting a Counter Input (S5-90U)

Example S5-95U: A pulse generator is connected to counter A (IW36).

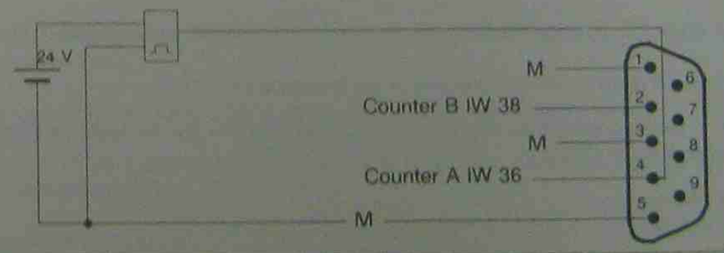


Figure 11-2. Connecting Counter Inputs (S5-95U)

You can connect counter B (iW 38) the same way as counter A.

11.2 Setting Parameters in DB1

To set counter parameters, enter the following information into DB1.

- Which counter should count (S5-95U)
- Whether a counter should count the positive or the negative edge of a pulse (S5-95U) (The S5-90U counts only positive edges.)
- What amount the counter count should count to (comparison value)

Independent of the parameters set, the following statements are true for the counters.

- The counters only count up.
- The maximum counter frequency without connection to a programmer, an OP or to SINEC L1 is 1 kHz (S5-90U) or 5 kHz for counter A and 2 kHz for counter B (S5-95U).
- Appropriate shielding makes it possible to use the counter inputs with the as digital inputs (24 V).

Example: Setting Parameters for Counter A and Counter B (S5-95U)

Table 11-1. Setting Parameters for Counter Inputs

Setting Parameters for Counter Inputs	Explanation
`12: KS =' OBC: CAP 500 CBN 999 ;` `24: KS =' '`	Counter A counts positive edges; comparison value 500 Counter B counts negative edges; comparison value 999

Explanations to the Example:

S5-95U
Counter A and counter B count independently of each other. If the counters reach the comparison value (counter overflow), the following takes place.

- An interrupt is triggered, and OB3 is called up, if it has been programmed.
- Bit 35.0 is set to "1" (for counter A) in the diagnostic byte (IB 35), independent of the presence of OB3.
- Bit 35.1 is set to "1" (for counter B) in the diagnostic byte (IB 35), independent of the presence of OB3.
- The counters are reset to "0" independent of the presence of OB3.

The "set" bits in the diagnostic byte that display a counter overflow can be reset by using either of the following operations in OB3:

R I 35.0 for counter A
R I 35.1 for counter B

S5-90U
You set the parameters for counter input I 33.1 as shown for counter A for the S5-95U (see Table 11-1).

- An interrupt is triggered and OB3 is called up, if it has been programmed.
- Independent of the presence of OB3, bit 35.0 is set to "1" in the diagnostic byte (IB 35).
- The counter is always reset to "0".

You reset the "set" bit in the diagnostic byte with the R I 35.0 operation.

Table 11-2. Resetting the Diagnostic Byte after Counter Overflow

Example	STL	Explanation

You want to reset the appropriate bit in a diagnostic byte after the counter overflow.	`OB3`
	`A I 35.0`
	`R I 35.0`
	
	`JC FB3`
	`.`
	`.`
	`BE`
		Counter A counted until it reached the comparison value. An interrupt was triggered.
		Bit 0 in the diagnostic byte is reset to "0".
		The counter reaction program in FB3 is processed.

Cascading Counters (with the S5-95U)

Instead of using counter A and counter B individually and independently of each other, you can combine them (cascading). Use cascading counters if you want to count amounts larger than 65,535 (up to 4,294,967,295). The registers of both counters work together like a larger counter with more digits. If you cascade counters, counter input B is automatically disabled; therefore, you must transfer the incoming pulses to counter input A. (see section 11.3).

Example: Setting parameters for cascading counters

Table 11-3. Setting Parameters for Cascading Counters

Set Parameters for Cascading Counter	Explanation
`12: KS =' ; OBC: CCN 90000 ;'`	Counter cascade counts negative edges; comparison value 90,000

Explanations to the Example:

Both counter registers work together as one counter. If the cascading counter exceeds the comparison value, then:

- An interrupt is triggered and OB3 is called up, if it has been programmed.
- Bit 35.0 and bit 35.1 are set to "1" at the same time in the diagnostic byte (IB 35) independent of the presence of OB3.
- The counter is reset to "0".

The "set" bits in the diagnostic byte that displayed a counter overflow can be reset by using bit operations R I 35.0 and R I 35.1.

To scan the counter status of the cascading counter and to set the counter to "0", use the same operations that you use to set both counters individually (see section 11.3).

Possible DB1 Parameters
You set parameters in the S5-90U for counter A - positive edge. You can use all parameters in the S5-95U. See chapter 9 on how to program DB1.

Table 11-4. Parameters for Counter Inputs

Parameter	Argument	Explanation
Block ID: OBC:		**Onboard Counter**
CAP	p/N	Counter A, positive edge, comparison value p
CBP	p/N	Counter B, positive edge, comparison value p
CAN	p/N	Counter A, negative edge, comparison value p
CBN	p/N	Counter B, negative edge, comparison value p
CCP	q/N	Cascading counter, positive edge, comparison value q
CCN	q/N	Cascading counter, negative edge, comparison value q
p = 0 to 65,535		q = 0 to 4,294,967,295 N/n = not activated

11.3 Scanning Counter Status and Resetting

Table 11.5 shows the access possibilities for counters in both programmable controllers. The table is followed by a program example that shows how to enter a new comparison value with the S5-95U.

Table 11-5. Direct Access Possibilities for Counters

Program. Controllers / Access Possibilities:	S5-90U	S5-95U
Location of the counters	IB 36 to 37	Counter A: IB 36 to 37 Counter B: IB 38 to 39
Scanning counter status by reading in the PII (The value read in corresponds to the last signal status read into the PII.)	L IW 36	L IW 36 L IW 38
by direct access to I/Os (The value read in corresponds to the current signal status.)	not possible	L PW 36 L PW 38
Reset counter to "0" and input a new comparison value.	not possible	L KF x1 T PW 36 T PW 38

x = 0 to 65,535 comparison value up to where the counter should start counting (you **must** enter this).

Note

If you enter a comparison value larger than 65,536, the programmable controller stops with the message NNN in the ISTACK.
If you enter KF 0 as the comparison value, the counter is set to the maximum value.

Example: How to enter a new comparison value in counter A.

STL	Explanation
OB3 A I 35.0 R I 35.0 JC FB10 BE	Counter A counted until it reached a comparison value. An interrupt was triggered. Set bit 35.0 in the diagnostic byte is reset. The program branches to FB10.

STL	Explanation
FB10 L KF x T PW 36 BE	Reaction to "comparison value reached" in the control program. The new comparison value x is loaded in ACCU 1 and counter A is set.

Example: How to enter the new comparison value 1,280,520 for the cascading counter.

- Set the parameters for the cascading counter in DB1 (see Table 11.3).
- Calculate the hexadecimal value of 1, 280, 520.

Counter Register - Cascading Counter

	IW 38											IW 36											
	16											0											
dual:	0	0	0	0	0	0	1	0	0	1	1	1	0	0	0	1	0	1	0	0	0	0	0	1	0	0	0
hex.:	0			1			3			8			A			0			8					

- Use your programmer to enter the following STL into the programmable controller.

STL	Explanation
FB11 L KH 0013 L KH 8A08 T PW 36 BE	Reaction to "comparison value reached" in the control program. The new comparison value is loaded in ACCU 1 and the cascading counter is set.

Figures

Tables

EWA 4NEB 812 6065-02a

12 Analog Value Processing

12.1 Analog Value Processing with Onboard I/Os

The S5-95U has eight analog inputs and one analog output in one 15-pin sub-D female socket.

12.1.1 Setting Parameters and Wiring of the Interface for Analog Inputs and Outputs

There are two possibilities for reading in analog values via analog inputs of the onboard I/Os.

- Via direct I/O access
- Via FB250 (analog value reading)

Reading Analog Values via Direct I/O Access

Default settings in DB1 allow only direct access to the analog I/O. The parameters in DB1 must be modified to allow **cyclical** scanning of the analog I/O. Direct access to the I/O (PW command) is only allowed in the time and interrupt OBs (OB3 and OB13).

Example:
Read in Channel 0 with:

```
L PW 40
T IW 40
```

Reading Analog Values via FB250

The prerequisite for reading in an analog value with FB250 (analog value reading and scaling) is that the analog value is stored in the PII, in an input word. If the analog inputs are to be cyclically read into the PII set the parameters in DB1 before starting up the programmable controller.

Procedure:

- Display the default DB1 on the programmer
- Change parameter block "OBA: AI 0:..." to "OBA: AI n" (n = 1 to 8, n = number of analog inputs (channels) that are to be read in cyclically into the PII

The programmable controller then scans channels 0 through n-1 cyclically. The analog value that was cyclically read in is stored in the PII (IW 40 through IW 54). Direct I/O access is no longer necessary.

Example:

`OBA: AI 2` means: Channels 0 and 1 are read in cyclically
`OBA: AI 3` means: Channels 0, 1 and 2 are read in cyclically.

The analog values that were read in are assigned to the input address range IW 40 through IW 54:

Channel 0: IW 40
Channel 1: IW 42
:
Channel 7: IW 54.

EWA 4NEB 812 6065-02a

The channels that were scanned cyclically can be read in directly with the integrated function block FB250 (analog value and scaling).

If the channels are not cyclically read in but you still want to read in analog values with the integrated function block FB250, you must do one of the following.

- Set the bit "single scan" of function block FB250.
- Trigger the conversion of the analog value before calling up the FB250 by reading in the analog value with "L PW..." and transferring it to the appropriate input range (direct I/O access).

The analog voltage output (output "+QV", pin 14) or the analog current output (output "+QI", pin 10) is assigned to output word QW40. You can, however, connect only one of these analog outputs.

Example: A voltage sensor is connected to Channel 0 (IW 40) and a load resistor is connected to the analog output "Voltage" (QW40).

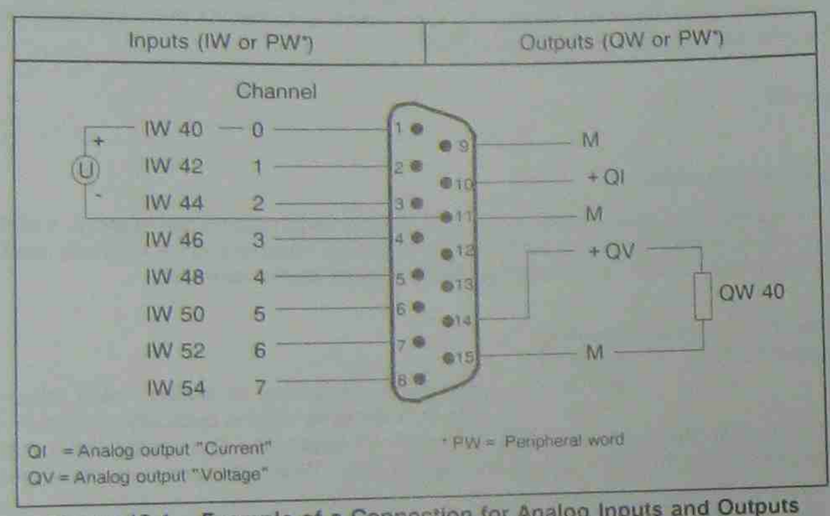


Figure 12-1. Example of a Connection for Analog Inputs and Outputs

Note

Terminals 12 and 13 are assigned internally and can not be connected. Terminals 9, 11, and 15 are internally connected chassis ground terminals (M terminals) of the analog inputs/outputs.

