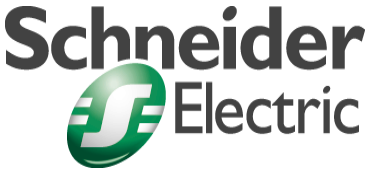
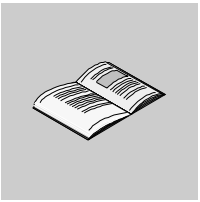
Concept
IEC block library
Part: EXPERTS

840 USE 504 00 eng Version 2.6

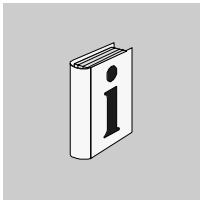


33002213.00

Table of Contents



About the book



At a Glance

Document Scope This documentation will help you in configuring the functions and the function blocks.

Validity Note This documentation applies to Concept 2.6 in Microsoft Windows 98, Microsoft Windows 2000 and Microsoft Windows NT 4.x.

Note: Additional up-to-date tips can be found in the Concept README data.

Related Documents

Title of Documentation	Reference Number
Concept Installation instructions	840 USE 502 00
Concept Installation Instructions	840 USE 503 00
Concept-EFB User Manual	840 USE 505 00
Concept LL984 Block Library	840 USE 506 00

User Comments We welcome your comments about this document. You can reach us by e-mail at TECHCOMM@modicon.com

General information on the EXPERTS block library

I

Overview

Introduction

This section contains general information about the EXPERTS block library.

What's in this part?

This part contains the following chapters:

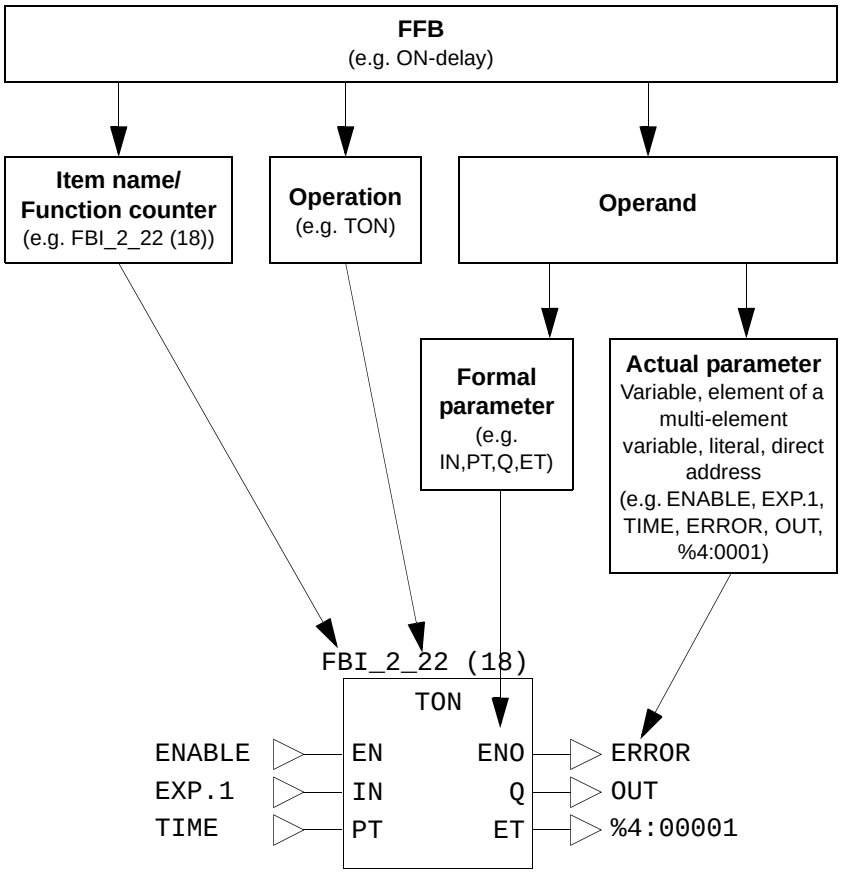
Parameterizing functions and function blocks

1

Parameterizing functions and function blocks

General

Each FFB consists of an operation, the operands needed for the operation and an instance name or function counter.



Operation

The operation determines which function is to be executed with the FFB, e.g. shift register, conversion operations.

Operand

The operand specifies what the operation is to be executed with. With FFBs, this consists of formal and actual parameters.

Formal/actual parameters

The formal parameter holds the place for an operand. During parameterization, an actual parameter is assigned to the formal parameter.

The actual parameter can be a variable, a multi-element variable, an element of a multi-element variable, a literal or a direct address.

Conditional/ unconditional calls

"Unconditional" or "conditional" calls are possible with each FFB. The condition is realized by pre-linking the input EN.

- Displayed EN
  conditional calls (the FFB is only processed if EN = 1)
- EN not displayed
  unconditional calls (FFB is always processed)

Note: If the EN input is not parameterized, it must be disabled. Any input pin that is not parameterized is automatically assigned a "0" value. Therefore, the FFB should never be processed.

Calling functions and function blocks in IL and ST

Information on calling functions and function blocks in IL (Instruction List) and ST (Structured Text) can be found in the relevant chapters of the user manual.

EFB descriptions

II

Overview

Introduction

These EFB descriptions are listed in alphabetical order.

Note: The number of inputs of some EFBs can be increased to a maximum of 32 by changing the size of the FFB symbol vertically. Please refer to the description of the individual EFBs to see which EFBs are involved.

What's in this part?

This part contains the following chapters:

ERT_854_10: Data transfer EFB

2

Overview

Introduction

This chapter describes the ERT_854_10 block.

What's in this chapter?

This chapter contains the following topics:

Brief description

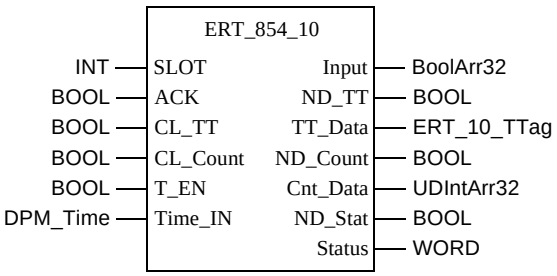
Function description

The ERT_854_10 EFB provides the programmer with a software interface to the ERT 854 10 module. It allows easy access to functions like counters, time stamp, status or time synchronization. Using the input and output registers, the ERT_854_10 EFB can coordinate the flow of Multiplex data from the ERT to the PLC. It also ensures that the intermediate counter values are stored in an internal memory area until the data is complete, so a consistent set of all counter values is made available to the statement list. A flag "New data" is always set for every data type if the input data type was copied into the corresponding EFB output structure. The parameters EN and ENO can also be configured.

Representation

Symbol

Function Block representation:



Parameter description

Description of the function block parameters:

Parameter	Data type	Meaning
SLOT	INT	The Slot index is assigned to the ERT-EFB from either the QUANTUM EFB or DROP EFB and contains the configured input and output references (3x and 4x registers)
ACK	BOOL	Event confirmation: Setting ACK signals that the user is ready to receive the next result and deletes the TT_Data register. If ACK remains set, "continuous operation" is done.
CL_TT	BOOL	Delete the ERT event FIFO buffer by setting CL_TT. Storage of events is blocked until the CL_TT is reset to 0.
CL_Count	BOOL	Delete all ERT counters by setting CL_Count. Counting is interrupted until CL_Count is reset to 0.
T_EN	BOOL	Enables a time transfer, e.g. from the ESI via Time_IN, if set
Time_IN	DPM_Time	Structure of the input time, e.g. from the ESI, for time synchronization of the ERT (contains the edge controlled time synchronization in the "Sync" element)
Input	BOOLArr32	Output array for all 32 digital inputs in BOOL format (also provided in the form of word references as 3x registers 1+2)
ND_TT	BOOL	Flag, new data in TT_Data structure: remains set until user confirmation with ACK
TT_Data	ERT_10_TTag	Event message output structure with time stamp. An event is held and NDTT is set to 1 until there is a user enable with ACK = 1.
ND_Count	BOOL	Flag, new counter data in Cnt_Data structure: The value 1 is set for only one cycle and is not acknowledged.
Cnt_Data	UDIntArr32	Output array for 32 counter values (is overwritten after the EFB has received a complete set of consistent counter values (configured as:8, 16, 24, or 32).
ND_Stat	BOOL	Flag; new status data in status word: The value 1 is set for only one cycle and is not acknowledged.
Status	WORD	Output word for EFB/ERT status (for internal details see *Data Flow, p. 21*)

Internal time synchronization

Structure of DPM_Time for ERT internal time synchronization e.g. via the ESI:

Element	Element type	Meaning
Sync	BOOL	Clock synchronization with positive edge (hourly or on command)
Ms	WORD	Time in milliseconds
Min	BYTE	Time invalid / minutes
Hour	BYTE	Summer time / hours
Day	BYTE	Day of the week / Day in the month
Mon	BYTE	Month
Year	BYTE	Year

Event structure

Event structure of the ERT_10_TTag with 5 Byte time stamp (further information can be found in *Data Flow, p. 21*):

Element	Element type	Meaning
User	BYTE	Complete time / user number [module number]
Input	BYTE	Event set type / No. of the first input
In	BYTE	Event data: 1, 2 or 8 managed positions
Ms	WORD	Time in milliseconds
Min	BYTE	Time invalid / minutes
Hour	BYTE	Summer time / hours
Day	BYTE	Day of the week / Day of the month

Mode of Functioning

ERT data transfer

The number of I/O words available on the local and remote subracks is limited to 64 inputs and 64 outputs. For this reason, the number of ERT modules which can be used per local/remote backplane is limited to 9, with the currently selected minimum requirements of 7 input words and 5 output words per module.

