Training document for the company-wide automation solution Totally Integrated Automation (T I A)

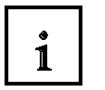
MODULE B2

Analog value processing

We thank the company Michael Dziallas Engineering and the instructors of vocational schools as well as further persons for the support with the production of the document.

PAGE:

The following symbols stand for the specified modules:

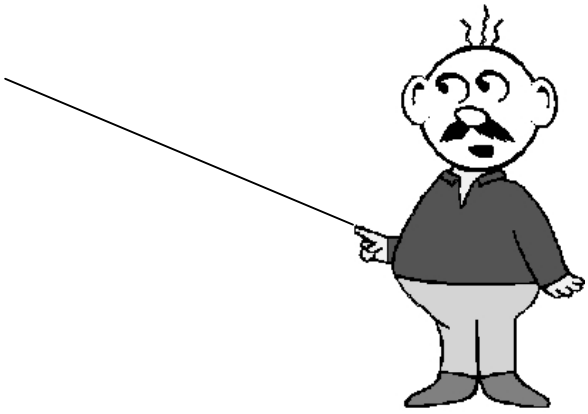


Information

Programming

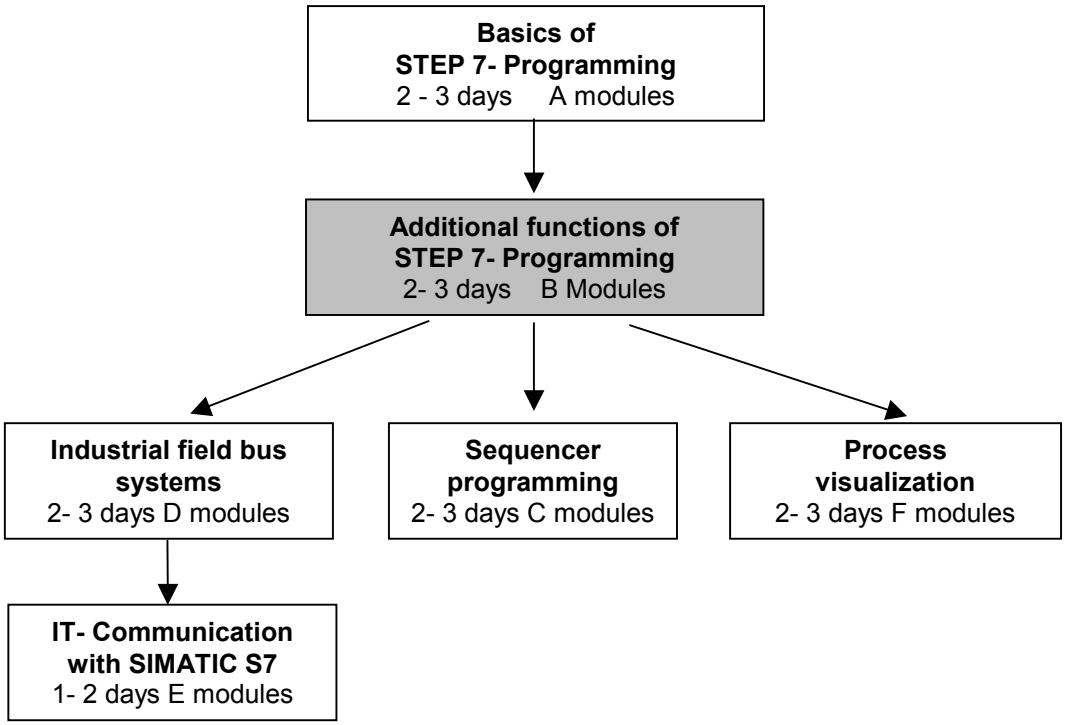
Example exercise

Notes



1. FORWARD

This module B2 is assigned content wise to **Additional functions of STEP 7- Programming**.



Learning goal:

In this module, the reader will learn about how analog values are inputted, processed, and outputted into a SIMATIC S7.