Using Analog Inputs as Additional Digital Inputs

With a trick you can use the analog inputs of the onboard I/Os as digital inputs.

- Connect sensors with signal voltages >10 V to analog inputs of the onboard I/Os (assign channels in iascending sequence).
  Also when using the analog inputs as "digital" inputs, check the installation guidelines for analog signal cables (e.g., use shielded cables).
- Set parameters in DB1 for cyclical scanning of the analog inputs used.
- In the control program, scan bit 0 of the respective input word assigned to the connected analog input (overflow bit or O-bit):
  O-bit = 1 → Signal status "1"
  O-bit = 0 → Signal status "0"

Now you have eight additional "digital" inputs that you can use.

12.1.2 Representation of Analog Values (Onboard I/Os)

The following representation of the bit pattern is required so that analog input/output values can be read in or out with the aid of the integrated FB250 (analog value reading and scaling) and FB251 (output analog value).

Table 12-1. Representation of an Analog Value as a Bit Pattern (Onboard I/Os)

	High Byte								Low Byte							
Bit Number	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Analog Input	S	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	0	X	X	OV
Analog Output	S	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	X	X	X	X

Key:
S Sign bit: 0 = "+"
X Irrelevant bits
OV* Overflow bit 0 = Measured value 1023 units at the most
1 = Measured value greater than 1023 units

* set to ≥ 10V

The assignments between units (IW 40 to 54) and the measured value look as indicated in Table 12-2.

Table 12-2. Analog Value Representation of the Inputs (0 to 10V) (Onboard I/Os)

Units	Measured Values in V	High Byte	Low Byte	Range
1024	≥ 10	0 1 0 0 0 0 0 0	0 0 0 0 0 0 0 1	Overflow
1023	9.9904	0 0 1 1 1 1 1 1	1 1 1 1 0 0 0 0	Nominal range
512	5	0 0 1 0 0 0 0 0	0 0 0 0 0 0 0 0	
1	0.0098	0 0 0 0 0 0 0 0	0 0 0 1 0 0 0 0	
0	0.0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	
0	<0.0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 1	Overflow

The output voltages or currents corresponding to the units (in QW40) are represented in Table 12-3. Should the value in QW40 be larger than 1024 units, 10 V (or 20 mA) are output automatically. If the sign bit (bit 7 in high-byte) is set (negative value), 0 V (or 0 mA) is output automatically.

Table 12-3. Analog Value Representation of the Outputs (Onboard I/Os)

Units	Output Value in V ①	Output Value in mA ②	High Byte	Low Byte	Range
1024	10	20	0 1 0 0 0 0 0 0	0 0 0 0 0 0 0 0	Nominal range
512	5.0	10	0 0 1 0 0 0 0 0	0 0 0 0 0 0 0 0	
1	0.0098	0.0195	0 0 0 0 0 0 0 0	0 0 0 1 0 0 0 0	
0	0.0	0.0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	

① Voltage output 0 to 10V
② Current output 0 to 20 mA

12.2 Analog Input Modules - Connecting Current and Voltage Sensors to Analog Input Modules

Analog input modules convert analog process signals to digital values that the CPU can process (via the process image input table, PII). In the following sections, you will find information about the operating principle, wiring methods, and start-up and programming of analog input modules.

Observe the following rules to connect current and voltage sensors to analog input modules.

- When you have multi-channel operations, assign the channels in ascending order. This shortens the data cycle.
- Use terminals 1 and 2 for the connection of a compensating box (464-8MA11) or for the supply of two-wire transducers (464-8ME11)

Terminals 1 and 2 cannot be used with the remaining analog input modules.

- Short-circuit the terminals of unused inputs.
- The potential difference between the common references of the inputs must not exceed 1 V. To prevent this, set the reference potentials of the sensors to a common reference potential.

12.2.1 Voltage Measurement with Isolated/Non-Isolated Thermocouples

Module **464-8MA11/8MA21** is recommended for voltage measurement with thermocouples. With **floating sensors** (e. g., isolated thermocouples), the permissible potential difference V_{CM} between theminus terminals of the inputs and the potential of the standard mounting rail must not be exceeded. To avoid this, the negative potential of the sensor must be connected to the central ground point (see Figure 12-2). Jumper terminals 1 and 2 together if you do not use compensation boxes.

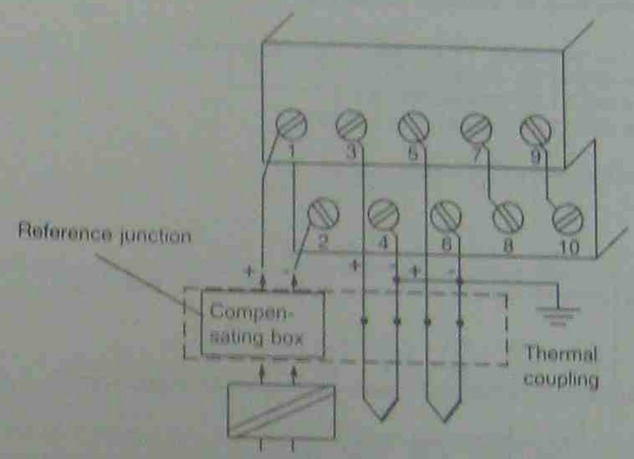


Figure 12-2. Voltage Measuring with Isolated Thermocouples (6ES5 464-8MA11/8MA21)

With **non-floating** sensors (e. g., non-isolated thermocouples), the permissible potential difference V_{CM} must not be exceeded (see maximum values of the individual modules).

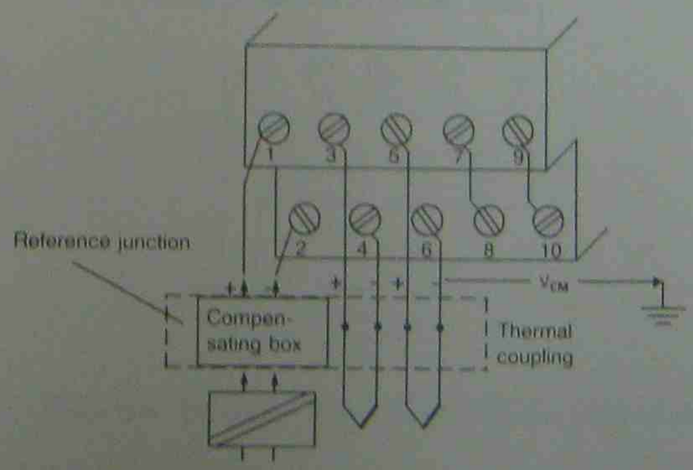


Figure 12-3. Voltage Measuring with Non-Isolated Thermocouples (Module 6ES5 464-8MA11/8MA21)

12

Connection of Thermocouples with Compensating Box to Module 464-8MA11/8MA21

The influence of the temperature on the reference junction (e. g., terminal box) can be compensated for with a compensation box. Observe the following rules.

- The compensation box must have a floating supply.
- The power supply must have a grounded shielding winding.
- The compensation box must be connected to terminals 1 and 2 of the terminal block.

12.2.2 Two-Wire Connection of Voltage Sensors

You can use the following three modules for the connection of voltage sensors.

- Analog Input Module **464-8MB11** for voltages of ± 1 V
- Analog Input Module **464-8MC11** for voltages of ± 10 V
- Analog Input Module **466-8MC11** for voltages from 0 to 10 V

Figure 12-4 shows the two-wire connection of voltage sensors.

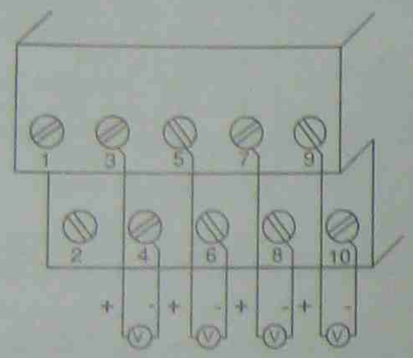


Figure 12-4. Two-Wire Connection of Voltage Sensors (6ES5 464-8MB11, 464-8MC11, 466-8MC11)

12.2.3 Two-Wire Connection of Current Sensors

You can use module **464-8MD11** for the two-wire connection of current sensors.
Figure 12-5 shows the two-wire connections of current sensors.

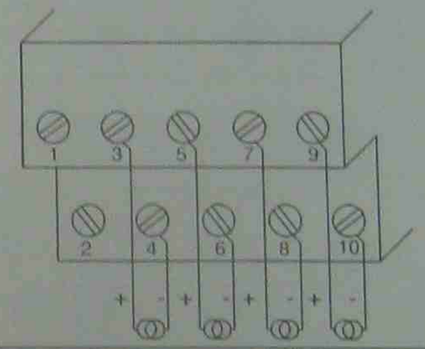


Figure 12-5. Two-Wire Connection for Current Sensors (6ES5 464-8MD11)

12.2.4 Connection of Two-Wire and Four-Wire Transducers

Use the 24-V inputs 1 and 2 of analog input module **464-8ME11** to supply the two-wire transducers.
The two-wire transducer converts the supplied voltage to a current of 4 to 20 mA.
For wiring connections, see Figure 12-6.

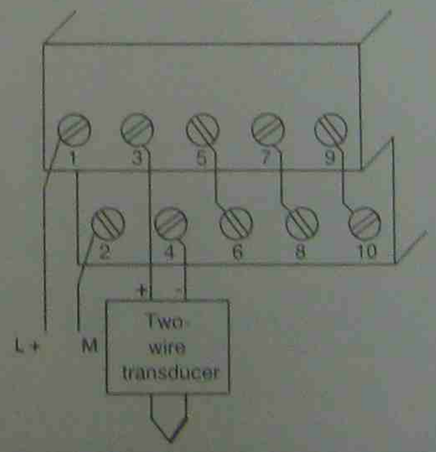


Figure 12-6. Connection of Two-Wire Transducers (6ES5 464-8ME11)

12

If you use a four-wire transducer connect it as shown in Figure 12-7.

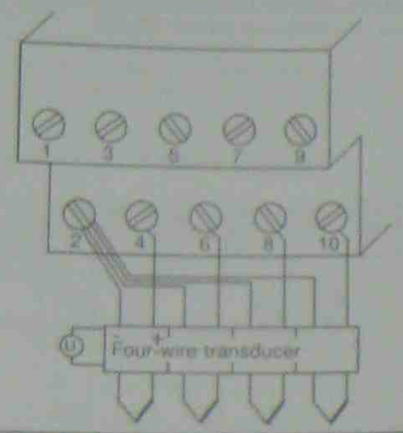


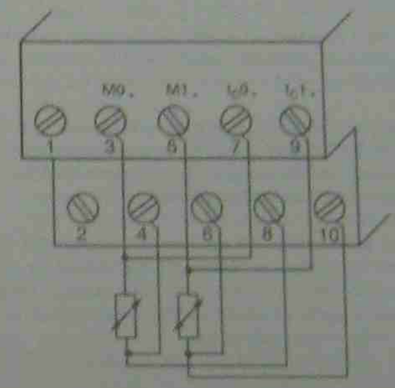
Figure 12-7. Connection for Four-Wire Transducers (6ES5 464-8ME11)

Four-wire transducers require their own power supply. Connect the "+" pole of the four-wire transducer to the corresponding "-" pole of the terminal block (a connection technique that is the opposite of the two-wire transducer). Connect negative terminals of the four-wire transducer to terminal two of the terminal block.