The size of the required ERT data transfer is considerably larger:

- 32 counters = 64 words,
- an event with a 5 byte time stamp = 4 words,
- 32 digital values and the ERT status = 3 words.

These inconsistent size requirements necessitate the use of a special transfer EFB called ERT_854_10 to execute the required operations on the PLC and to adjust the ERT representation of the data in Multiplex form. An EFB is required for every ERT module.

To simplify matters, only the EFB parameters which will actually be used need to be configured. This saves on the amount of configuration effort, particularly when the counter inputs and event inputs are not mixed together. Unfortunately memory cannot be reserved for this because Concept has occupied the outputs with invisible dummy variables.

Basic structure of the ERT_854_10 input register block with seven 3x registers for transfer from the ERT to the PLC

Basic structure of the register block

ERT_854_10 input register block:

Contents	Function
Digital inputs 1 …. 16	Digitally processed input data which is cyclically updated (the module's input address corresponds to that of the digital standard input modules, i.e. inputs 1 … 16 correspond to bits 15 … 0)
Digital inputs 17 …. 32	
Transfer status	IN transfer status (TS_IN)
MUX 1	Multiplex data block for block transfer
MUX 2	1 event with 5 byte time stamp or
MUX 3	2 counter values of possible configured maximum 32 or
MUX 4	1 status word

Simplified structure of the ERT_854_10 output register block with five 4x registers for the transfer of the SPS to the ERT
ERT_854_10 output register block:

Contents	Function
Transfer status	OUT transfer status (TS_OUT)
MUX 1	Time data block for the ERT for the clock synchronization
MUX 2	
MUX 3	
MUX 4	

Note: User interface is normally for the inputs and outputs of the ERT_854_10 EFB, not the 3x and 4x registers.

EFB configuration

EFB connection

The EFB connection to the input and output references (3x and 4x registers) is accomplished through a graphic connection to the ERT slot number, in the same way as with analog modules. The currently available QUANTUM and DROP EFBs from the ANA_IO library are used as follows: QUANTUM for local and DROP for remote backplanes. These EFBs transfer an integer index to every specified slot, which points to an internal data structure with the configured values. The module parameters and the ID are stored there, in addition to the addresses and lengths of the assigned input and output references (3x and 4x registers).
A significant improvement in the runtime can be achieved by deactivating the QUANTUM or the DROP EFB after the first execution. The average runtime of the ERT_854_10 EFB in a CPU x13-0x is approximately 0.6 ms, minimum 0.4 ms, maximum 1.6 ms. Every Quantum or DROP-EFB runs on average at approximately 1 ms, min. approx. 0.9 ms, max. approx. 1.3 ms.

Data Flow

Digital Inputs

No flag for new data is provided for this input type. The digital inputs in the first two input register words are updated every second cycle directly by the ERT. The EFB makes the processed values available as Bool if the BoolArr32 output field has been configured accordingly.

Counter Inputs

Cyclic updating of the counter values takes significantly longer than for other data types. Counter values are saved as a data record in "Cnt_Data" after a complete series (configured as: 8, 16, 24 or 32) of time consistent counter values in multiplex form has been transferred from the ERT. The flag for new data "ND_Count" is set for one cycle.

Event Inputs

As readiness to receive new events must be actively confirmed by the user, the management of the registers becomes somewhat more complex (a handshake mechanism is required). Event data remain in the data structure ERT_10_TTag and the flag for new data "ND_TT" stays set until the "ACK" input is set by the user and therefore requests a new event. The EFB responds to this by resetting "ND_TT" for at least one cycle. After the new event has been sent to the ERT_10_TT register structure, "ND_TT" is reset by the EFB. To prevent the new event data from being overwritten, the user must take care that the "ACK" input is reset after the EFB has reset the "ND_TT" flag. This state can then be kept stable to allow the user program enough time for event processing. Each subsequent event which is recorded with the ERT is temporarily stored within the event FIFO buffer.
New events are sent directly from the internal buffer of the EFB in intervals of at least 2 cycles for as long as the "ACK" input is set (for the special continuous operating mode); the effect is, however, that the "ND_TT" only stays set for one cycle. In this special mode, it is still the job of the user program to finish event processing before "ND_TT" signals the transfer of other new events to the ERT_10_TT structure because handshake protection by "ACK" is not available in this case.

ERT_10_TTag ERT_10_TTag event structure with 5 byte time stamps

Byte	Bits	Function
1	D0...D6 = Module No.. 0...127 D7 = CT	Rough time: CT = 1 indicates that this time stamp contains the whole time value including month and year in bytes 2 + 3. The Module no. can be set in any way in the parameter screen.
2	D0…D5 = input no. D6 = P1 D7 = P2	No. of the first input of the event group: 1...32 Type of the event message (P2, P1). 1.. 59 see *Note 1:, p. 22* [Month value if CT = 1]
3	D0…D7 = data from the event group (D7…D0 with right alignment)	1, 2 or 8 managed positions [Month value if CT = 1]
4	Time in milliseconds (least significant byte)	0 ... 59999 milliseconds (max. 61100) see *Note 2:, p. 23*
5	Time in milliseconds (most significant byte)	0 ... 59999 milliseconds (max. 61100) see *Note 2:, p. 23* and *Note 3:, p. 23*
6	D0...D5 = minutes D6 = R D7 = TI	Minutes: 0...59 Time invalid: TI = 1 means invalid time / reserved = 0 see *Note 3:, p. 23*
7	D0...D4 = hours D5 = R D6 = R D7 = DS	Hours: 0...23 Summer time: DS = 1 indicates that summer time is set With switchover from ST -> WT, hour 2A has ST, and hour 2B has WT
8	D0...D4 = DOW D5...D7 = DOM	Weekday: Mon-Sun = 1…7 Day of the month: 1...31 The code corresponds to CET and thus deviates from the standard used in the US, Sun = 1.

Note 1: Interpretation for Byte 2

D7 D6	Type of the event message	D5...D0	No. of the first input of the event group
0 1	1 pin message	1 ... 32	Input pin number
1 0	2 pin message	1, 3, 5, ...31	First input of the group
1 1	8 pin message	1, 9, 17, 25	First input of the group

Note 2: The value for the milliseconds is a maximum of 61100 ms with the second of transition (61000 plus a tolerance of 100 milliseconds)

Note 3: For time stamps containing an invalid time (TI = 1), the time in milliseconds is set to FFFF HEX. Minutes, hours and DOW/DOM values are invalid (i.e. undefined).

Rough Time Output If the "rough time declaration" has been activated during the ERT configuration, the transfer of the complete time (with month/year) is executed under the following conditions: when the month changes, after the module restarts, during every start or stop of the PLC user program, when the event FIFO buffer is deleted, when the clock is started or set. The transfer of this complete time output without the data input values is "triggered" basically takes place through a correct time stamped event. If this does not happen the values remain "stuck" in the ERT until an event occurs. Within the time stamp of a "rough time output", the CT bit is always set so that byte 2 contains the information about the month, byte 3 the information about the year and bytes 4 to 8 show the same time stamp values of the triggering event, which is immediately followed by the event message for rough time output.