- Analog signals
- Data types in STEP 7
- Mathematical operations
- Conversion of data types in STEP 7
- Input and normalize an analog value
- Normalize and output an analog value

Requirements:

For the successful use of this module, the following knowledge is assumed:

- Knowledge in the use of Windows 95/98/2000/ME/NT4.0
- Basics of PLC- Programming with STEP 7 (e.g. Module A3 - 'Startup' PLC programming with STEP 7)

Required hardware and software

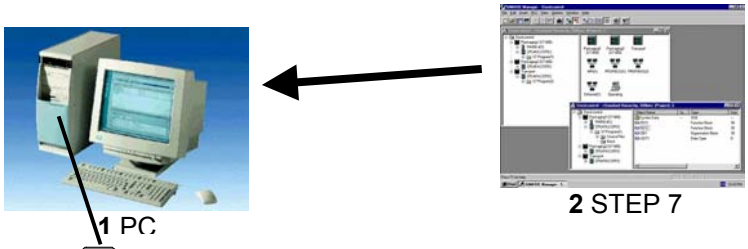
1 PC, Operating system Windows 95/98/2000/ME/NT4.0 with
- Minimal: 133MHz and 64MB RAM, approx. 65 MB free hard disk space
- Optimal: 500MHz and 128MB RAM, approx. 65 MB free hard disk space

2 Software STEP 7 V 5.x

3 MPI- Interface for the PC (e.g. PC- Adapter)

4 PLC SIMATIC S7-300 with at least one analog input/output module, by which a potentiometer or another analog signal transmitter is connected. Also, an analog value display must be connected to an analog output.

Example configuration:
- Power supply: PS 307 2A
- CPU: CPU 314
- Digital inputs: DI 16x DC24V
- Digital outputs: DO 16x DC24V / 0.5 A
- Analog In-/ Outputs: AI 4/ AO 2 x 8Bit



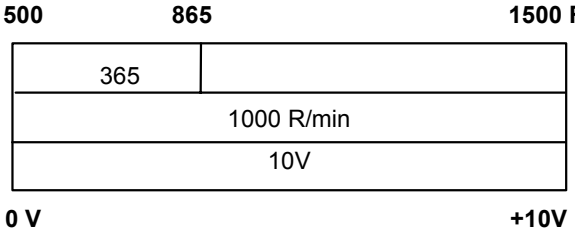
2. ANALOG SIGNALS

Contrary to a binary signal, which can only accept two signal statuses ‘Voltage available +24V‘ and ‘Voltage not available 0V‘, analog signals can also take as many values as desired within a certain range. A typical example of an analog transmitter is a potentiometer. Depending upon the position of the rotary button any resistance can be stopped here up to the maximum value.

Examples for analog measurements in control systems technology:

- Temperature -50 ... +150°C
- Flow 0 ... 200l/min
- Revolutions 500 ... 1500 R/min
- Etc...

These sizes are converted into electrical voltages, currents, or reisistances with the help of a measuring transducer. E.g. if a number of revolutions is to be collected, the speed range can be converted from 500... 1500 R/min over a measuring transducer into a voltage range from 0... +10V. At a measured number of revolutions of 865 R/min, the measuring transducer would emit a voltage level of + 3.65 V.



10V: 1000 R/min = 0,01 V/R/min

365 R/min x 0,01 V/R/min = 3,65

If similar measurements with a PLC are processed, then the inputted voltage, current or resistance value must be converted into digital information. One indicates this conversion as analog--digital conversion (A/D conversion). This means, that e.g. the voltage value of 3.65V is deposited as information into a row of equivalent binary digits. The more equivalent binary digits are used for the digital representation, thus the resolution becomes finer. If one would have used 1 bit e.g. for the voltage range 0... +10V, only one statement could be met, if the measured voltage in the range 0...+5V or in the range +5V...+10V. With 2 bits, the range can be partitioned into 4 single areas, 0... 2.5/2.5... 5/5... 7.5/7.5... 10V. Usually A/D conversion in control systems engineering changes with 8 or 11 bit. You have 256 single areas with 8 bits and with 11 bit a resolution of 2048 single areas.