Inputs 4, 6, 8, and 10 of the analog input module **464-8ME11** are connected internally via shunt resistors. Because of the internal shunt resistors, broken wire signaling is not possible.

12.2.5 Connection of Resistance Thermometers

Analog input module **464-8MF11/8MF21** is suited for the connection of resistance thermometers (e.g., PT 100).
The resistance of the PT 100 is measured in a four-wire circuit. A constant current is supplied to the resistance thermometer via terminals 7 and 8 as well as via terminals 9 and 10, so that voltage drops in these "constant current circuits" do not affect the measurement results. The measuring inputs have a high resistance so that only a negligible current loss develops in the measuring circuits.



Terminal assignments:

(3/4):	Measuring circuit M0
(5/6):	Measuring circuits M1
(7/8):	Constant current circuit I_C 0
(9/10):	Constant current circuit I_C 1

Figure 12-8. Wiring Method for PT 100 (6ES5 464-8MF11/8MF21)

If you use only one channel for PT 100 measurement (e.g., channel 0), then you can use the other channel for voltage measurement (± 500 mV). In this case, use terminals M +/M- for the signal connection and short circuit the terminals I_{C+} and I_{C-}.

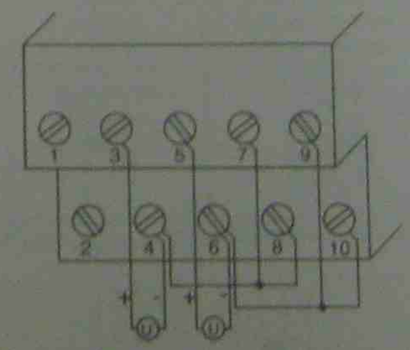


Figure 12-9. Wiring Possibilities for Input Modules (6ES5 464-8MF11)

12

12.3 Start-Up of Analog Input Modules

Set the intended operating mode using the switches on the front panel of analog input modules 464-8 through 11. These switches are located on the right side at the top of the front panel of the module.

Power supply frequency: Set the switch to the available power supply frequency. This selects the integration time of the A/D converters for optimal interference voltage suppression.
Power frequency 50 Hz: Integration time 20 ms
Power frequency 60 Hz: Integration time 16.66 ms

Operation: Set the number of channels you wish to assign on the input module. If there are fewer than four channels, less address space will be assigned and measured values will be updated faster.

Broken wire: Once the broken wire signal has been activated, a break on one of the lines to the sensor (thermocouple or PT 100) or of the sensor itself causes the red LED above the function selection switch to light up. At the same time, the broken wire error bit F (bit 1, byte 1) for the faulty channel is set.

The module "recognizes" a wire break by applying a conventional tripping current to the input terminals and by comparing the resulting voltage to a limit value. If there is a wire break in the sensor or the lines, the voltage exceeds the limit value and a "wire break" signal is sent. When the signal at the input is measured with a digital voltmeter, the tripping current pulses cause apparent fluctuations of the signal. Deactivation of the wire break signal does **not** turn off the tripping current.

Table 12-4. Settings for the Operating Mode Switch for Analog Input Modules 464-8 to11

Function	Settings for Operating Mode Switch	
Power supply frequency	50 Hz 	60 Hz 
Operation	1 channel (channel 0) 	2 channels (channel 0 and channel 1) 
	4 channels (channel 0 - channel 3) 	
Wire break	With wire break signal 	No wire break signal 

EWA 4NEB 812 6065-02a

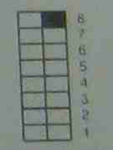
Additional operating mode switch selections possible with analog module **464-8MA21**:

Linearization: With this function, you can obtain a characteristic linearization of the thermocouples of type J, K, and L or of the resistance thermometer PT 100. With module 464-8MA21, the linearization must always be activated together with the corresponding compensation of the reference point temperature.

Thermocouples:
Type J: - 200° C (-328° F) to + 1200° C (1392° F)
Type K: - 200° C (-328° F) to + 1369° C (2497° F)
Type L: - 199° C (-326° F) to + 900° C (1652° F) (in steps each of 1° C (1.8° F)).

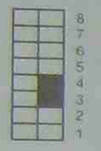
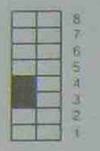
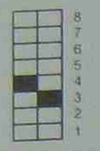
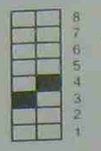
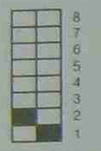
Temperature compensation: For the thermocouples of type J, K, and L, you can compensate, on the one hand, the temperature of the reference point using a compensating box. (See Figure 12-2.)
On the other hand, it is possible to move the reference point to the front of the module by activating the "temperature compensation" function. When thermocouples are directly connected, an internal circuit on the module causes the digital value "0" to be displayed independently of the temperature of the terminal when the temperature at the measuring junction is 0° C (32° F). In order to accomplish this, the terminals of the sensors have to be connected directly to the module, i.e., without a copper extension cable.

Table 12-5. Settings for the Operating Mode Switch for Analog Input Module 464-8MA21

Function	Settings for Operating Mode Switch	
Power supply frequency	50 Hz 	60 Hz 
Operation	1 channel (channel 0) 	2 channels (channel 0 and channel 1) 
	4 channels (channel 0 - channel 3) 	
Wire break	With wire break signal 	No wire break signal 

EWA 4NEB 812 6065-02a

Table 12-5. Settings for the Operating Mode Switch for Analog Input Module 464-8MA21 (continued)

Function	Settings for Operating Mode Switch			
Characteristic linearization of the thermocouples	No linearization 	Linearization type K 	Linearization type J 	Linearization type L 
Temperature compensation	No temperature compensation 	Temperature compensation for type K 	Temperature compensation for types J and L 	

If you have set "Characteristic linearization" and "Temperature compensation" with the operating mode switches on module **464-8MA21** for the thermocouple used, then the reference temperature is 0° C (32° F). This means that with 0° C (32° F) at the measuring junction, the value "0" is displayed.

If you equip several channels with thermocouples, use the same type of thermocouple. If you select mixed thermocouples, or if you use thermocouples other than type J, K, or L, then you must choose the following settings.

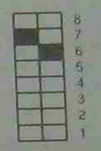
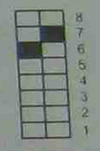
- "No linearization"
- "No temperature compensation"

Compensation is then not possible even with a compensating box because the compensating box is designed only for a certain type of thermocouple. It is possible to use a thermostat in the terminal box if you use the thermostat temperature in the application software to adjust the thermocouple input offset.

When you set the switches to "no linearization" and "no temperature compensation", then module 464-8MA21 functions just like module 464-8MA11.

Set the switches on analog module **464-8MF21** as illustrated in Table 12-6.

Table 12-6. Settings for the Operating Mode Switch for Analog Input Module 464-8MF21

Function	Settings for Operating Mode Switch	
Power supply frequency	50 Hz 	60 Hz 
Operation	1 channel (channel 0) 	2 channel (channel 0 channel 1) 
Wire break	With wire break signal 	No wire break signal 
Characteristic linearization for the PT 100	No linearization 	Linearization for PT 100 

Position 1 and 2 on the operating mode switch have no function.

If you set the switch to "no linearization" and "no temperature compensation", module 464-8MF21 functions just like module 464-8MF11.

The characteristic linearization is possible for the following temperature ranges.
PT 100: -100° C (-148° F) to +850° C (1562° F) (in steps of 0.5° C (0.9° F))

12

12.4 Analog Value Representation of Analog Input Modules

Each analog process signal has to be converted into a digital format, to be stored in the process image input table (PII). The analog signals are converted into a binary digit that is written in one of the following ways.

- In one byte (466-8MA11)
- In two bytes (the remaining analog input modules)

Each bit position has a fixed value in powers of two (see Tables 12-7 and 12-8). Analog values are represented in two's complement.
The following tables show the analog value representations of the different analog inputs in 2-byte format.

Table 12-7. Representation of an Analog Input Value as Bit Pattern

	High Byte								Low Byte							
Bit Number	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Analog Value Represent.	S	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	X	E	OV

Key:
- S Sign bit — 0 = "+", 1 = "-"
- X Irrelevant bits
- E Error bit — 0 = no wire break; 1 = wire break
- OV Overflow bit — 0 = Measured value 4095 units at the most; 1 = Measured value greater than or equal to 4096 units

Table 12-8. Analog Input Module 464-8MA11, -8MF11, -8MB11 (Bipolar Fixed-Point Number)

Units	Measured Value in mV			High Byte								Low Byte								Range
	①	②	③																	
>4095	100.0	1000.0	2000.0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	Overflow
4095	99.976	999.75	1999.5	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	Overrange
2049	50.024	500.24	1000.48	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	
2048	50.0	500.0	1000.0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1024	25.0	250.0	500.0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	0.024	0.24	0.48	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	
0	0.0	0.0	0.0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Nominal range
-1	- 0.024	-0.24	-0.48	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	
-1024	- 25.0	- 250.0	- 500.0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
-2048	-50.0	-500.0	-1000.0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
-2049	-50.024	-500.24	-1000.48	1	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	Overrange
-4095	-99.976	-999.75	-1999.5	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	
<-4095	-100.0	-1000.0	-2000.0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	Overflow

① 464-8MA11/-8MA21 "No linearization" (4x ±50 mV)
② 464-8MF11 (2x ±500 mV)
③ 464-8MB11 (4x ±1 V)

Table 12-9. Analog Input Module 464-8MC11, -8MD11 (Bipolar Fixed-Point Number)

Units	Measured Value in V ①	Measured Value in mA ②	High Byte								Low Byte								Range
>4095	20.000	40.0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	Overflow
4095	19.995	39.9902	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	Overrange
2049	10.0048	20.0098	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	
2048	10.000	20.0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1024	5.000	10.0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	0.0048	0.0098	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	
0	0.0	0.0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Nominal range
-1	-0.0048	-0.0098	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	
-1024	- 5.000	-10.0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
-2048	-10.000	-20.0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
-2049	-10.0048	-20.0098	1	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	Overrange
-4095	-19.995	- 39.9902	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	
<-4095	-20.000	-40.0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	Overflow

① 464-8MC11 (4x ±10 V)
② 464-8MD11 (4x ±20 mA)

Table 12-10. Analog Input Module 464-8ME11, 4x4 to 20 mA (Absolute Value)

Units	Measured Value In mA	High Byte								Low Byte								Range*
>4095	> 32.769	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	Overflow
4095	31.992	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	Overrange
2561	20.008	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	0	
2560	20.0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	
2048	16.0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Nominal range
512	4.0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	
511	3.992	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	
384	3.0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	
0	0.0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Transducer failure?
-1	-0.008	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	
<-4095	<-32.769	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	

* Because of tolerances of components used in the module, the converted value can also be negative (e.g. $FFF8_H$ → Unit: -1).