Status Inputs The flag for new status data "ND_Stat" is set for one cycle. The status inputs can be overwritten after 2 query cycles.
The status word contains EFB and ERT error bits

Assignments of the Error Bits Internal structure of the EFB/ERT status word:

EFB error bits					**ERT error bits**										
D15 ...	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

ERT Error Bits

D8 ... D0 ERT error bits

Bit	Abbreviation	Meaning
D0	FW	Firmware errors, self test errors within EPROM, RAM or DPM (severe module errors)
D1	FP	Parameter errors (severe internal errors)
D2	TE	External time reference error (time-basis signal disrupted or not available)
D3	TU	Time became invalid
D4	TA	Time is not synchronized (Free running mode, permanent running without time error message, see also:*Without power reserve, p. 27*
D5	PF	FIFO buffer overflow (loss of the most recent event data)
D6	PH	FIFO buffer half full
D7	DC	Dechattering active (some event data lost)
D8	CE	ERT communication error (procedure errors or time out)

When configuring the *, p.* Sreens parameter, some of these errors can be assigned to grouped error messages with the "F" light as well as the module's error byte within the status table. All other errors are then defined as warnings.
D11 ... D9 reserved

EFB Error Bits

D15 ... D12 EFB error bits:

Bin.	Hex.	Meaning
1000	8 HEX	EFB communication time out
0101	5 HEX	Wrong slot
0110	6 HEX	Health status bit is not set (ERT appears not to be available)
other values		internal error

Online error display

The following ERT/ERB error messages are displayed in the **Online** → **Event viewer** Concept window with an error number and explanation.
EFB error messages:

Message	Error	Meaning
-2710	User error 11	EFB communication time out
-2711	[User error 12] =	EFB internal error
-2712	[User error 13] =	EFB internal error
-2713	[User error 14] =	EFB internal error
-2714	[User error 15] =	EFB internal error
-2715	[User error 16] =	Wrong slot
-2716	[User error 17]	Health status bit is not set (ERT appears not to be available)
-2717	[User error 18]	EFB internal error

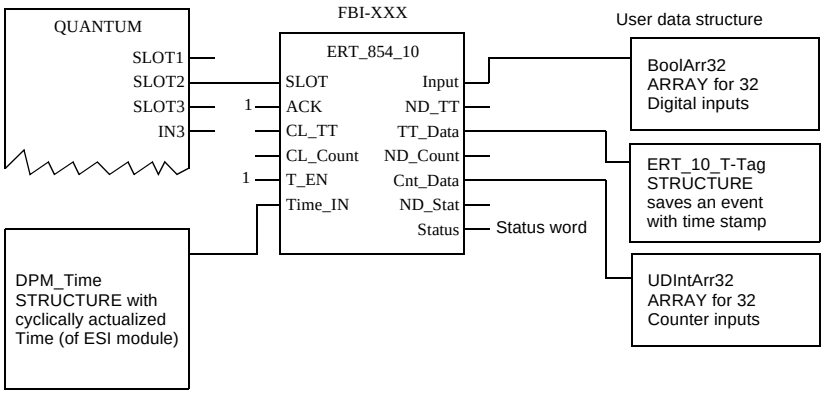
ERT error messages:

Message	Error	Meaning
-2700	User error 1]	ERT internal error
...	...	...
-2707	[User error 8]	ERT internal error
-2704	[User error 5]	ERT communication timeout (e.g. EFB disabled too long)

Simple example

Structure diagram

Principle structure



Other functions

Input markers

Setting the input marker "CL_TT" causes the FIFO buffer event of the ERT to be cleared. Setting the markers for one cycle is sufficient.
Setting the input marker "CL_Count" causes the ERT counter to be cleared by the ERT. Setting the markers for one cycle is sufficient.

Use of the DPM_Time Structure for the synchronization of the internal ERT clock

Time synchronization

If the time can not be synchronized through a standard time receiver, the time information can alternatively be transferred from the 140 ESI 062 01 communication module. The ESI makes the updated time available in a DPM_Time structure directly using the "Time_IN" parameter. The data structure can also be filled by the user program and the corresponding bits can be set. In this manner, the time can also be set, for example, by the CPU.

With power reserve

As soon as the "clock" parameter of the ERT is configured to "internal clock" with a power reserve not equal to zero (i.e. not free running), the EFB must use the time provided by the ESI for synchronizing the internal ERT clock. Until the first synchronization has taken place, the ERT sends back "status" output word with the bit "invalid time" set (Bit 3 TU).
The conditions of the first synchronization of the internal ERT using above the DPM_Time structure are:
The EFB Parameter "T_EN" must change from 0 to 1 to enable the time setting.

The time in "TIME_IN" provided by ESI must be represented as follows:

- valid (i.e. the bit for the message "time invalid" in "Min" value must not be set),
- and the values in "Ms" must change continually.

If, at a later point in time, the time data is invalid or no longer set, the TU changes to 1 after the configured power reserve has run out.

The synchronization/setting of the internal ERT clock takes place using the DPM_Time structure, if:

- EFB-Parameter "T_EN" is set to 1 to enable the time setting.
- The time data in "Time_IN" provided by ESI are valid (i.e. the "Time invalid" Bit in the "Min"value must not be set).
- The status of the DPM_Time element "Sync" changes from 0 to 1. This change is done every complete hour by the 140 ESI 062 01, but can also be triggered by a suitable telecontrol command.

The precision of the ESI and ERT synchronized time can be influenced by delay caused by the PLC cycle time, as well as by the cumulative components, which reflect the differences of the ERT software clock (< 360 milliseconds/second).

Without power reserve

If the "clock" parameter of the ERT was configured as an "internal clock" in free running mode (with a power reserve of zero), the internal clock starts with a default setting at hour 0 on 1/1/1990. In this case, the time can also be provided by using the DPM_Time data structure of the 140 ESI 062 01 module, as described above. As there is no power reserve to "run out", the time will never be invalid and the bit "Time not synchronized" is always set in the "status" output word (Bit 4 TA) which is returned by the EFB, .

Using the ERT >EFB time data flow

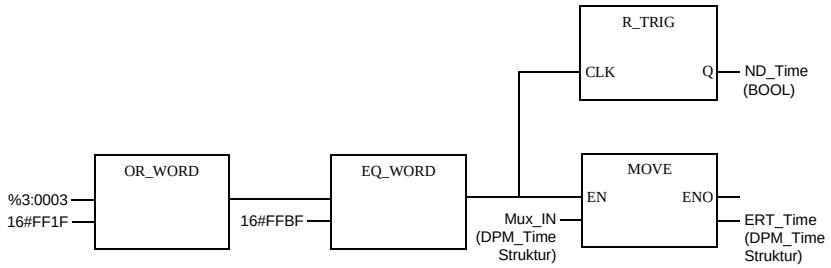
Application Examples:

This section shows an internal function which is made available by the ERT for diagnostics and development. It covers the cyclic transfer of the ERT internal time to the corresponding EFB in greater intervals. This time can be used for display or setting the PLC clock and so on, irrespective of whether it comes from the free-running internal clock or was synchronized through an external reference clock signal. The time appears as a DPM_Time structure beginning at word 4 of the IN register block of the ERT. The following diagram shows the program elements involved in selection.

Startup information:

During the I/O addressing, the IN references 30001 …30007 were assigned to an ERT_854_10. The IN transfer status (TS_IN) in the third word of the register block is sent to an OR_WORD block. A DPM_Time structure is defined in the variable editor as Variable Mux_IN in the fourth word of the IN register block and has address 30004 ... 30007. This variable is given as an input to the MOVE block. The MOVE block output is a DPM_Time structure defined by the variable editor as variable ERT_Time.

Typical recording mechanism for ERT time data



Note: The ERT_854_10-EFB must be active and error free.

Explanation:

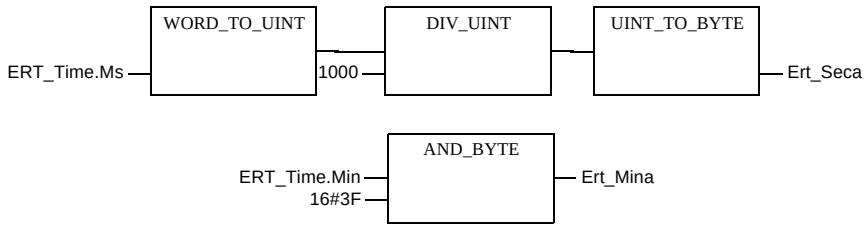
The MOVE block transfers the time data (which is cyclically stored in the MUX range of the IN register block) to the DPM_Time structure ERT_Time of the user as soon as the OR_WORD and the EQ_WORD block signal for a time data transfer. R_TRIG provides a signal in "ND_Time" for one cycle to allow further processing of the time data. The BOOL "Sync" element value of the ERT_Time should begin to "tick" during each new transfer from the ERT. There is a new transfer after a maximum of each 200 PLC cycles.

Example 1: Using time values for display (or with SET_TOD-EFB)

A number of simple logical operations is needed to obtain a meaningful display of the time information of the DPM_Time structure. The same commands can also be used for the ERT_10_T Tag structure. As example 2 deals with setting the PLC clock while using the SET_TOD-EFB, individual values are directly converted into the required formats.

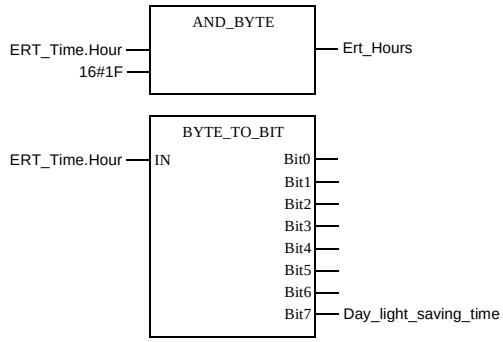
Note: The reference data editor (RDE) can provide the "ms" value directly in the Uns-Dec-WORD format and the "Min" value in the Dec-BYTE format.