3. DATA TYPES IN STEP 7

In the SIMATIC S7 there is a multiple of different data types, with which different formats of numbers are represented. In the following a complete listing of the elementary data types are given

Type and description	Size In Bits	Format-Options	Range and number notation (lowest to highest values)	Example
BOOL (Bit)	1	Boolean text	TRUE/FALSE	TRUE
BYTE (Byte)	8	Hexadecimal number	B#16#0 to B#16#FF	B#16#10
WORD (Word)	16	Binary number	2#0 to 2#1111_1111_1111_1111	2#0001_0000_0000_0000
		Hexadecimal number	W#16#0 to W#16#FFFF	W#16#1000
		BCD	C#0 to C#999	C#998
		Decimal number unsigned	B#(0,0) to B#(255,255)	B#(10,20)
DWORD (Double word)	32	Binary number	2#0 to 2#1111_1111_1111_1111_1111_1111_1111_1111	2#1000_0001_0001_1000_1011_1011_0111_1111
		Hexadecimal number	DW#16#0000_0000 to DW#16#FFFF_FFFF	DW#16#00A2_1234
		Decimal number unsigned	B#(0,0,0,0) to B#(255,255,255,255)	B#(1,14,100,120)
INT (Integer)	16	Decimal number signed	-32768 to 32767	1
DINT (Int,32 bit)	32	Decimal number signed	L#-2147483648 to L#2147483647	L#1
REAL (Floating-point number)	32	IEEE floating-point number	Upper limit: +/-3.402823e+38 Lower limit: +/-1.175495e-38	1.234567e+13
S5TIME (Simatic-Time)	16	S7-Time in steps of 10 ms	S5T#0H_0M_0S_10MS to S5T#2H_46M_30S_0MS and S5T#0H_0M_0S_0MS	S5T#0H_1M_0S_0MS S5TIME#1H_1M_0S_0MS
TIME (IEC-Time)	32	IEC-Time in steps from 1ms, integer signed	-T#24D_20H_31M_23S_648MS to T#24D_20H_31M_23S_647MS	T#0D_1H_1M_0S_0MS TIME#0D_1H_1M_0S_0MS
DATE (IEC-Date)	16	IEC-Date in step of 1 day	D#1990-1-1 to D#2168-12-31	DATE#1994-3-15
TIME_OF_DAY (Time)	32	Time in steps of 1ms	TOD#0:0:0.0 to TOD#23:59:59.999	TIME_OF_DAY#1:10:3.3
CHAR (Character)	8	ASCII-Characters	´A´, ´B´ etc.	´B´

Note : For analog value processing, Data types **INT** and **REAL** play a big role, because inputted analog values exist as real numbers in the format **INT**. Due rounding errors by **INT**, only real numbers **REAL** come into question for an accurate further processing.

4. MATH OPERATIONS

4.1 CALCULATION WITH FIXED-POINT NUMBERS (INT AND DINT)

With fixed-point numbers, the mathematical unit operations addition, subtraction, multiplication and division are possible. However the places after the decimal point remain unconsidered, which lead to rounding errors with the division.

Operation	Size in Bit	Function
+I	16	Add the contents of the low-order word of ACCUs 1 and 2 and save the result in the lower-order word of the ACCU 1.
-I	16	Subtract the contents of the low-order word of ACCU 1 from the contents of the low-order word of ACCU 2 and saves the result in the low-order word of ACCU 1.
*I	16	Multiply the contents of the low-order word of ACCUs 1 and 2 and save the result (32 Bit) in ACCU 1.
/I	16	Divide the contents of the low-order word from ACCU 2 by the contents of low-order word of ACCU 1. The result is saved in the low-order word of ACCU 1. The remainder is saved in the high-order word of the ACCU 1.
+D	32	Add the contents of the ACCUs 1 and 2 and saves the result in ACCU 1.
-D	32	Subtract the contents of ACCU 1 from the contents of ACCU 2 and save the result in ACCU 1.
*D	32	Multiply ACCU 1 with the contents of ACCU 2 and save the result in ACCU 1.
/D	32	Divide the contents of ACCU 2 by the contents of ACCU 1 and save the Quotient in ACCU 1.
MOD	32	Divide the contents of ACCU 2 by the contents of ACCU 1 and saves the remainder as a result in ACCU 1.