12

**Table 12-11. Analog Input Module 464-8MF11, 2x PT 100 (Unipolar)
Analog Input Module 464-8MF21, 2x PT 100 "No Linearization" (Unipolar)**

Units	Resistance in Ω	High Byte	Low Byte	Range
>4095	≥400.0	0 1 1 1 1 1 1 1	1 1 1 1 1 0 0 1	Overflow
4095	399.90	0 1 1 1 1 1 1 1	1 1 1 1 1 0 0 0	Overrange
2049	200.098	0 1 0 0 0 0 0 0	0 0 0 0 1 0 0 0	
2048	200.0	0 1 0 0 0 0 0 0	0 0 0 0 0 0 0 0	Nominal range
1024	100.0	0 0 1 0 0 0 0 0	0 0 0 0 0 0 0 0	
1	0.098	0 0 0 0 0 0 0 0	0 0 0 0 1 0 0 0	
0	0.0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	

Table 12-12. Analog Input Module 464-8MF21, 2x PT 100 "with linearization" (Bipolar)

Units	Resis-tance in Ω	Temperature in ° C (° F)	High Byte	Low Byte	Range
>1766	>400	>883 (>1531)	0 0 1 1 0 1 1 1	0 0 1 1 0 0 0 1	Overflow
1766		883 (1531)	0 0 1 1 0 1 1 1	0 0 1 1 0 0 0 1	Overrange*
1702		851(1564)	0 0 1 1 0 1 0 1	0 0 1 1 0 0 0 1	
1700	390.26	850 (1562)	0 0 1 1 0 1 0 1	0 0 1 0 0 0 0 0	
1400	345.13	700(1292)	0 0 1 0 1 0 1 1	1 1 0 0 0 0 0 0	
1000	280.90	500 (932)	0 0 0 1 1 1 1 1	0 1 0 0 0 0 0 0	
600	212.02	300 (572)	0 0 0 1 0 0 1 0	1 1 0 0 0 0 0 0	
300	157.31	150 (302)	0 0 0 0 1 0 0 1	0 1 1 0 0 0 0 0	
200	138.50	100 (212)	0 0 0 0 0 1 1 0	0 1 0 0 0 0 0 0	Nominal range
2	100.39	1 (34)	0 0 0 0 0 0 0 0	0 0 0 1 0 0 0 0	
0	100.00	0 (32)	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	
-40	92.16	-20 (-4)	1 1 1 1 1 1 1 0	1 1 0 0 0 0 0 0	
-80	84.27	-40 (-40)	1 1 1 1 1 1 0 1	1 0 0 0 0 0 0 0	
-200	60.25	-100 (-148)	1 1 1 1 1 1 0 0 1	1 1 0 0 0 0 0 0	
-202		-101 (-150)	1 1 1 1 1 1 0 0 1	1 0 1 1 0 0 0 1	Overrange*
-494		-247 (-413)	1 1 1 1 1 0 0 0 0	1 0 0 1 0 0 0 1	
< -494		< -247 (-413)	1 1 1 1 1 0 0 0 0	1 0 0 1 0 0 0 1	Overflow

* In the overrange the current slope of the characteristic curve is maintained when leaving the linearized nominal range.

Table 12-13. Analog Input Module 464-8MA21, 4x ± 50 mV with Linearization and with Temperature Compensation (Bipolar); Thermoelement Type K (Nickel-Chromium/Nickel-Aluminium, according to IEC 584)

Units	Thermal Voltage in mV*	Temperature in ° C (° F)	High Byte	Low Byte	Range
>2359			0 1 0 0 1 0 0 1	1 0 1 1 1 0 0 1	Overflow
					Overrange**
1370		1370 (2498)	0 0 1 0 1 0 1 0	1 1 0 1 0 0 0 1	
1369	54.773	1369 (2496)	0 0 1 0 1 0 1 0	1 1 0 0 1 0 0 0	
1000	41.269	1000 (1832)	0 0 0 1 1 1 1 1	0 1 0 0 0 0 0 0	
500	20.640	500 (932)	0 0 0 0 1 1 1 1	1 0 1 0 0 0 0 0	
150	6.137	150 (302)	0 0 0 0 0 1 0 0	1 0 1 1 0 0 0 0	
100	4.095	100 (212)	0 0 0 0 0 0 1 1	0 0 1 0 0 0 0 0	Nominal range
1	0.039	1 (34)	0 0 0 0 0 0 0 0	0 0 0 0 1 0 0 0	
0	0	0 (32)	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	
-1	-0.039	-1(-30)	1 1 1 1 1 1 1 1	1 1 1 1 1 0 0 0	
-100	-3.553	-100 (-148)	1 1 1 1 1 1 0 0	1 1 1 0 0 0 0 0	
-101	-3.584	-101 (-150)	1 1 1 1 1 1 0 0	1 1 1 0 0 0 0 0	Accuracy ≤2 K
-150	-4.912	-150 (-238)	1 1 1 1 1 0 1 1	0 1 0 1 0 0 0 0	
-200	-5.891	-200 (-328)	1 1 1 1 1 0 0 1	1 1 0 0 0 0 0 0	
-201		-201 (-330)	1 1 1 1 1 0 0 1	1 0 1 1 1 0 0 1	Overrange**
-273			1 1 1 1 0 0 0 0	1 0 0 1 0 0 0 1	Overflow
X		X	X X X X X X X X	X X X X X 0 1 0	Wire break

This value corresponds to the terminal temperature at wire break

* For a reference temperature of 0° C (32° F)

** In the overrange, the current slope of the characteristic curve is maintained when leaving the linearized nominal range.

12

Table 12-14. Analog Input Module 464-8MA21, 4x ± 50 mV with Linearization and with Temperature Compensation (Bipolar); Thermoelement Type J (Iron/Copper-Nickel (Konstantan), according to IEC 584)

Units	Thermal Voltage in mV*	Temperature in ° C (° F)	High Byte	Low Byte	Range
1485			0 0 1 0 1 1 1 0	0 1 1 0 1 0 0 1	Overflow
1201		1201 (2194)	0 0 1 0 0 1 0 1	1 0 0 0 1 0 0 1	Overrange**
1200	69.536	1200 (2192)	0 0 1 0 0 1 0 1	1 0 0 0 0 0 0 0	Nominal range
1000	57.942	1000 (1832)	0 0 0 1 1 1 1 1	0 1 0 0 0 0 0 0	
500	27.388	500 (932)	0 0 0 0 1 1 1 1	1 0 1 0 0 0 0 0	
100	5.268	100 (212)	0 0 0 0 0 0 1 1	0 0 1 0 0 0 0 0	
1	0.05	1 (34)	0 0 0 0 0 0 0 0	0 0 0 0 1 0 0 0	
0	0	0 (32)	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	
-1	-0.05	-1 (-30)	1 1 1 1 1 1 1 1	1 1 1 1 1 0 0 0	
-100	-4.632	-100 (-148)	1 1 1 1 1 1 0 0	1 1 1 0 0 0 0 0	
-150	-6.499	-150 (-238)	1 1 1 1 1 0 1 1	0 1 0 1 0 0 0 0	
-199	-7.868	-199 (-326)	1 1 1 1 1 0 0 1	1 1 0 0 1 0 0 0	
-200	-7.890	-200 (-328)	1 1 1 1 1 0 0 1	1 1 0 0 0 0 0 0	
-201		-201 (-330)	1 1 1 1 1 0 0 1	1 0 1 1 1 0 0 1	Overrange**
-273			1 1 1 1 1 0 1 1	0 1 1 1 1 0 0 1	Overflow
X		X	X X X X X X X X	X X X X X 0 F 0	Wire break

This value corresponds to the terminal temperature at wire break

* For a reference temperature of 0° C (32° F)

** In the overrange, area the current slope of the characteristic curves is maintained when leaving the linearized nominal range.

Table 12-15. Analog Input Module 464-8MA21, 4x ± 50 mV with Linearization and with Temperature Compensation (Bipolar); Thermoelement Type L (Iron/Copper-Nickel (Konstantan), according to DIN 43710)

Units	Thermal Voltage in mV*	Temperature in ° C (° F)	High-Byte	Low-Byte	Range
1361			0 0 1 0 1 0 1 0	1 0 0 0 1 0 0 1	Overflow
901		901 (1654)	0 0 0 1 1 1 0 0	0 0 1 0 1 0 0 1	Overrange**
900	53.14	900 (1652)	0 0 0 1 1 1 0 0	0 0 1 0 0 0 0 0	Nominal range
500	27.85	500 (932)	0 0 0 0 1 1 1 1	1 0 1 0 0 0 0 0	
250	13.75	250 (482)	0 0 0 0 0 1 1 1	1 1 0 1 0 0 0 0	
100	+5.37	100 (212)	0 0 0 0 0 0 1 1	0 0 1 0 0 0 0 0	
1	0.05	1 (34)	0 0 0 0 0 0 0 0	0 0 0 0 1 0 0 0	
0	0	0 (32)	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	
-1	-0.05	-1 (-30)	1 1 1 1 1 1 1 1	1 1 1 1 1 0 0 0	
-100	-4.75	-100 (-148)	1 1 1 1 1 1 0 0	1 1 1 0 0 0 0 0	
-150	-6.60	-150 (-238)	1 1 1 1 1 0 1 1	0 1 0 1 0 0 0 0	
-190	-7.86	-190 (-310)	1 1 1 1 1 0 1 0	0 0 0 1 0 0 0 0	
-199	-8.12	-199 (-326)	1 1 1 1 1 0 0 1	1 1 0 0 1 0 0 0	
-200		-200 (-328)	1 1 1 1 1 0 0 1	1 1 0 0 0 0 0 1	Overrange**
-273			1 1 1 1 0 1 1 1	0 1 1 1 1 0 0 1	Overflow
X		X	X X X X X X X X	X X X X X 0 1 0	Wire break

This value corresponds to the terminal temperature at wire break

* For a reference temperature of 0° C (32° F)

** In the overrange, the current slope of the characteristic curve is maintained when leaving the linearized nominal range.

The **466-8MC11** analog input module stores each analog value in a single byte. The other analog input modules store the analog values in words (see Table 12-7).

Table 12-16. Analog Input Module 466-8MC11, 4x 0 to 10

Units	Voltage in mV	Bit Representation
255	≥ 9961	1 1 1 1 1 1 1 1
254	9922	1 1 1 1 1 1 1 0
.	.	.
128	5000	1 0 0 0 0 0 0 0
.	.	.
1	39	0 0 0 0 0 0 0 1
0	0	0 0 0 0 0 0 0 0

If you want to read in the analog value with function block FB250 (analog value reading and scaling), you have to pre-process the analog value before calling up FB250.

Example 1:

Analog input module 466-8MC11 is inserted in slot 1, which means that the module's start address is 72.
The analog values are stored in four consecutive bytes.

1st analog value (channel 0) → in IB72
2nd analog value (channel 1) → in IB73
3rd analog value (channel 2) → in IB74
4th analog value (channel 3) → in IB75.

Function block FB72, pictured below, reads in the analog values and pre-processes them for Function block FB250 (analog value reading and scaling).

FB72				Explanation
NAME :READ 466				
0005	:			READ IN ALL CHANNELS
0006	:L	IW	72	OF AI 466
0007	:T	FW	72	READ ALL FOUR CHANNELS
0008	:L	IW	74	AND REARRANGE
0009	:T	FW	74	
000A	:			
000B	:L	FY	72	PROCESS EACH ANALOG VALUE
000C	:SLW		3	AND REWRITE THEM IN
000D	:T	IW	72	THE PII SO THAT FB250
000E	:			CAN ACCESS THEM
000F	:L	FY	73	WITHIN THAT SCAN.
0010	:SLW		3	
0011	:T	IW	74	
0012	:			
0013	:L	FY	74	
0014	:SLW		3	
0015	:T	IW	76	
0016	:			
0017	:L	FY	75	
0018	:SLW		3	
0019	:T	IW	78	
001A	:			
001B	:BE			

Example 2:

Analog input module 466-8MC11 is inserted in slot 0, which means that the module's start address is 64.