SET_TOD requires that the WORD millisecond value "ms" is converted into a BYTE second value. The BYTE minute value "Min" contains the error bit which must be removed (values greater than 127 are invalid).
Conversion of the WORD millisecond value into a seconds BYTE

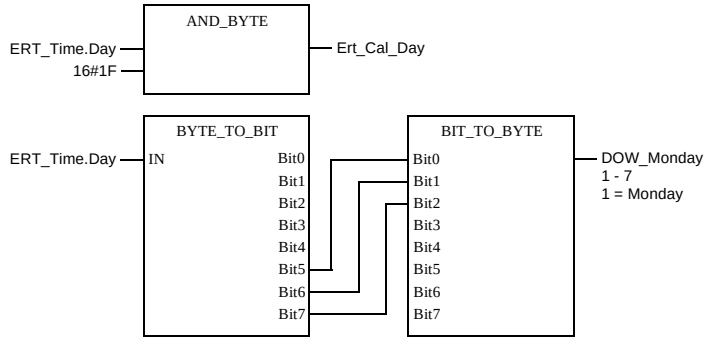


The BYTE value "Day" contains week and calendar day values. The weekday Monday is displayed as 1 in the DPM_Time structure. The weekday parameter in SET_TOD uses the value 1 for Sunday.

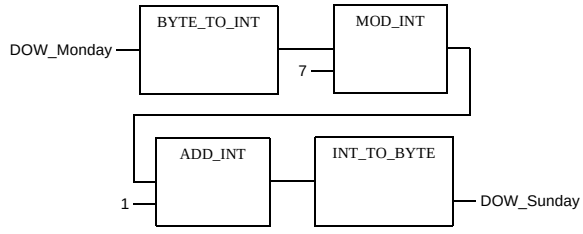
Removing/restoring the bit for the summer time of the "Hour" value.



The BYTE value "Day" contains week and calendar day values. The weekday Monday is displayed as 1 in the DPM_Time structure. The weekday parameter in SET_TOD uses the value 1 for Sunday.
Using the calendar day and weekday based on Monday



Further steps must be taken to convert the weekday based on the value of 1 for Monday into the value of 1 for Sunday.
Calculating the remainder values (Mod) and addition for converting the weekday values

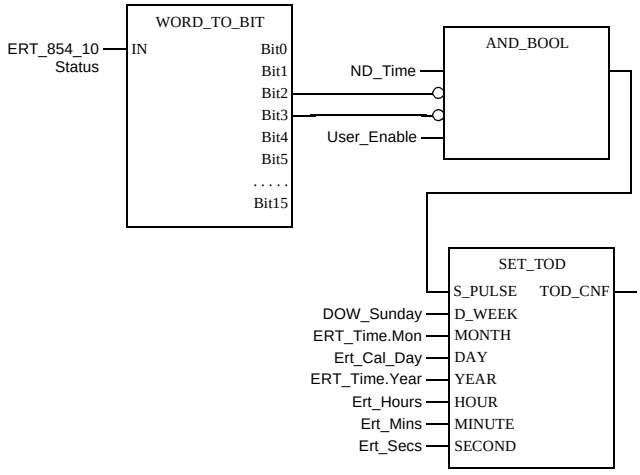


Example 2: Setting the PLC clock with the SET_TOD EFB while using ERT time data

All the parameter values required for the SET_TOD-EFB were created in example 1. The "ND_Time" signal required for transferring the time into the DPM_Time structure with the MOVE block is combined with a user enable here (e.g. only once per hour) to set the PLC clock only when new, error-free time data have been transferred by the ERT. (The ERT error bits are never set when the internal clock is in free run mode).
The SET_TOD-EFB is in the HSBY group of the SYSTEM block library. If it is used, the clock must be activated by storing the TIME OF DAY register in the SPECIALS range of the configuration with 4x addresses.

Note: The "status" parameter value is not exactly synchronized with the time data flow and for this reason can only "tend" to reflect the correct value.

User-enabled setting system for the PLC clock while using the SET_TOD-EFB



ERT_TIME: Time transfer to the ERT854

3

Overview

Introduction

This chapter describes the ERT_TIME block.

What's in this chapter?

This chapter contains the following topics:

Brief description

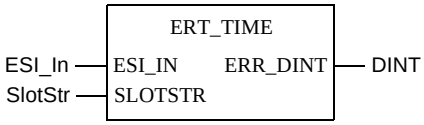
Function description

The function block reads the DCF time from the ESI 062 00 in order to enable synchronization of the internal clocks of all ERT modules in a TSX Quantum eliminating the need for the ERT modules having to be equipped with DCF receivers. The synchronization process is repeated hourly.
After the synchronization process, all ERT modules will show the same time. However, exact time synchronization with the "ESI time" cannot be achieved. Time deviation to the ESI 062 00 depends on the position of the block in the user program and on program runtime. There is maximum equality with program runtime if the block is located directly at program start.
The core of the function block are the ESI_IN and SLOTSTR parameters. ESI_IN is the parameter where the ESI 062 stores "its" DCF time, and the slots for all ERT modules that are to be synchronized with this time are determined in SLOTSTR.
The parameters EN and ENO can additionally be projected‘.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
ESI_IN	ESI_In	Data structure which stores the DCF time received by the ESI. Enter here the exact name used with the ESI 062 00.
SLOTSTR	SlotStr	Enter the slots of all the ERT modules whose time is to be synchronized with this block here. The data structure comprises 32 elements which the slot numbers of your ERT modules must be assigned to according to row. The numbers correspond to those in the I/O map. The remaining fields must be "0". Up to 14 ERT modules can be entered; any additional entries will be ignored.
ERR_DINT	DINT	The 32 bits are error bits for the ERT module indicated in SLOTSTR. Each bit corresponds to one element of SLOTSTR. The transfer has been free of errors if all bits = 0. Meaning of the bits: ● Bit 0 = 1: Error during transfer to 1st ERT 854 ● Bit 1 = 1: Error during transfer to 2nd ERT 854 ● Bit 3 = 1: ... **Note:** The bits are counted from right to left.

EXFR: Feedback data enable for Experts

4

Overview

Introduction

This chapter describes the EXFR block.

What's in this chapter?

This chapter contains the following topics:

Brief description

Function description

The function can be applied to the expert modules 140 ERT 854 00 and 140 ESI 062 00.
It will write a "0" to a byte in the 3x reference range of the expert (transfer status) in order to facilitate data transfer from an expert into State RAM of the PLC.
The write enable is revoked automatically after transfer completion, leaving the data write protected. EFB must be reinvoked before the next data transfer.

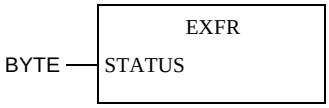
Note: Use this function only once in text languages (IL and ST), otherwise a faulty code will be generated.

The parameters EN and ENO can additionally be projected.

Representation

Symbol

Block representation:



Parameter description

Description of the block parameters:

Parameters	Data type	Meaning
STATUS	BYTE	Only the "tstat" structure element of the expert data structure should be referenced here. Example: "xxx.in.tstat" with xxx as the name of the variable for the derived data type ESI_In or ERT_In assigned in the Variable editor.

EXRB: Accepting feedback values from the expert

5

Overview

Introduction

This chapter describes the EXRB block.

What's in this chapter?

This chapter contains the following topics:

Brief description

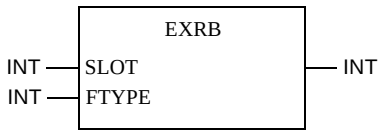
Function description

The function can be applied to the expert modules 140 ERT 000 00 and 140 ESI 062 00.
It copies expert feedback values and status data from the expert's dual port ram into the 3x-registers of the state RAM. This occurs directly upon execution of the EFB. Write access to the destination area must have been enabled by the "EXFR" EFB before the transfer.
The CPU automatically determines register addresses from the configuration data present in the I/O component list and experts dual port ram.
The parameters EN and ENO can additionally be projected.

Representation

Symbol

Block representation:



Parameter description

Description of the block parameters:

Parameters	Data type	Meaning
SLOT	INT	Module slot number.
FTYPE	INT	Field type to be read: 1 = Feedback data field 5 = status field 1 2, 6, 7 = reserved for future applications. Currently not to be used!
OUT	INT	Access execution status 0 = field was read without errors not equal to 0 = error during reading of field (see *Runtime error, p. 39*)

Runtime error

Error message

Error messages and their Significance

Error number	Meaning	Corrective Action
-2940	Invalid parameter	Examine the EFB parameter assignments
-2941	Internal error	Contact the hotline
-2942	Another task's I/O operation is still active.	Contact the hotline
-2943	Mode = end-transfer active.	Start the EFB again later
-2945	Destination area access not enabled	Enable destination area access with "EXFR"
-2946	Expert not mounted or in wrong slot	Examine the EFB parameter assignments and/or the I/O component list entry
-2947	Expert not connected or recognized	Check the hardware configuration, perform a hardware reset
-2948	Field is not configured	Examine the EFB parameter assignments
-2949	Internal error	Contact the hotline
-2950	No/wrong expert mounted.	check the hardware configuration
-2951	Wrong firmware or expert mode	check the hardware configuration
-2952	Synchronization error	Perform a hardware reset
-2953	Elementary data transfer error	Perform a hardware reset
-2954	Loadable "ULEX" not present or "ULEX" misbehavior.	Load "ULEX" or contact the hotline
-2955	Source buffer not in use	Start the EFB again later

EXWB: Transferring set points to the expert

6

Overview

Introduction

This chapter describes the EXWB block.