4.2 CALCULATION WITH FLOATING-POINT NUMBERS (REAL)

With floating-point numbers, a multiple of mathematical operations can be accomplished. The positions to the right of the decimal point are always considered here.

Operation	Function
+R	Add the floating-pt. numbers (32 Bit, IEEE-FP) in the ACCUs 1 and 2 and save the 32-Bit-Result in ACCU 1.
-R	Subtract the floating-pt. number (32 Bit, IEEE-FP) in ACCU 1 from the floating pt. number (32 Bit, IEEE-FP) in ACCU 2 and save the 32-Bit-result in ACCU 1.
*R	Multiply the floating-pt. number (32 Bit, IEEE-FP) in ACCU 1 with the floating-pt. number (32 Bit, IEEE-FP) in ACCU 2 and save the 32-Bit-result in ACCU 1.
/R	Divide the floating-pt. number (32 Bit, IEEE-FP) in ACCU 2 by the floating-pt. number (32 Bit, IEEE-FP) in ACCU 1. The 32-Bit-result is saved in ACCU 1.
SQRT	Calculate the square root of the floating-pt. number (32 Bit, IEEE-FP) in ACCU 1 and save the 32 Bit-result in ACCU 1.
SQR	Calculate the square of the floating-pt. number (32 Bit, IEEE-FP) in ACCU 1 and save the 32 Bit-result in ACCU 1.
LN	Calculate the natural log of the floating-pt. number (32 Bit, IEEE-FP) in ACCU 1 and Save the 32 Bit-result in ACCU 1.
EXP	Calculate the exponential value of the floating-pt. number (32 Bit, IEEE-FP) to base e and save 32 Bit-result in ACCU 1.
SIN	Calculate the sine the floating-pt. number (32 Bit, IEEE-FP) in ACCU 1 and save the 32 Bit-result in ACCU 1.
COS	Calculate the Cosine of the floating-pt. number (32 Bit, IEEE-FP) in ACCU 1 and save the 32 Bit-result in ACCU 1.
TAN	Calculate the tangent of the floating-pt. number (32 Bit, IEEE-FP) in ACCU 1 and save the 32 Bit-result in ACCU 1
ASIN	Calculate the arcsine of the floating-pt. number (32 Bit, IEEE-FP) in ACCU 1 and save the 32 Bit-result in ACCU 1.
ACOS	Calculate the arccosine of the floating-pt. number (32 Bit, IEEE-FP) in ACCU 1 and save the 32 Bit-result in ACCU 1.
ATAN	Calculate the arctangent of the floating-pt. number (32 Bit, IEEE-FP) in ACCU 1 and save the 32 Bit-result in ACCU 1.

4.3 DATA TYPE- CONVERSION OPERATIONS

Since the numbers do not often exist for the further processing of important number formats, these numbers must be adjusted with help of the conversion operations.

Operation	Function
BTI	BCD converted into an integer (16 Bit). This operation converts a binary-coded decimal in ACCU 1 into an integer (16 Bit).
BTD	BCD converted into an integer (32 Bit). This operation converts a binary-coded decimal in ACCU 1 into an integer (32 Bit).
ITB	Integer (16 Bit) converted into BCD. This operation converts an integer (16 Bit) in the low-order word of the ACCU 1 into a binary-coded decimal value.
ITD	Integer (16 Bit) converted into an integer (32 Bit). This operation converts an integer (16 Bit) in the low-order word of the ACCU 1 into an integer (32 Bit).
DTB	Integer (32 Bit) converted into BCD. This operation converts an integer (32 Bit) in ACCU 1 into a binary-coded decimal value.
DTR	Integer (32 Bit) converted into integer (32 Bit, IEEE-FP). This operation converts an integer (32 Bit) in ACCU 1 into an integer (32 Bit, IEEE-FP).
RND	Round for integer. This operation rounds the converted number to the next integer. When the fraction of the converted number is exactly between an even and uneven result, the operation choose the even result.
RND+	Round to the next highest integer. This operation rounds the converted number to the smallest integer, that is greater than or equal to the converted integer.
RND-	Round to the next lowest integer. This operation rounds the converted number to the largest integer, that is smaller or equal to the converted integer.
TRUNC	Round with truncation. This operation converts the whole part of the integer.