The analog values that are read in are stored in four consecutive bytes:

1st analog value (channel 0) → in IB64
2nd analog value (channel 1) → in IB65
3rd analog value (channel 2) → in IB66
4th analog value (channel 3) → in IB67

Function block 73, pictured below, reads in the analog values and pre-processes them for FB250. The additional processing with FB250 is done just like module 464, however without an overflow bit.

FB73				Explanation
NAME :READ AI				
0005	:			
0006	:			
0007	:L	IB	67	Read in Channel 3
0008	:SLW		3	
0009	:T	IW	70	
000A	:			
000B	:L	IB	66	Read in Channel 2
000C	:SLW		3	
000D	:T	IW	68	
000E	:			
000F	:L	IB	65	Read in Channel 1
0010	:SLW		3	
0011	:T	IW	66	
0012	:			
0013	:L	IB	64	Read in Channel 0
0014	:SLW		3	
0015	:T	IW	64	
0016	:			
0017	:			

12.5 Analog Output Modules

Analog output modules convert the bit patterns that are output by the CPU into analog output voltages or currents.

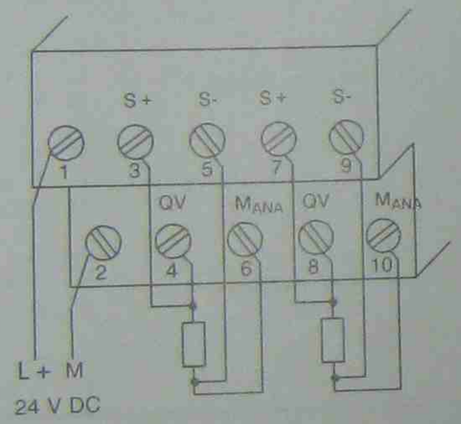
12.5.1 Connection of Loads to Analog Output Modules

No adjustments are necessary if you want to connect loads to the analog outputs.

Check the following items before connecting loads.
- The load voltage 24 V DC must be connected to terminals 1 and 2.
- The maximum permissible potential difference between the outputs is 60 V AC.
- Unused outputs must be left open-circuited.

Figure 12-10 shows how to connect loads to the voltage outputs of the following modules.
- 470-8MA11 (2x ± 10 V)
- 470-8MD11 (2x + 1 to 5 V)

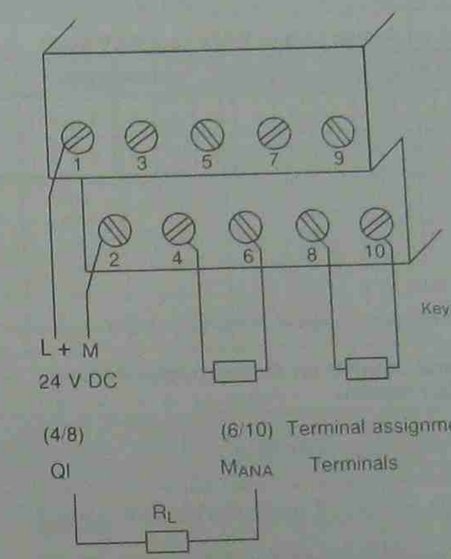
The sensor lines (S+ and S-) must be directly connected to the load, so that the voltage is measured and regulated directly at the load. In this manner, voltage drops of up to 3 V per line can be compensated for.
The sensor lines can be left out if the resistances of the QV and M lines are negligible compared to the load resistance. In such a case, connect terminal S + to terminal QV, and terminal S to M_{ANA}.



Key:
OV: Analog output "Voltage"
S±: Sensor line
M_{ANA}: Chassis ground terminal of the analog unit
R_L: Load resistor

Figure 12-10. Load Connection via a Four-Wire Circuit (6ES5 470-8MA11 or 6ES5 470-8MD11)

Figure 12-11 shows how to connect loads to the current outputs of the following modules.
- 470-8MB11 (2x ± 20 mA)
- 470-8MC11 (2x + 4 to 20 mA).



Key:
QI: Analog output "Current"
M_{ANA}: Chassis ground terminal of the analog unit

Figure 12-11. Connection via a Two-Wire Circuit (6ES5 470-8MB11 or 6ES5 470-8MC11)

12.5.2 Analog Value Representation of Analog Output Modules

Table 12-17 shows how the analog output value has to be stored in the process image output table (PIQ).

Table 12-17. Representation of an Analog Output Value as a Bit Pattern

	High Byte								Low Byte							
Bit number	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Analog value represent.	S	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	X	X	X	X

Key: S sign bit
X irrelevant bits

Tables 12-18 and 12-19 show the voltage and currents assigned to the bit patterns.

Table 12-18. Output Voltages and Currents for Analog Output Modules (Fixed-Point Number Bipolar)

Units	Output Values in V	Output Values in mA	High Byte	Low Byte	Range
1280	12.5	25.0	0 1 0 1 0 0 0 0	0 0 0 0 x x x x	Overrange
1025	10.0098	20.0195	0 1 0 0 0 0 0 0	0 0 0 0 x x x x	
1024	10.0	20.0	0 1 0 0 0 0 0 0	0 0 0 0 x x x x	Nominal range
512	5.0	10.0	0 0 1 0 0 0 0 0	0 0 0 0 x x x x	
1	0.0098	0.0195	0 0 0 0 0 0 0 0	0 0 0 0 x x x x	
0	0.0	0.0	0 0 0 0 0 0 0 0	1 1 1 1 x x x x	
-1	-0.0098	-0.0195	1 1 1 1 1 1 1 1	0 0 0 0 x x x x	
-512	-5.0	-10.0	1 1 1 0 0 0 0 0	0 0 0 0 x x x x	
-1024	-10.0	-20.0	1 1 0 0 0 0 0 0	0 0 0 0 x x x x	
-1025	-10.0098	-20.0195	1 0 1 1 1 1 1 1	1 1 1 1 x x x x	Overrange
-1280	-12.5	-25.0	1 0 1 1 0 0 0 0	0 0 0 0 x x x x	
	①	②			

① 2x ± 10 V 6ES5 470-8MA11
② 2x ± 20 mA 6ES5 470-8MB11

Table 12-19. Output Voltages and Currents for Analog Output Modules (Unipolar)

Units	Output Values in V	Output Values in mA	High-Byte	Low-Byte	Range
1280	6.0	24.0	0 1 0 1 0 0 0 0	0 0 0 0 x x x x	Overflow
1025	5.004	20.016	0 1 0 0 0 0 0 0	0 0 0 1 x x x x	
1024	5.0	20.0	0 1 0 0 0 0 0 0	0 0 0 0 x x x x	Nominal range
512	3.0	12.0	0 0 1 0 0 0 0 0	0 0 0 0 x x x x	
1	1.004	4.016	0 0 0 0 0 0 0 0	0 0 0 1 x x x x	
0	1.0	4.0	0 0 0 0 0 0 0 0	0 0 0 0 x x x x	
-1	0.996	3.984	1 1 1 1 1 1 1 1	1 1 1 1 x x x x	Overrange
-256	0.0	0.0	1 1 1 1 0 0 0 0	0 0 0 0 x x x x	
-512	-1.0	-4.0	1 1 0 0 0 0 0 0	0 0 0 0 x x x x	
-1024	-3.0	-12.0	1 1 0 0 0 0 0 0	0 0 0 0 x x x x	
-1280	-4.0	-16.0	1 0 1 1 0 0 0 0	0 0 0 0 x x x x	
	①	②			

① 2x 1 to 5 V 6ES5 470-8MD11
② 2x 4 to 20 mA 6ES5 470-8MC11

12.6 Analog Value-Interface Function Blocks FB250 and FB251

12.6.1 Reading in and Scaling an Analog Value - FB250 -

Function block FB250 reads in an analog value from an analog input module and outputs a value XA in the scale range specified by the user.

Specify the type of analog value representation for the module (channel type) in the KNKT parameter (see Table 12-20). Define the desired range using the "upper limit" (OGR) and "lower limit" (UGR) parameters.

12

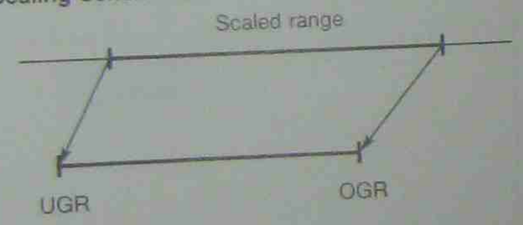
STL
: JU FB 250
NAME : RLG:AI
BG :
KNKT :
OGR :
UGR :
EINZ :
XA :
FB :
BU :

Figure 12-12. Call Up of FB250

Table 12-20. Call and Parameter Assignments of FB250

Parameter	Explanation	Type	Assignment	
			External I/Os	Onboard I/Os
BG	Slot number	D KF	0 to 7	8
KNKT	Channel number Channel type	D KY	KY = x,y x = 0 to 3 y = 3 to 6 3: Absolute value representation (4 to 20 mA) 4: Unipolar representation 5: Bipolar absolute value 6: Bipolar fixed-point number	KY = x,y x = 0 to 7 y = 4 Unipolar representation
OGR	Upper limit of the output value	D KF	-32767 to +32767	
UGR	Lower limit of the output value	D KF	-32767 to +32767	
EINZ	Single scan	I BI	Not relevant	Single scan is triggered with signal status "1"
XA	Output value	Q W	Scaled analog value is "0" on wirebreak	Scaled analog value
FB	Error bit	Q BI	"1" on wirebreak, illegal channel or slot number or illegal channel type	"1" on illegal channel or slot number or illegal channel type
BU	Range violation	Q BI	"1" when nominal range is exceeded	

Scaling Schematic



Representation of the analog input module

User scaled range

Figure 12-13. Scaling Schematic for FB250

12.6.2 Outputting Analog Values -FB251-

This function block allows analog values to be output to analog output modules. In doing so, values from the range between the "UGR" (lower limit) parameters and the "OGR" (upper limit) parameters are converted to the nominal range of the appropriate module.

STL
: JU FB 251 NAME : RLG:AQ XE : BG : KNKT : OGR : UGR : FEH : BU :

Figure 12.14. Call Up of FB251

Table 12-21. Parameter Assignment of FB251

Parameter	Explanation	Type	Assignment	
			External I/Os	Onboard I/Os
XE	Analog value to be output	I W	Input value (two's complement) in the range UGR to OGR	
BG	Slot address	D KF	0 to 7	8
KNKT	Channel number Channel type	D KY	KY = x,y x = 0;1 y = 0;1 0: unipolar representation 1: bipolar fixed-point	KY = x,y x = 0 y = 0 unipolar representation
OGR	Upper limit of the output value	D KF	-32767 to +32767	
UGR	Lower limit of the output value	D KF	-32767 to +32767	
FEH	Error in limit value setting	Q BI	"1" if UGR = OGR, invalid channel or slot number, or invalid channel type	
BU	Input value exceeds UGR or OGR	Q BI	"1" if XE lies outside limits (UGR; OGR). XE assumes the limit value	

12.6.3 Example: Analog Value Processing with FB250 and FB251

There are eight analog inputs and one analog output on the S5-95U. The inputs are preset at the factory so that if no parameters are entered, the inputs can only be read in via a direct I/O access (L PW). If you want to read in the inputs cyclically into the PII, you must first enter parameters in DB1 (see section 9.1).