What's in this chapter?

This chapter contains the following topics:

Brief description

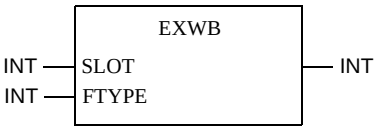
Function description

The function can be applied to the expert modules 140 ERT 000 00 and 140 ESI 062 00.
It copies expert command data from 4x- -Reference area of the PLC state memory into the expert's dual port ram. This occurs directly upon execution of the EFB.
The CPU automatically determines register addresses from the configuration data present in the I/O component list and experts dual port ram.
The parameters EN and ENO can additionally be projected.

Representation

Symbol

Block representation:



Parameter description

Description of the block parameters:

Parameters	Data type	Meaning
SLOT	INT	Module slot number.
FTYPE	INT	Field type to be written: 2 = command data field 1, 5, 6, 7 = reserved for future applications. Currently not to be used!
OUT	INT	0 = field was written without errors not equal to 0 = error occurred while field was being written (see Runtime errors of the EXRB block (See *Runtime error, p. 39*)).

Runtime error

Runtime error

Runtime error see EXRB (See *Runtime error, p. 39*).

MUX_DINTARR_125: Multiplexer for arrays of the data type DIntArr125

7

Overview

Introduction

This chapter describes the MUX_DINTARR_125 block.

What's in this chapter?

This chapter contains the following topics:

Brief description

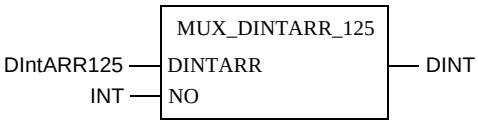
Function description

Use this function to address and select a single element in an array.
The parameters EN and ENO can additionally be projected.

Representation

Symbol

Block representation:



Parameter description

Description of the MUX_DINTARR_125: block parameter

Parameters	Data type	Meaning
DINTARR	DIntArr125	Array an element is to be selected from
NO	INT	Position in the array where the element to be selected is placed (range 0 ... 124)
OUT	DINT	selected element

Description of the DIntArr125 block parameter:

Element	Data type	Meaning
varname[1]	DINT	1. element of the array
...	...	...
varname[125]	DINT	125. element of the array

Runtime error

Error message

There will be an error message when the authorized value range for the parameter "NO" is violated. The error number is also entered at the output of the EFB.

MVB_IN: Data exchange between CPU and MVB-258A

8

Overview

Introduction

This chapter describes the MVB_IN block.

What's in this chapter?

This chapter contains the following topics:

Brief description

Function description

This function block realizes data exchange between MVB258A and CPU. The data block length in this case is subject to number and type of the variables. A data block with a maximum of 1024 words can be distributed onto 300 ports. Use the DATASNK data structure to copy the data block into the CPU.
Restrictions:

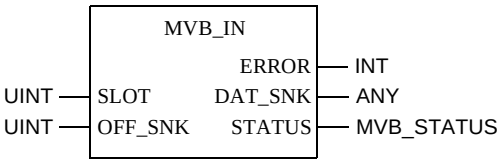
- Authorized word length of sink ports = 1024
- Up to 300 source and 300 sink ports can be addressed
- If not addressing any source ports, up to 500 sink ports can be configured
- The function block cannot be used with Concept simulators.

The parameters EN and ENO can additionally be projected.

Representation

Symbol

Block representation:



Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
SLOT	UINT	Enter the AS-BMVB-258A slot in the primary backplane here. Since the first two slots are occupied by the CPU, the communications module can only be installed on slots 3, 4 or 5.
OFF_SNK	UINT	An offset can be set using these parameters. This value is entered in bytes.
ERROR	INT	The ERROR output has four different signal states:0: 0: no errors 1: invalid slot address (only 3 -5 authorized in basic rack) 2: a wrong module has been placed onto this backplane slot 4: the data block together with the offset is greater than 1024 or 0
DATA_SNK	ANY	This parameter contains the input data structure (up to 1024 words). The default of the DATA_SNK parameter is data type ANY. In this case, either a self-defined data type or the predefined data type MVB_IN which is defined as an array of 1024 words, can be used.
STATUS	MVB_STATUS	The status parameter is an array of 32 words. Each bit from the array reflects the validity of a port. The bits are stored in ascending order of the ports. The status bits are updated for all configured ports. If the bit = "0", the port variable is valid. Accordingly, the port variable is invalid if the bit = "1". All unused bits are set to "1" and therefore invalid.

Detailed description

Runtime Optimization

In order to achieve optimum performance with regard to runtime routine of the MVB Function blocks, the number of words to be transferred must be adjusted according to the respective user program.
If there is no array optimization, all (1024) words in every program cycle of the user program are edited.

Word count adjustment

Word count configuration:

Step	Action
1	Open the EXPERTS.DTY file from **c:\Concept\Lib**.
2	Adjust the ARRAY ranges as required.
3	Save the changes using **File** → **Save** and close the file.

MVB_INFO: Requesting bus data via MVB

9

Overview

Introduction

This chapter describes the MVB_INFO block.

What's in this chapter?

This chapter contains the following topics:

Brief description

Function description

With the MVB_INFO function block, bus data can be requested via the MVB. This provides a view of information through the line, the configuration and through error messages.

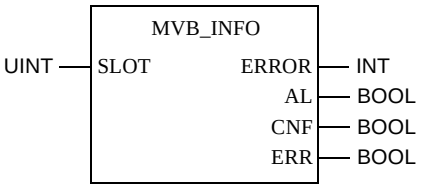
Note: The function block cannot be used with Concept simulators.

The parameters EN and ENO can additionally be projected.

Representation

Symbol

Block representation:



Parameter description

Description of the block parameters:

Parameters	Data type	Meaning
SLOT	UINT	Enter the ASB-MVB-258A slot in the primary backplane here. Since the first two slots are occupied by the CPU, the communications module can only be installed on slots 3, 4 or 5.
ERROR	INT	The MVB hardware configuration is checked with the ERROR parameter. The parameter displays three different states: ● 0: Error free function ● 1: An incorrect slot address has been entered ● 2: No communications module is installed on the parameterized slot
AL	BOOL	The AL parameter informs about the active line. AL=1 means Line_A is active.. AL= 1 means Line_2 is active.
CNF	BOOL	The CNF parameter provides the configuration status. A 0 value means, there is no configuration error, and the MVB I/O task is active. A 1 value means, there is a configuration error.

Detailed description

Runtime Optimization

In order to achieve optimum performance with regard to runtime routine of the MVB function blocks, the number of words to be transferred must be adjusted according to the respective user program.
If there is no array optimization, all (1024) words in every program cycle of the user program are edited.

Word count adjustment

Word count configuration

Step	Action
1	Open the file EXPERTS.DTY from **c:\Concept\Lib**
2	Adjust the ARRAY ranges as required.
3	Save the changes using **File** → **Save** and close the file.

MVB_OUT: Data exchange between AS-BMVB-258A and CPU

10

Overview

Introduction

This chapter describes the MVB_OUT block.

What's in this chapter?

This chapter contains the following topics:

Brief description

Function description

This function block realizes data exchange between MVB258A and CPU. The data block length in this case is subject to number and type of the variables. A data block with a maximum of 1024 words can be distributed onto 300 ports. The data packets are copied into the communications module using the DATA_SRC parameter.
Restrictions:

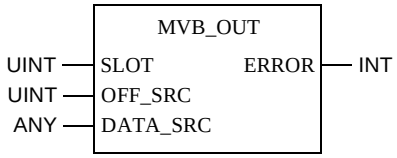
- The maximum word length of all source ports (incl. redundancy ports, if configured) = 1024 words.
- Up to 300 source and 300 sink ports can be addressed
- If not addressing any sink ports, up to 750 source ports can be configured
- The function block cannot be used with Concept simulators.

The parameters EN and ENO can additionally be projected.

Representation

Symbol

Block representation:



Parameter description

Description of the block parameters:

Parameters	Data type	Meaning
SLOT	UINT	Enter the AS-BMVB-258A slot in the primary backplane here. Since the first two slots are occupied by the CPU, the communications module can only be installed on slots 3, 4 or 5.
OFF_SRC	UINT	An offset can be set using these parameters. This value is entered in bytes.
DATA_SRC	ANY	This parameter contains the actual output data block (1024 words max.).
ERROR	INT	The ERROR parameter output has four different signal states: ● 0: no errors ● 1: invalid slot address (only 3 -5 authorized in basic rack) ● 2: a wrong module has been placed onto this backplane slot ● 4: the data block together with the offset is greater than 1024 or 0

Detailed description

Optimizing runtime

In order to achieve optimum performance with regard to runtime routine of the MVB Function blocks, the number of words that will be transferred has to be adjusted according to the respective user program.
If there is no array optimization, all (1024) words in every program cycle of the user program will be edited.