Note: In the case of analog value processing, the analog value lies in **INT** and should be converted for accurate further processing because of the round error by **INT** in integer **REAL**. Since the conversion is not direct, it must first convert **ITD** into **DINT** and then into **REAL** with **DTD**.

.

5. INPUTTING/OUTPUTTING ANALOG VALUES

Analog values are inputted as word information in the PLC. Respectively, the access of this word is performed with the instructions:

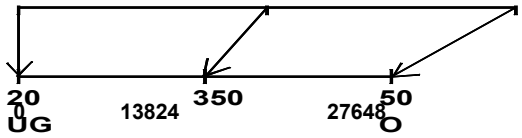
L	PIW x	for 'Load analog input word'
T	PQW x	for 'Transfer analog output word'

Each analog value ("Channel") allocates a peripheral input-output word. The format is an integer **INT**.

The addressing of the in and/or output words depends on the module start address. If the analog module is placed in slot 4, then it has the default start address 256. The start address of each further analog module increases by 16 for each slot. This default start address can be checked in the hardware configuration table under a detailed view.

The address of the first analog input would be on slot 6 PIW 288 with an analog input. Those of the second analog input on PIW 290, those of the first analog output on PQW 288 etc. .

The analog value transformation to the further processing in the PLC (digitized) is equal to analog input and output.
For the module SM334, with 4 analog inputs and 2 analog outputs, with the analog image processing from 0 to 10V as well as 0 to 20mA, the digitized value range appears as follows:



These digitized values must often be normalized through corresponding further processing in the PLC.

5.1. INPUT AND NORMALIZE ANALOG VALUE

If an analog value is presented as a digitized value, it must still be normalized before it can be processed in the PLC.
Likewise, the analog output from the peripheral output value follows after the normalization of the output value.

In a STEP 7- Program, normalizing is caught in the math operation.
For this reason, the math operation can take place as accurately as possible. The values to be normalized must be converted into the data type REAL so that the rounding errors are at a minimum.

Exercise:

In the following example, a value from 0 to 10V with an analog module SM334 in slot 6 is inputted (PIW288). It is first presented as an INTEGER (16 Bit) and should be normalized from 100 to 1000 in REAL format and saved in the memory bit double word MD10.

Solution in STL:

```
L     PIW 288      //Analog value input 0 to 10 V contains 0 to 27648 integers (16 Bit)
ITD                //Value of integer (16 Bit) converted into integer (32 Bit)
DTR                //Value of integer (32 Bit) converted into a real number
L     2.7648e+4    //
/R                 //Division with real number 27648
L     9.000e+2     //
*R                 // Multiplication with real number 900 (1000-100)
L     1.000e+2     //
+R                 // Addition with real number 100 (Offset)
T     MD10         //Normalized value 100 to 1000 in real format
```

5.2. NORMALIZE AND OUTPUT ANALOG VALUE

If a standardized value is present and is to be used on an analog output module, then it must be normalized.
In a STEP 7- Program, normalizing is caught in the math operation. This occurs in the data type REAL so that the rounding errors are at a minimum. This value is only then rounded to an integer value. The places behind the decimal point are however lost.

Example:

In the following example, a value from 100 to 1000 is saved in real format in memory bit double word MD20 and should be outputted normalized from 0 to 10V with an analog module SM334 (PQW288) .

Solution in STL:

```
L     MD20        // Value 100 to 1000 in real format
L     1.000e+2    //
-R                // Subtraction with real number 100 (Offset)
L     9.000e+2    //
/R                // Division with real number 900
L     2.7648e+4   //
*R                // Multiplication with real number 27648
RND               // Round to an integer
T     PQW 288     // 0 to 27648 real number (16 Bit) corresponds to analog value output 0 to 10 V
```