Example: You can use a fan to help regulate the temperature in a room. The temperature should fluctuate between 20 and 28 degrees Celsius (68 degrees Fahrenheit and 82 degrees Fahrenheit). The fan speed depends on the room temperature. You can input the room temperature into FB250. You can output the setpoint value for the fan speed in FB251.

How to proceed:

- Wire the analog connections (see section 12.2.1).
- Enter the analog input parameters in default DB1.
  - Display default DB1.
  - Change parameter block OBA: _ AI_0_:_ as indicated in Table 12-22.
  - Transfer the changed DB1 to the programmable controller.
  - Switch the programmable controller from "STOP" to "RUN".
- Enter the parameter settings for FB250 and FB251.

Table 12-22. Changing Default Parameter for Analog Inputs

Setting parameters for analog input	Explanation
`0:  KS  = DB1 OBA: AI 1 : DBI:  '`	The first analog input (channel 0) is read in cyclically.

Note

If you replace "AI 1" with "AI 3", the first three analog inputs (Channel 0 through 2) are read in cyclically. We recommend that you connect the analog inputs in ascending order. Do not enter parameters for any more analog inputs than you need.

Reading in Analog Values (FB250)

In order to read in the analog value, call up FB250 on the programmer and enter its parameters as they are shown in the example (see Table 12-23). Figure 12-15 shows you how the function block operates. There is an explanation of the individual parameters in Table 12-24. When FB 250 is called, it reads in the analog values of analog input channel 0 and sends an XA value to the output in the range specified by the user (see Figure 12-15).

Table 12-23. Example for Setting Parameters in FB250

STL	Explanation

`: JU FB 250`
`NAME : RLG:AI`
`BG : KF=+8`
`KNKT KY=0,4`
`OGR : KF=+280`
`UGR : KF=+200`
`EINZ : F 50.0`
`XA : FW 130`
`FB : F 120.0`
`BU : F 121.0`
	In the temperature range from 20 to 28 degrees Celsius, the analog input (channel number 0) delivers a nominal value from 0 to 10 V (channel type 4). If you assign the value KF = +200 to parameter UGR (lower limit) and the value +280 to the OGR (upper limit), the function block gives the temperature value in 1/10 degrees at the XA output. The value is stored as a fixed-point number in FW 130.

FB250 transforms the range 0 to 10 V into the range 20 to 28° C.

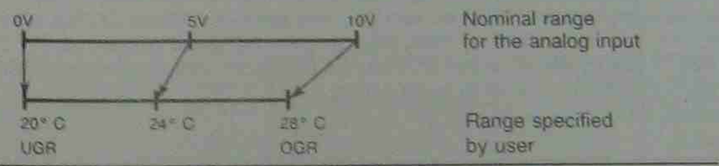


Figure 12-15. Transforming the Nominal Range to a Specified Range

Table 12-24. Parameters for FB250

Parameter	Explanation	Type	Assignment
BG	Slot number	D KF	8 (Onboard I/Os)
KNKT	Channel number Channel type	D KY	KY = x,y x = 0 to 7 y = 4 Unipolar representation
OGR	Upper limit of the output value	D KF	-32767 to +32767
UGR	Lower limit of the output value	D KF	-32767 to +32767
EINZ	Single scan	I BI	Single scan is triggered with signal state "1"
XA	Output value	Q W	Scaled analog value
FB	Error bit	Q BI	"1" if an illegal channel number or slot number, or illegal channel type
BU	Range violation	Q BI	"1" if nominal range is exceeded

Outputting an Analog Value (FB251)

In order to output the analog value, call up FB251 on the programmer and assign it parameters as shown in the example (see Table 12-25). Flag word 130 contains the fixed-point number that FB250 calculated. This number is now converted into the RPM setpoint in FB251.

Figure 12-16 shows you how the FB functions. There is an explanation of the individual parameters in Table 12-26.

When FB251 is called it outputs the analog values calculated from the digital values at the analog output channel.

Table 12-25. Entering Parameters for FB251

STL	Explanation
: JU FB 251 NAME : ANA-AG XE : FW130 BG : KF=8 KNKT : KF=0,1 OGR : KF=1500 UGR : KF=120 FEH : F 120.1 BU : F 121.1	In the range of 120 to 1500 RPM a speed setpoint is to be displayed (corresponding to the nominal range of the analog output). If the parameter UGR is set with the value KF = + 120 and OGR is set with the value KF = + 1500, you can enter the speed setpoint into parameter XE in RPMs. Function Block 251 writes the corresponding value (0 V to 10 V) to the analog output.

FB251 transforms the range + 120 to 1500 RPM into the range 0 to 10 V (nominal range of the analog output)

Figure 12-16. Transforming the Specified Range to the Nominal Range

Table 12-26. Parameters for FB251

Parameter	Explanation	Type	Assignment
XE	Analog value to be output	I/W	Input value (2's complement in the UGR to OGR range
BG	Slot number	D KF	8 (for onboard I/Os)
KNKT	Channel number Channel type	D KY	KY = x,y x = 0 (for onboard I/Os) y = 0 (unipolar representation
OGR	Upper limit of the output value	D KF	-32767 to +32767
UGR	Lower limit of the output value	D KF	-32767 to +32767
FEH	Error in the limit value setting	Q BI	"1" if UGR = OGR, for invalid channel or slot number, or invalid channel type
BU	Input value exceeds UGR or OGR	Q BI	"1" if XE is outside the range (UGR, OGR). XE accepts the limit value

EWA 4NEB 812 6065-02a

Figures

Tables

13 Integral Real-Time Clock

(Beginning with version 2 only of the S5-95U)

13.1 Function

The integral real-time clock offers the following possibilities of controlling the process sequence.

- Clock and calendar function
  used to configure clock-time dependent control, for example.
- Prompt and alarm function
  used to monitor the duration of a process, for example.
- Operating hours counter
  used to monitor inspection intervals, for example.

The clock begins running when you supply voltage to the programmable controller. The default is October 1, 91, 12:00 o'clock. You set the clock by setting its parameters. You can set parameters for the clock in DB1 (see section 13.2). If you have an extensive system knowledge, you can also set parameters for the clock in the system data (see section 13.6) and program the clock in the user program (see section 13.7).

The hardware clock requires a clock data area and a status word in order to function. The location of both the clock data area and the status word must be stored in system data 8 to 10.

Operating Principle of the Clock

Data exchange between the integral real-time clock and the control program always goes through the clock data area. The clock stores current values for time, date, and operating hours counter in the clock data area. You can transfer into the clock data area the values for the time, date, prompt time, and operating hours counter that you want the clock to use.

You can scan the status word to identify setting errors, for example. Or you can change certain status word bits to deliberately disable or enable transfer or read operations.

Refer to sections 13.4 and 13.5 for additional information about the clock data area and the status word. These sections are especially important if you want to set clock parameters in the system data. If you are new to SIMATIC, you may prefer to set clock parameters in DB1.

3.2 Setting Parameters in DB1

ou must set the clock parameters in DB1 to be able to use the clock functions. Follow the same ules you used in setting parameters for other functions. Refer to section 9.1.

rocedures for Setting Parameters in DB1

Perform an overall reset.
Output default DB1 to the programmer.
Use the cursor to jump into the clock parameter block.
Change the parameters.
Transfer the changed DB1 to the programmable controller.
Switch the programmable controller from STOP to RUN.

very time there is a change from STOP to RUN, the programmable controller accepts the new lock data.

Note

The system data contents are deleted during an overall reset. The clock continues to run internally with the current values.
The clock time is updated one second after the next cycle starts.

3.2.1 Defaults

he following values are preset in the parameter block when you output the default DB1.

```
 84:    KS  =' 100    : #CLP: STW MW10';
 96:    KS  ='2        CLK DB5 DW0    ';
108:    KS  ='  SET 3  01.10.91 12:00:';
120:    KS  ='00    OHS 000000:00:00  ';
132:    KS  ='  TIS 3 01.10. 12:00:00 ';
144:    KS  ='   STP Y SAV Y  CF 00   ';
156:    KS  =' ; #END ';
```

Figure 13-1. DB1 with the Default Parameters for the Integral Real-Time Clock

fter the CLP block ID for the integral real-time clock, the STW parameter specifies the location of he status word (in flag word MW102, for example). The CLK parameter defines the location of the lock data (in DB5 beginning with DW0, for example). You must specify both parameters if you want o read the clock. Section 13.2.2 describes the procedures you must follow to read the clock.

13.2.2 Reading the Current Clock Time and the Current Date

Proceed as follows to see how and with which values the clock runs.

- Perform an overall reset.
- Output DB1 to the programmer.
- Delete both (#) comment characters before CLP and END.
- Generate DB5 with DW0 to DW21. See Table 13.2 for information about storing the current clock time and current date.
- Switch the programmable controller from STOP to RUN. The clock accepts the values present in DB1.
- Enter DB5 and DW0 to DW3 on the programmer by using the FORCE VAR function.

Table 13-1. Reading the Current Clock Time and Current Date

Operand	Signal States	Explanation
DB 5		
DW 0	KH = 0003	Tuesday
DW 1	KH = 0110	October 1
DW 2	KH = 9112	1991, 12:00
DW 3	KH = 0000	

- Press the "ENTER" key twice. The clock runs using the current values.

13.2.3 DB1 Parameters Used for the Integral Real-Time Clock

Table 13-2. DB1 Parameters for the Integral Real-Time Clock.

Parameters	Argument	Meaning
Block ID: CLP:		**Clock Parameters**
STW	DBxDWy, MYz,EWv or AWv	Location of the status word (*ST*atus *W*ord)
CLK	DBxDWy, MWz,EWv or AWv	Location of the clock data (*CL*oc*K* Data)
SET	wd dd.mm.yy hh:mn:ss[1] AM/PM[2]	Setting the clock time and date
OHS	hhhhhh:mn:ss[1]	Setting the operating hours counter (*O*perating *H*ours counter *S*et)
OHE	J/Y/N	Enabling the operating hours counter (*O*perating *H*ours counter *E*nable)
TIS	wd dd.mm. hh:mn:ss[1] AM/PM[2]	Setting the prompt time (*TI*mer *S*et)
STP	J/Y/N	Updating the clock during STOP (*ST*o*P*)
SAV	J/Y/N	Saving the clock time after the last change from RUN to STOP or Power OFF (*SAV*e)
CF	p	Entering the correction factor (*C*orrection *F*actor)

wd	= 1 to 7	(weekday = Sun. - Sat.)	p	=	-400 to 400
dd	= 01 to 31	(day)	v	=	0 to 126
mm	= 01 to 12	(month)	x	=	2 to 255
yy	= 0 to 99	(year)	y	=	0 to 255
hh	= 00 to 23	(hours)	z	=	0 to 254
mn	= 00 to 59	(minutes)	j/J	=	yes
ss	= 00 to 59	(seconds)	y/Y	=	yes
hhhhhh	= 0 to 999999	(hours)	n/N	=	no

[1] If an argument such as seconds, for example, is not to be entered, enter XX. The clock continues to run with the current data. The TIS parameter block does not acknowledge this argument.