Adjustment of wordcount

Configuration of Wordcount

Step	Action
1	Open the EXPERTS.DTY file from **c:\Concept\Lib**.
2	Adjust the ARRAY range according to your own requirements.
3	Save the changes with **File** → **Save** and close the file.

MVB_RED: Switching redundant source ports

11

Overview

Introduction

This chapter describes the MVB_RED block.

What's in this chapter?

This chapter contains the following topics:

Brief description

Function description

MVB_RED is a function block for redundant source port switching. If there are defined redundant source ports, they can be actuated actively or passively using the EFBs.

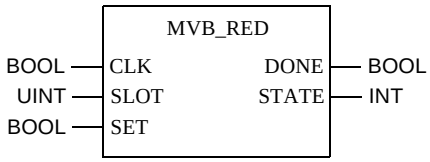
Note: The function block cannot be used with Concept simulators.

The parameters EN and ENO can additionally be projected.

Representation

Symbol

Block representation:



Parameter description

Description of the block parameters:

Parameters	Data type	Meaning
CLK	BOOL	The SET parameter acceptance is achieved through transition control. But the SET parameter is only analyzed if a positive transition is applied at the CLK parameter.
SLOT	UINT	Enter the MVB-258A slot in the primary backplane here. Since the first two slots are occupied by the CPU, the communications module can only be installed on slots 3, 4 or 5.
SET	BOOL	Use this parameter (bit) to actuate the ports actively (1) or passively (0).
DONT	BOOL	This parameter (response bit) uses status 0 or 1 to inform about the redundant port status. This means, if the status = "0", the redundant port is passively actuated, if it is = "1", it is actively actuated.

STATE	INT	The status or an error code is provided through the STATE parameter.
		The individual messages mean the following:
		● Status messages:
		  ● 0: Redundancy is not configured
		  ● 1: Switchover from passive to active
		  ● 2: Redundant ports are active
		  ● 3: Switchover from active to passive
		  ● 4: Redundant ports are passive
		● Error messages:
		  ● -1: An incorrect slot address has been entered
		  ● -2: No communications module is installed on the parametered slot

Detailed description

Runtime Optimization

In order to achieve optimum performance with regard to runtime routine of the MVB function blocks, the number of words to be transferred must be adjusted according to the respective user program.
If there is no array optimization, all (1024) words in every program cycle of the user program are edited.

Word count adjustment

Word count configuration:

Step	Action
1	Open the EXPERTS.DTY file from **c:\Concept\Lib**.
2	Adjust the ARRAY ranges as required.
3	Save the changes using **File** → **Save** and close the file.

SIMTSX: TSX Simulation

12

Overview

Introduction

This chapter describes the SIMTSX block.

What's in this chapter?

This chapter contains the following topics:

Brief description

Function description

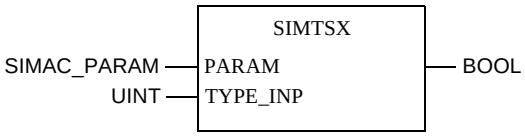
Note: This function is only for internal use.

The SIMTSX is a function used in conjunction with the SIMTSX software product to test and validate PLC programs.
The parameters EN and ENO can additionally be projected.

Representation

Symbol

Block representation:



Parameter description

Description of the block parameters:

Parameters	Data type	Meaning
PARAM	SIMAC_PARAM	3 4x registers
TYPE_INP	UINT	1 = 1x 3 = 3x
STATUS	BOOL	1 = execution OK

ULEXSTAT: Expert Status Signals

13

Overview

Introduction

This chapter describes the ULEXSTAT block.

What's in this chapter?

This chapter contains the following topics:

Brief description

Function description

The function block can be applied to the expert modules.

- 140 NOA 611 00,
- 140 NOA 611 10,
- 140 ERT 854 00 and
- 140 ESI 062 00.

It provides detailed information covering

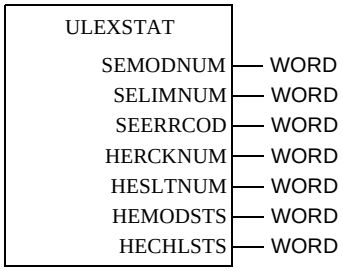
- hardware faults, recognized by the loadable "ULEX"
- software errors, occurring during the loadable "ULEX" execution

Should fault messages appear from several experts, the EFB always returns the status signals of the expert with the lowest slot number.
The parameters EN and ENO can additionally be projected.

Representation

Symbol

Block representation:

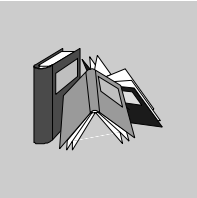


Parameter description

Description of the block parameters:

Parameters	Data type	Meaning
SEMODNUM	WORD	0: no software errors have occurred not equal to 0: Software errors. The number displayed is an aid to error localization. Please report when requested by the hotline.
SELIMNUM	WORD	See above.
SEERRCOD	WORD	See above.
HERCKNUM	WORD	0: Hardware faults were not noted. 1: Hardware fault recognized.
HESLTNUM	WORD	0: no hardware faults noted. 1 .. 16: Slot number, for which the hardware fault was recognized.
HEMODSTS	WORD	Module status. Identical with the "USERSTATUS" structure element in the "EXPSTATUS" data structure.
HECHLSTS	WORD	Module fault code. Identical with the "ERRNO" structure element in the "EXPSTATUS" data structure.

Glossary



A

Active window The window, which is currently selected. Only one window can be active at any one given time. When a window is active, the heading changes color, in order to distinguish it from other windows. Unselected windows are inactive.

Actual parameter Currently connected Input/Output parameters.

Addresses (Direct) addresses are memory areas on the PLC. These are found in the State RAM and can be assigned input/output modules.
The display/input of direct addresses is possible in the following formats:
- Standard format (400001)
- Separator format (4:00001)
- Compact format (4:1)
- IEC format (QW1)

ANL_IN ANL_IN stands for the data type "Analog Input" and is used for processing analog values. The 3x References of the configured analog input module, which is specified in the I/O component list is automatically assigned the data type and should therefore only be occupied by Unlocated variables.

ANL_OUT ANL_OUT stands for the data type "Analog Output" and is used for processing analog values. The 4x-References of the configured analog output module, which is specified in the I/O component list is automatically assigned the data type and should therefore only be occupied by Unlocated variables.

ANY In the existing version "ANY" covers the elementary data types BOOL, BYTE, DINT, INT, REAL, UDINT, UINT, TIME and WORD and therefore derived data types.

ANY_BIT In the existing version, "ANY_BIT" covers the data types BOOL, BYTE and WORD.

ANY_ELEM In the existing version "ANY_ELEM" covers the elementary data types BOOL, BYTE, DINT, INT, REAL, UDINT, UINT, TIME and WORD.

ANY_INT In the existing version, "ANY_INT" covers the data types DINT, INT, UDINT and UINT.

ANY_NUM In the existing version, "ANY_NUM" covers the data types DINT, INT, REAL, UDINT and UINT.

ANY_REAL In the existing version "ANY_REAL" covers the data type REAL.

Application window The window, which contains the working area, the menu bar and the tool bar for the application. The name of the application appears in the heading. An application window can contain several document windows. In Concept the application window corresponds to a Project.

Argument Synonymous with Actual parameters.

ASCII mode American Standard Code for Information Interchange. The ASCII mode is used for communication with various host devices. ASCII works with 7 data bits.

Atrium The PC based controller is located on a standard AT board, and can be operated within a host computer in an ISA bus slot. The module occupies a motherboard (requires SA85 driver) with two slots for PC104 daughter boards. From this, a PC104 daughter board is used as a CPU and the others for INTERBUS control.

B

Back up data file (Concept EFB) The back up file is a copy of the last Source files. The name of this back up file is "backup??.c" (it is accepted that there are no more than 100 copies of the source files. The first back up file is called "backup00.c". If changes have been made on the Definition file, which do not create any changes to the interface in the EFB, there is no need to create a back up file by editing the source files (**Objects** → **Source**). If a back up file can be assigned, the name of the source file can be given.

Base 16 literals

Base 16 literals function as the input of whole number values in the hexadecimal system. The base must be denoted by the prefix 16#. The values may not be preceded by signs (+/-). Single underline signs (_) between figures are not significant.

Example
16#F_F or 16#FF (decimal 255)
16#E_0 or 16#E0 (decimal 224)

Base 8 literal

Base 8 literals function as the input of whole number values in the octal system. The base must be denoted by the prefix 3.63kg. The values may not be preceded by signs (+/-). Single underline signs (_) between figures are not significant.

Example
8#3_1111 or 8#377 (decimal 255)
8#34_1111 or 8#340 (decimal 224)

Basis 2 literals

Base 2 literals function as the input of whole number values in the dual system. The base must be denoted by the prefix 0.91kg. The values may not be preceded by signs (+/-). Single underline signs (_) between figures are not significant.

Example
2#1111_1111 or 2#11111111 (decimal 255)
2#1110_1111 or 2#11100000 (decimal 224)

Binary connections

Connections between outputs and inputs of FFBs of data type BOOL.

Bit sequence

A data element, which is made up from one or more bits.

BOOL

BOOL stands for the data type "Boolean". The length of the data elements is 1 bit (in the memory contained in 1 byte). The range of values for variables of this type is 0 (FALSE) and 1 (TRUE).

Bridge

A bridge serves to connect networks. It enables communication between nodes on the two networks. Each network has its own token rotation sequence – the token is not deployed via bridges.

BYTE

BYTE stands for the data type "Bit sequence 8". The input appears as Base 2 literal, Base 8 literal or Base 1 16 literal. The length of the data element is 8 bit. A numerical range of values cannot be assigned to this data type.

C

Cache The cache is a temporary memory for cut or copied objects. These objects can be inserted into sections. The old content in the cache is overwritten for each new Cut or Copy.

Call up The operation, by which the execution of an operation is initiated.

Coil A coil is a LD element, which transfers (without alteration) the status of the horizontal link on the left side to the horizontal link on the right side. In this way, the status is saved in the associated Variable/ direct address.

Compact format (4:1) The first figure (the Reference) is separated from the following address with a colon (:), where the leading zero are not entered in the address.

Connection A check or flow of data connection between graphic objects (e.g. steps in the SFC editor, Function blocks in the FBD editor) within a section, is graphically shown as a line.

Constants Constants are Unlocated variables, which are assigned a value that cannot be altered from the program logic (write protected).

Contact A contact is a LD element, which transfers a horizontal connection status onto the right side. This status is from the Boolean AND- operation of the horizontal connection status on the left side with the status of the associated Variables/direct Address. A contact does not alter the value of the associated variables/direct address.

D

Data transfer settings Settings, which determine how information from the programming device is transferred to the PLC.

Data types

The overview shows the hierarchy of data types, as they are used with inputs and outputs of Functions and Function blocks. Generic data types are denoted by the prefix "ANY".

- ANY_ELEM
  - ANY_NUM
    ANY_REAL (REAL)
    ANY_INT (DINT, INT, UDINT, UINT)
  - ANY_BIT (BOOL, BYTE, WORD)
  - TIME
- System data types (IEC extensions)
- Derived (from "ANY" data types)

DCP I/O station

With a Distributed Control Processor (D908) a remote network can be set up with a parent PLC. When using a D908 with remote PLC, the parent PLC views the remote PLC as a remote I/O station. The D908 and the remote PLC communicate via the system bus, which results in high performance, with minimum effect on the cycle time. The data exchange between the D908 and the parent PLC takes place at 1.5 Megabits per second via the remote I/O bus. A parent PLC can support up to 31 (Address 2-32) D908 processors.

DDE (Dynamic Data Exchange)

The DDE interface enables a dynamic data exchange between two programs under Windows. The DDE interface can be used in the extended monitor to call up its own display applications. With this interface, the user (i.e. the DDE client) can not only read data from the extended monitor (DDE server), but also write data onto the PLC via the server. Data can therefore be altered directly in the PLC, while it monitors and analyzes the results. When using this interface, the user is able to make their own "Graphic-Tool", "Face Plate" or "Tuning Tool", and integrate this into the system. The tools can be written in any DDE supporting language, e.g. Visual Basic and Visual-C++. The tools are called up, when the one of the buttons in the dialog box extended monitor uses Concept Graphic Tool: Signals of a projection can be displayed as timing diagrams via the DDE connection between Concept and Concept Graphic Tool.

Decentral Network (DIO)

A remote programming in Modbus Plus network enables maximum data transfer performance and no specific requests on the links. The programming of a remote net is easy. To set up the net, no additional ladder diagram logic is needed. Via corresponding entries into the Peer Cop processor all data transfer requests are met.

Declaration

Mechanism for determining the definition of a Language element. A declaration normally covers the connection of an Identifier with a language element and the assignment of attributes such as Data types and algorithms.

Definition data file (Concept EFB)	The definition file contains general descriptive information about the selected FFB and its formal parameters.
Derived data type	Derived data types are types of data, which are derived from the Elementary data types and/or other derived data types. The definition of the derived data types appears in the data type editor in Concept. Distinctions are made between global data types and local data types.
Derived Function Block (DFB)	A derived function block represents the Call up of a derived function block type. Details of the graphic form of call up can be found in the definition " Function block (Item)". Contrary to calling up EFB types, calling up DFB types is denoted by double vertical lines on the left and right side of the rectangular block symbol. The body of a derived function block type is designed using FBD language, but only in the current version of the programming system. Other IEC languages cannot yet be used for defining DFB types, nor can derived functions be defined in the current version. Distinctions are made between local and global DFBs.
DINT	DINT stands for the data type "double integer". The input appears as Integer literal, Base 2 literal, Base 8 literal or Base 16 literal. The length of the data element is 32 bit. The range of values for variables of this data type is from –2 exp (31) to 2 exp (31) –1.
Direct display	A method of displaying variables in the PLC program, from which the assignment of configured memory can be directly and indirectly derived from the physical memory.
Document window	A window within an Application window. Several document windows can be opened at the same time in an application window. However, only one document window can be active. Document windows in Concept are, for example, sections, the message window, the reference data editor and the PLC configuration.
Dummy	An empty data file, which consists of a text header with general file information, i.e. author, date of creation, EFB identifier etc. The user must complete this dummy file with additional entries.
DX Zoom	This property enables connection to a programming object to observe and, if necessary, change its data value.

E

Elementary functions/ function blocks (EFB)
Identifier for Functions or Function blocks, whose type definitions are not formulated in one of the IEC languages, i.e. whose bodies, for example, cannot be modified with the DFB Editor (Concept-DFB). EFB types are programmed in "C" and mounted via Libraries in precompiled form.

EN / ENO (Enable / Error display)
If the value of EN is "0" when the FFB is called up, the algorithms defined by the FFB are not executed and all outputs contain the previous value. The value of ENO is automatically set to "0" in this case. If the value of EN is "1" when the FFB is called up, the algorithms defined by the FFB are executed. After the error free execution of the algorithms, the ENO value is automatically set to "1". If an error occurs during the execution of the algorithm, ENO is automatically set to "0". The output behavior of the FFB depends whether the FFBs are called up without EN/ENO or with EN=1. If the EN/ENO display is enabled, the EN input must be active. Otherwise, the FFB is not executed. The projection of EN and ENO is enabled/disabled in the block properties dialog box. The dialog box is called up via the menu commands **Objects** → **Properties...** or via a double click on the FFB.

Error
When processing a FFB or a Step an error is detected (e.g. unauthorized input value or a time error), an error message appears, which can be viewed with the menu command **Online** → **Event display...** . With FFBs the ENO output is set to "0".

Evaluation
The process, by which a value for a Function or for the outputs of a Function block during the Program execution is transmitted.

Expression
Expressions consist of operators and operands.

F

FFB (functions/ function blocks)
Collective term for EFB (elementary functions/function blocks) and DFB (derived function blocks)

Field variables
Variables, one of which is assigned, with the assistance of the key word ARRAY (field), a defined Derived data type. A field is a collection of data elements of the same Data type.

FIR filter
Finite Impulse Response Filter

Formal parameters

Input/Output parameters, which are used within the logic of a FFB and led out of the FFB as inputs/outputs.

Function (FUNC)

A Program organization unit, which exactly supplies a data element when executing. A function has no internal status information. Multiple call ups of the same function with the same input parameter values always supply the same output values. Details of the graphic form of function call up can be found in the definition " Function block (Item)". In contrast to the call up of function blocks, the function call ups only have one unnamed output, whose name is the name of the function itself. In FBD each call up is denoted by a unique number over the graphic block; this number is automatically generated and cannot be altered.

Function block (item) (FB)

A function block is a Program organization unit, which correspondingly calculates the functionality values, defined in the function block type description, for the output and internal variables, when it is called up as a certain item. All output values and internal variables of a certain function block item remain as a call up of the function block until the next. Multiple call up of the same function block item with the same arguments (Input parameter values) supply generally supply the same output value(s).
Each function block item is displayed graphically by a rectangular block symbol. The name of the function block type is located on the top center within the rectangle. The name of the function block item is located also at the top, but on the outside of the rectangle. An instance is automatically generated when creating, which can however be altered manually, if required. Inputs are displayed on the left side and outputs on the right of the block. The names of the formal input/output parameters are displayed within the rectangle in the corresponding places.
The above description of the graphic presentation is principally applicable to Function call ups and to DFB call ups. Differences are described in the corresponding definitions.