[2] If you enter AM or PM after the clock time, the clock runs in the 12-hour mode. If you omit this argument, the clock runs in the 24-hour mode. You must use the same time mode in the SET and TIS parameter blocks.

13.3 Programming the Integral Real-Time Clock in DB1

Sections 13.3 and 13.4 contain examples for programming the clock. Adhere to the rules described in Chapter 9 for setting parameters when you enter these examples into the programmable controller.

> **Note**
>
> If the programmable controller recognizes a parameter setting error in DB1, the programmable controller remains in the STOP mode even after it has been switched from STOP to RUN. The red LED is lit.

13.3.1 Setting the Clock in DB1

How to set the clock in DB1.

- Perform an overall reset on the programmable controller.
- Generate DB5 with DW0 to DW21.
- Output default DB1 to the programmer.
- Use the cursor to jump into the CLP parameter block.
- Delete both (#) comment characters before CLP and END.
- Enter the example. Set the clock using the following date: Friday, August 11, 1991, 15:30.

Setting the Clock	Explanation

`84: KS =' 100 ; CLP: STW MW10';`
`96: KS ='2 CLK DB5 DW0 ';`
 	
`108: KS =' SET 6 08.11.91 15:30:';`
`120: KS ='00 OHS 000000:00:00 ';`
.
.
.
`144: KS =' STP Y SAV Y CF 00 '`
	The status word is located in flag word MW102. The clock data is stored in data block 5 beginning with data word DW0. After the SET parameter, enter the weekday, the date, and the clock time you want the clock to use when it begins running. Be certain to include the blank spaces. The clock runs in the 24-hour time mode since you do not enter either AM or PM. The clock is updated when the programmable controller is in STOP. The clock time is saved in the clock data area. See Table 13.2.

Figure 13-2. Example - Setting the Clock in DB1

- Transfer the changed DB1 to the programmable controller.
- Switch the programmable controller from STOP to RUN.

Each time the programmable controller is switched from STOP to RUN, it accepts the new clock data.

13.3.2 Setting the Prompt Time in DB1

How to Set the Prompt Time in DB1.

- Perform an overall reset on the programmable controller.
- Generate DB5 with DW0 to DW21.
- Output default DB1 to the programmer.
- Use the cursor to jump to the CLP parameter block.
- Delete both (#) comment characters before CLP and END.
- Enter the example. Set the clock with the following prompt time: Tuesday, 01.07, 8:00 o'clock.

Setting the Prompt Time	Explanation

`84: KS =' 100 ; CLP: STW MW10';`
`96: KS ='2 CLK DB5 DW0 ';`
`.`
`.`
`.`
`132: KS =' TIS 3 07.01 08:00:00 ';`
`144: KS ='AM STP Y SAV Y CF 00 ';`
	The status word is in flag word MW102. The clock data is stored in data block DB5 beginning with data word DW0.
	
	After the TIS parameter, enter the weekday, the data, and the clock time to initiate the prompt time. You can enter the parameter for the clock mode. The clock in the example runs in the 12-hour mode.
	The clock is updated when the programmable controller is in STOP. The clock time is saved in the clock data area. Refer to Table 13.2.

Figure 13-3. Example - Setting the Prompt Time in DB1

- Transfer the changed DB1 to the programmable controller.
- Switch the programmable controller from STOP to RUN.

Each time the programmable controller is switched from STOP to RUN, it accepts the new clock data.

13.3.3 Setting the Operating Hours Counter in DB1

How to set the operating hours counter in DB1

- Perform an overall reset on the programmable controller.
- Generate DB5 with DW0 to DW21.
- Output default DB1 to the programmer.
- Use the cursor to jump to the CLP parameter block.
- Delete both (#) comment characters before CLP and END.
- Enter the example. The programmable controller has been replaced. You set the start value for the operating hours counter at 1600 hours.

Setting the Operating Hours Counter	Explanation

`84: KS =' 100 ; CLP: STW MW10';`
`96: KS ='2 CLK DB5 DW0 ';`
`.`
`.`
`120: KS ='00 OHS 1600:00:00 ';`
`144: KS =' STP Y SAV Y CF 00 ';`
`155: KS =' OHE Y ; END ';`
	The status word is located in flag word MW102. The clock data is stored in data block DB5 beginning with data word DW0.
	
	After the OHS parameter, enter the start value for the operating hours counter.
	The clock is updated when the programmable controller is in STOP. The time is saved in the clock data area. The operating hours counter is enabled.
	OHE is not a default parameter; that is, you must enter it completely.

Figure 13-4. Example - Setting the Operating Hours Counter in DB1

- Transfer the changed DB1 to the programmable controller.
- Switch the programmable controller from STOP to RUN.

Each time the programmable controller is changed from STOP to RUN, it accepts the new clock data.

13.3.4 Entering the Clock Time Correction Factor in DB1

The exactness of the clock is temperature-dependent. You can configure a correction value to increase the clock's exactness. The correction value is output in s/month. You must measure how many seconds per month the clock runs fast or slow. A month is defined as 30 days.

Example: Your measurements indicate the clock is 12 s too slow in four days. That would be 90 s too slow in 30 days. The correction value is +90 s/month.

- In addition to the changed clock parameters, enter the example into DB1 as follows:

Enter the Clock Time Correction Factor	Explanation
`158: KS =' SIP Y SAV Y CF +90 ';`	The correction value of +90 s is loaded into the clock.

Figure 13-5. Example - Entering a Correction Factor in DB1

13.4 Structure of the Clock Data Area

You need only to change the default values in DB1 to program the clock in DB1. See section 13.2. During start-up, the DB1 interpreter writes all information into the system data area.

TIP: Do not attempt to set parameters in the system data, or to access directly from the user program unless you have extensive knowledge of the system.

You must store the location of the clock data area in system data 8 and 9.

Data exchange between DB1 or the control program and the integral real-time clock is always through the clock data area.

- The integral real-time clock stores current time, date, and operating hours counter values in the clock data area (flag area, data block, input area, or output area).
- DB1 and the control program store the settings for prompt times and operating hours counters in the same data area.

The control program can read or write only the clock data area. The control program can never access the clock directly. Figure 13.6 illustrates the relationship between DB1 or the control program, the clock data area, and the integral real-time clock.

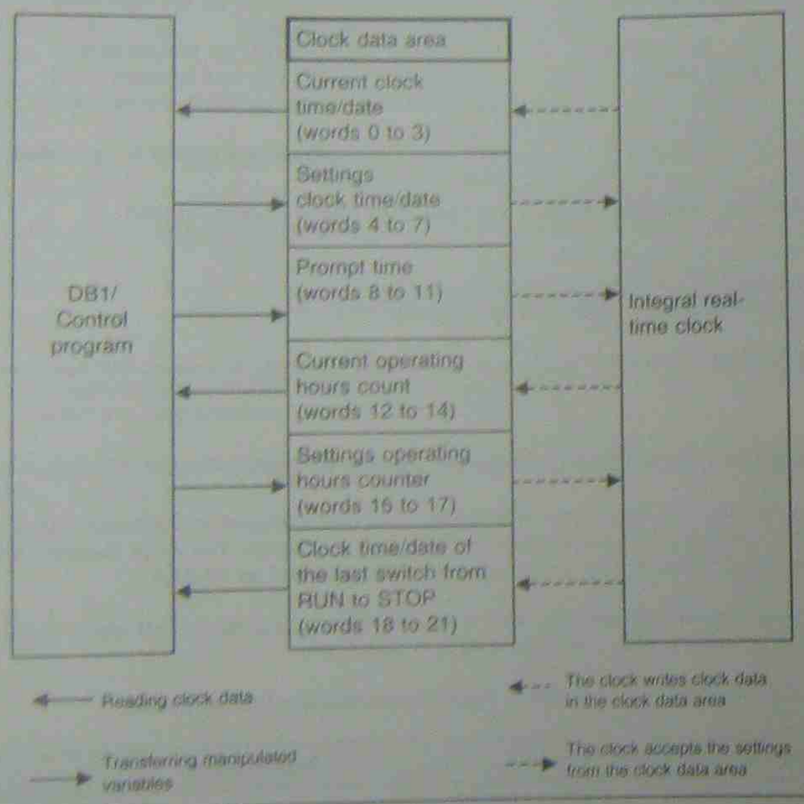


Figure 13-6. How DB1 or the Control Program and the Clock Access the Clock Data Area

When you set the clock, you have to transfer only the data needed to implement a particular function. For example, if you want to change only the clock function data, you do not have to enter data for the time prompt function or for the operating hours counter.

The clock data area has the same structure wherever it is located: in the data block area, the flag area, the input area, or the output area. Table 13-3 provides you with information about where specific clock data is located within the clock data area. The explanations for Table 13.3 follow the table.

Table 13-3. Clock Data in the Clock Data Area

Clock Data Area	Meaning	Left Word	Right Word
0	Current clock time/date	---	Weekday
1		Day	Month
2		Year	AM/PM (Bit 7)*, Hour
3		Minute	Second
4	Settings for clock time/date	---	Weekday
5		Day	Month
6		Year	AM/PM (Bit 7)*, Hour
7		Minute	Second
8	Time prompt	---	Weekday
9		Day	Month
10		---	AM/PM (Bit 7)*, Hour
11		Minute	Second
12	Current operating hours	---	Second
13		Minutes	Hour
14		Hours · 100	Hours · 10,000
15	Settings for operating hours counter	---	Second
16		Minutes	Hours
17		Hours · 100	Hours · 10,000
18	Clock time/date after a switch from RUN to STOP or Power OFF (only if bit 5 in the status word = 1)	---	Weekday
19		Day	Month
20		Year	AM/PM (Bit 7), Hour
21		Minute	Second

* Significant only in the 12-hour mode Bit 7 = 1 means PM, bit 7 = 0 means AM

13

Make certain you are aware of the following information when you make inputs into the clock data area.

- Entries into the clock data area must be in BCD code.
- The clock runs either in the 12-hour mode or the 24-hour mode depending on how you set bit 1 in the status word. See section 13.5 for additional information.
- The AM/PM flag (0 = AM, 1 = PM) is significant only for the 12-hour mode of the hardware clock. The AM/PM flag corresponds to bit 7 in words 2, 6, 10, and 20.

  In the 12-hour mode, you have to set the hours and the AM/PM flag the same way for the clock and prompt functions.

  In the 24-hour mode, if you set an AM/PM flag when you enter the values for the clock and prompting time, then the program sets the relevant error bit.
- The clock settings you enter must be within the range defined in Table 13.4.

Table 13-4. Range Definitions for Clock Data

Variable	Permissible Parameters	Variable	Permissible Parameters
Seconds	0 to 59	Day	1 to 31
Minutes	0 to 59	Month	1 to 12
Hours	In the 24-hour mode: 0 to 23 In the 12-hour mode: for AM: 1 to12 (12 = 12 o'clock noon) for PM: 81 to 92 (81 = 1 o'clock PM) 0 to 999999 when entering the operating hours	Year	0 to 99
Weekday	1 to 7 1 = Sunday 2 = Monday 3 = Tuesday 4 = Wednesday 5 = Thursday 6 = Friday 7 = Saturday		

If your inputs differ from the ones described, the operating system outputs error messages that are displayed in the status word. The operating system resets error messages displayed in the status word the next time you set the clock, prompt time, or the operating hours counter, if the new settings are within the definition range. See section 13.5.

If you do not wish to modify one of the setting values, you can enter at its place XX (ASCII code) in DB1, or FF (hexadecimal code) when you are programming in the system data.

If the clock data area is located at the end of other areas (flags, data blocks, inputs, and outputs) and there is insufficient memory space available for the clock data area, the amount of clock data transferred is only as much as will fit in the area available. Settings are not accepted if they lie outside of the available range.

- If clock data is located in the non-retentive flag area, then the following two events occur.
  - All the settings are lost after Power OFF and cold restart.
  - The time the last switch from RUN to STOP occurred is lost.
- Remember that you can determine where the clock data area is located. The word numbers listed in Table 13.3 are relative.
  - If your clock data area is located in a data block and does not begin with DW0 but DWX, then you must add the value X to the word number shown in Table 13-3.

    Example: Your clock data area begins with DW124. The data for the time and date is then stored in DW124 to DW127.
  - If you locate your clock data area in the flag area beginning with flag word 0, then you must multiply the appropriate word number listed in Table 13-3 by a factor of 2 to obtain the appropriate word address.

    Example: You locate your clock data area in the flag operand area beginning with flag word 0. Data for the operating hours counter is then stored beginning with the FW24 address.

    If, in addition, your clock data area does not begin at flag word 0, you also have to add the value to the word number shown in Table 13-3.

13

13.5 Structure of the Status Word and How to Scan it

You can scan the status word to identify errors in the entered settings. You can deliberately change certain bits in the status word to enable or disable transfer or read operations. You can use designated flag bits to govern the clock's behavior when the programmable controller is switched from the RUN to the STOP mode or during Power OFF.

- The status word can be located in the flag area or in a data block. You must define the location of the status word in DB1 or directly in system data 9 and 10. See section 13.6.
- The integral real-time clock runs independently of the set operating mode. Access to the clock data area depends on the set operating mode and the signal states of bits 4 and 5 in the status word. You can set or reset these bits using the "S" or "R" operations in the control program.

  If you use an operator panel, such as the OP 396, to monitor the program, it is an advantage to have the programmable controller update the clock time (the current date) even in the STOP mode.
- The operating system resets the "transfer settings" bits (bits 2, 10, and 14 in the status word) under the following conditions.
  - The settings have been transferred.
  - The settings have not been transferred because they were outside of the permissible range. The corresponding error bit (bits 0, 8, and 12 in the status word) is set.
- The operating system does not reset the "transfer settings" bits (bits 2, 10, and 14 in the status word) under the following conditions.
  - The system data for the clock is either incorrect or not available.
  - The clock data area is too small.
  - The clock is defective (hardware error).
- There are four types of bits in the status word.
  - Clock flags
  - Operating system flags
  - Operating hours counter flags
  - Prompt time flags

Tables 13.5 through 13.8 provide you with information about the significance of the signal states of the respective flags.

Clock Flags

Table 13-5. Significance of Bits 0, 1, and 2 of the Status Word

Bit Number	Signal State	Meaning
0	1	Error in setting entry
	0	No error in setting entry
1	1	12-hour clock mode
	0	24-hour clock mode
2	1	Transfer settings
	0	Do not transfer settings

Operating System Flags

Table 13-6. Significance of Bits 4 and 5 of the Status Word

Operating Mode	Bit Number Status Word	Signal State	Meaning
STOP	4	1	The clock updates only words 0 to 3 (current time/date) in the clock data area. You can set the clock by using the FORCE VAR programmer function.
		0	The clock does not update the clock data area. Words 0 to 3 contain the time at which the last switch from RUN to STOP occurred.
	5	1	Words 18 to 21 contain the time at which the last RUN to STOP switch occurred or the time at which the last Power OFF occurred if bit 4 is also set.
		0	Words 18 to 21 are not used.
RUN	4	1/0	The clock continually updates the clock data area (Words 0 to 17).
	5	1	Words 18 to 21 contain the time at which the last switch from RUN to STOP occurred or the time at which the last Power OFF occurred.
		0	Words 18 to 21 are not used.

Operating Hours Counter Flags

Table 13-7. Significance of the Operating Hours Counter Flags Bits 8, 9, and 10 of the Status Word

Bit Number	Signal State	Meaning
8	1	Error in setting entry
	0	No error in setting entry
9	1	Enable the operating hours counter
	0	Disable the operating hours counter
10	1	Transfer the settings
	0	Do not transfer the settings

Prompting Time Flags

Table 13-8. Significance of the Prompting Time Flags Bits 12, 13, and 14 of the Status Word

Bit Number	Signal State	Meaning
12	1	Error in setting entry
	0	No error in setting entry
13	1	The set prompting time is reached
	0	The set prompting time is not reached
14	1	Transfer the settings
	0	Do not transfer settings

The operating system requires bit numbers 6, 7, 11, and 15. You can not use these bits.

Scanning the Status Word

In a data block, you can use the "P <data word number> <bit number>" operation to scan the individual bits of a date word. In the flag area, you can scan the individual bits if you enter the <byte address> and the <bit number>.

Example: The status word is stored in DW13. You are checking to see if the set prompting time has been reached. The "P D 13.13" instruction triggers a scan. If the status word is stored in PW12, then the same scan would be "A F 12.5".

Backup of the Hardware Clock

If there is a backup battery, the clock continues to run even after Power OFF. If the programmable controller does not have a backup battery, the clock values will be set at "October 1, 1991, 12:00:00 o'clock, weekday: 3" when the clock is initialized after a Power ON. The default is the 24-hour time mode. You should install a battery only during Power ON; otherwise, you would lose the clock data.

13.6 Setting Parameters for the Clock Data Area and the Status Word in the System Data Area

Table 13-9. The System Data Area for the Integral Real-Time Clock

Absolute Address RAM	System Data Word	Meaning	Permissible Parameters
5D10	8	Operand area for the clock data	ASCII characters I, Q, F, D
5D11		Start address for the clock data Operand area D Operand Areas I, Q, F	DB number DB2 to DBFF$_H$ Byte address
5D12	9	Start address for the clock data Relevant only for operand area D	DB word number DW0 to DWFF$_H$
5D13		Operand area for the status word	ASCII characters I, Q, F, D
5D14	10	Start address for the status word Operand area D Operand areas I, Q, F	DB number DB2 to DBFF$_H$ Byte address
5D15		Start address for the status word Relevant only for operand area D	DB word number DW0 to DWFF$_H$
5D16	11	Status for hardware[1] (only bits 0 and 1 are relevant) • If either bit 0 or bit 1 is set, the clock chip is defective • If no bit is set, the clock chip is running	"0", "1"
5D17	11	Incorrect correction value? (only bit 15 is relevant) • If bit 15 is set, the correction value is incorrect (> +400 or < -400) • If bit 15 is not set, the correction value is correct	"0", "1"
5D18	12	Correction value[2]	-400 to 400

1 You can scan SD11 during start-up. You must call up an FB in OB21 or OB22 by using "L RS 11" to read out and then continue processing SD11.
2 Always use the "L KF" instruction to load the correction value in ACCU 1 since negative values can also be specified.

The following section is intended to help you to start running the integral real-time clock as quickly as possible by setting parameters in the system data. You need to be familiar with the clock data area described in sections 13.4 and 13.5 in order to understand this section.

> **Note**
>
> The clock time is updated one second after the start of the next cycle.

Assignment:

You want to set the clock to the following values.

Monday, 12.02.91; 10:30:00

The status word is assigned to flag word FW12. Clock data is stored in DB75 beginning with DW0. The two ways to transfer clock settings are as follows.

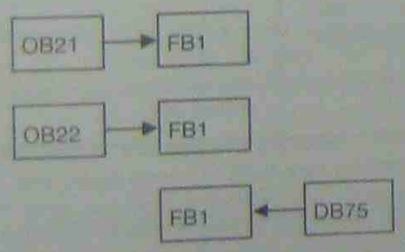
1. Use the FORCE VAR programmer function when the programmable controller is in the RUN mode.
2. Use the FORCE VAR programmer function when the programmable controller is in the STOP mode and status word bit 4 = 1.

The following example uses the first method. Proceed as follows.

- Set the programmable controller to Power OFF.
- Set the operating mode switch to STOP.
- Set the programmable controller to Power ON.
- Perform an overall reset on the programmable controller. See section 4.1.3.
- Use the following program to program the programmable controller.
- Set the operating mode switch to RUN.

The integral real-time clock is running.

Program Structure:



The system data is defined during the change from STOP to RUN.

The system data is defined when the programmable controller is switched on.

Set the new date and time.

The Block Entry Sequence and a Programming Example:

The following procedure is suggested:

1. Program FB1 - Defining system data for the integral real-time clock
2. Program OB21 - Calling up FB1 during a change from STOP to RUN
3. Program OB22 - Calling up FB1 when the programmable controller is switched on
4. Generate DB75 - Storing clock data
5. Transfer new data to the clock using the FORCE VAR programmer function (programmable controller in the RUN mode) .

The respective programming examples are shown in Tables 13.10 to 13.13.

Table 13-10. FB1 Program

STL	Meaning	Explanation
FB 1 NAME: CLOCK		
L KH 4 4 4 B	ASCII Code for the "D" character	Clock data is located in DB75 beginning with DW0
T RS 8	Block number "75_D" Storing in system data word 8 DW0 is the start address for clock data	
L KH 0 0 4 D	ASCII Code for the "M" Character	The status word is located in FW12
T RS 9	Storing in system data word 9 Flag word Number "12_D"	
L KH 0 C 0 0	ASCII Code is irrelevant	
T RS 10	Storing in system data word 10	
BE		

13

Table 13-11. OB21 Program

STL	Explanation
OB 21 JU FB 1 NAME: CLOCK BE	The function block is called up once during a switch from STOP to RUN.

Table 13-12. OB22 Program

STL	Explanation
OB 22 JU FB 1 NAME: CLOCK BE	The function block is called up once when the programmable controller is switched on.

Table 13-13. DB75 Program

STL	Explanation
DB 75 0: KH = 0000; 1: KH = 0000; 2: KH = 0000; 3: KH = 0000; 4: KH = 0000; 5: KH = 0000; 6: KH = 0000; 7: KH = 0000	Define the number of data words. (Data words 0 to 7 are used in the example.) Define the numerical representation. Hex is used in the examples.

Reading and Setting the Time and Date

After you enter the program, you can test it as follows.

- Switch the programmable controller to the RUN mode.
- Use the FORCE VAR programmer function to enter the following.
  1. Data block number
  2. Data words DW0 to DW7
  3. Clock data
  4. Status word

Table 13-14. FORCE VAR Function

Operand	Signal States	Explanation
DB 75		
DW 0	KH = 0003	Tuesday
DW 1	KH = 0110	October 1
DW 2	KH = 9112	1991, 12 o'clock
DW 3	KH = 0000	(reading current clock data)
DW 4	KH = 0002	Monday
DW 5	KH = 0212	December 2
DW 6	KH = 9110	1991, 10.30 o'clock
DW 7	KH = 3000	(writing new settings)
FW 12	KM = 00000000 00000100	If you set bit 2 in the status word to "1", the new settings are transferred to the clock.

- Begin status processing by pressing the ENTER key twice. Bit 2 in the status word is reset. The clock runs with the new settings.

Note

Besides using the method described on Table 13-14 (with the FORCE VAR function), you can also enter the new settings directly into the data block. In that case, store the new settings in data words DW4 to DW7 of data block DB75.
See Table 13.13.