Function block dialog (FBD)

One or more sections, which contain graphically displayed networks from Functions, Function blocks and Connections.

Function block type

A language element, consisting of: 1. the definition of a data structure, subdivided into input, output and internal variables, 2. A set of operations, which is used with the elements of the data structure, when a function block type instance is called up. This set of operations can be formulated either in one of the IEC languages (DFB type) or in "C" (EFB type). A function block type can be instanced (called up) several times.

Function counter The function counter serves as a unique identifier for the function in a Program or DFB. The function counter cannot be edited and is automatically assigned. The function counter always has the structure: .n.m

n = Section number (number running)
m = Number of the FFB object in the section (number running)

G

Generic data type A Data type, which stands in for several other data types.

Generic literal If the Data type of a literal is not relevant, simply enter the value for the literal. In this case Concept automatically assigns the literal to a suitable data type.

Global derived data types Global Derived data types are available in every Concept project and are contained in the DFB directory directly under the Concept directory.

Global DFBs Global DFBs are available in every Concept project and are contained in the DFB directory directly under the Concept directory.

Global macros Global Macros are available in every Concept project and are contained in the DFB directory directly under the Concept directory.

Groups (EFBs) Some EFB libraries (e.g. the IEC library) are subdivided into groups. This facilitates the search for FFBs, especially in extensive libraries.

I

I/O component list The I/O and expert assemblies of the various CPUs are configured in the I/O component list.

IEC 61131-3 International norm: Programmable controllers – part 3: Programming languages.

IEC format (QW1)

In the place of the address stands an IEC identifier, followed by a five figure address:

- %0x12345 = %Q12345
- %1x12345 = %I12345
- %3x12345 = %IW12345
- %4x12345 = %QW12345

IEC name conventions (identifier)

An identifier is a sequence of letters, figures, and underscores, which must start with a letter or underscores (e.g. name of a function block type, of an item or section). Letters from national sets of characters (e.g. ö,ü, é, õ) can be used, taken from project and DFB names.
Underscores are significant in identifiers; e.g. "A_BCD" and "AB_CD" are interpreted as different identifiers. Several leading and multiple underscores are not authorized consecutively.
Identifiers are not permitted to contain space characters. Upper and/or lower case is not significant; e.g. "ABCD" and "abcd" are interpreted as the same identifier.
Identifiers are not permitted to be Key words.

IIR filter

Infinite Impulse Response Filter

Initial step (starting step)

The first step in a chain. In each chain, an initial step must be defined. The chain is started with the initial step when first called up.

Initial value

The allocated value of one of the variables when starting the program. The value assignment appears in the form of a Literal.

Input bits (1x references)

The 1/0 status of input bits is controlled via the process data, which reaches the CPU from an entry device.

Note: The x, which comes after the first figure of the reference type, represents a five figure storage location in the application data store, i.e. if the reference 100201 signifies an input bit in the address 201 of the State RAM.

Input parameters (Input)

When calling up a FFB the associated Argument is transferred.

Input words (3x references)

An input word contains information, which come from an external source and are represented by a 16 bit figure. A 3x register can also contain 16 sequential input bits, which were read into the register in binary or BCD (binary coded decimal) format.
Note: The x, which comes after the first figure of the reference type, represents a five figure storage location in the user data store, i.e. if the reference 300201 signifies a 16 bit input word in the address 201 of the State RAM.

Instantiation

The generation of an Item.

Instruction (IL)

Instructions are "commands" of the IL programming language. Each operation begins on a new line and is succeeded by an operator (with modifier if needed) and, if necessary for each relevant operation, by one or more operands. If several operands are used, they are separated by commas. A tag can stand before the instruction, which is followed by a colon. The commentary must, if available, be the last element in the line.

Instruction (LL984)

When programming electric controllers, the task of implementing operational coded instructions in the form of picture objects, which are divided into recognizable contact forms, must be executed. The designed program objects are, on the user level, converted to computer useable OP codes during the loading process. The OP codes are deciphered in the CPU and processed by the controller's firmware functions so that the desired controller is implemented.

Instruction list (IL)

IL is a text language according to IEC 1131, in which operations, e.g. conditional/ unconditional call up of Function blocks and Functions, conditional/unconditional jumps etc. are displayed through instructions.

INT

INT stands for the data type "whole number". The input appears as Integer literal, Base 2 literal, Base 8 literal or Base 16 literal. The length of the data element is 16 bit. The range of values for variables of this data type is from –2 exp (15) to 2 exp (15) –1.

Integer literals

Integer literals function as the input of whole number values in the decimal system. The values may be preceded by the signs (+/-). Single underline signs (_) between figures are not significant.

Example
-12, 0, 123_456, +986

INTERBUS (PCP)

To use the INTERBUS PCP channel and the INTERBUS process data preprocessing (PDP), the new I/O station type INTERBUS (PCP) is led into the Concept configurator. This I/O station type is assigned fixed to the INTERBUS connection module 180-CRP-660-01.
The 180-CRP-660-01 differs from the 180-CRP-660-00 only by a clearly larger I/O area in the state RAM of the controller.

Item name

An Identifier, which belongs to a certain Function block item. The item name serves as a unique identifier for the function block in a program organization unit. The item name is automatically generated, but can be edited. The item name must be unique throughout the Program organization unit, and no distinction is made between upper/lower case. If the given name already exists, a warning is given and another name must be selected. The item name must conform to the IEC name conventions, otherwise an error message appears. The automatically generated instance name always has the structure: FBI_n_m

FBI = Function block item
n = Section number (number running)
m = Number of the FFB object in the section (number running)

J

Jump

Element of the SFC language. Jumps are used to jump over areas of the chain.

K

Key words

Key words are unique combinations of figures, which are used as special syntactic elements, as is defined in appendix B of the IEC 1131-3. All key words, which are used in the IEC 1131-3 and in Concept, are listed in appendix C of the IEC 1131-3. These listed keywords cannot be used for any other purpose, i.e. not as variable names, section names, item names etc.

L

Ladder Diagram (LD)

Ladder Diagram is a graphic programming language according to IEC1131, which optically orientates itself to the "rung" of a relay ladder diagram.

Ladder Logic 984 (LL)

In the terms Ladder Logic and Ladder Diagram, the word Ladder refers to execution. In contrast to a diagram, a ladder logic is used by engineers to draw up a circuit (with assistance from electrical symbols),which should chart the cycle of events and not the existing wires, which connect the parts together. A usual user interface for controlling the action by automated devices permits ladder logic interfaces, so that when implementing a control system, engineers do not have to learn any new programming languages, with which they are not conversant.
The structure of the actual ladder logic enables electrical elements to be linked in a way that generates a control output, which is dependant upon a configured flow of power through the electrical objects used, which displays the previously demanded condition of a physical electric appliance.
In simple form, the user interface is one of the video displays used by the PLC programming application, which establishes a vertical and horizontal grid, in which the programming objects are arranged. The logic is powered from the left side of the grid, and by connecting activated objects the electricity flows from left to right.

Landscape format

Landscape format means that the page is wider than it is long when looking at the printed text.

Language element

Each basic element in one of the IEC programming languages, e.g. a Step in SFC, a Function block item in FBD or the Start value of a variable.

Library

Collection of software objects, which are provided for reuse when programming new projects, or even when building new libraries. Examples are the Elementary function block types libraries.
EFB libraries can be subdivided into Groups.

Literals

Literals serve to directly supply values to inputs of FFBs, transition conditions etc. These values cannot be overwritten by the program logic (write protected). In this way, generic and standardized literals are differentiated.
Furthermore literals serve to assign a Constant a value or a Variable an Initial value. The input appears as Base 2 literal, Base 8 literal, Base 16 literal, Integer literal, Real literal or Real literal with exponent.

Local derived data types

Local derived data types are only available in a single Concept project and its local DFBs and are contained in the DFB directory under the project directory.

Local DFBs

Local DFBs are only available in a single Concept project and are contained in the DFB directory under the project directory.

Local link

The local network link is the network, which links the local nodes with other nodes either directly or via a bus amplifier.

Local macros

Local Macros are only available in a single Concept project and are contained in the DFB directory under the project directory.

Local network nodes

The local node is the one, which is projected evenly.

Located variable

Located variables are assigned a state RAM address (reference addresses 0x,1x, 3x, 4x). The value of these variables is saved in the state RAM and can be altered online with the reference data editor. These variables can be addressed by symbolic names or the reference addresses.

Collective PLC inputs and outputs are connected to the state RAM. The program access to the peripheral signals, which are connected to the PLC, appears only via located variables. PLC access from external sides via Modbus or Modbus plus interfaces, i.e. from visualizing systems, are likewise possible via located variables.

M

Macro

Macros are created with help from the software Concept DFB.
Macros function to duplicate frequently used sections and networks (including the logic, variables, and variable declaration).
Distinctions are made between local and global macros.

Macros have the following properties:

- Macros can only be created in the programming languages FBD and LD.
- Macros only contain one single section.
- Macros can contain any complex section.
- From a program technical point of view, there is no differentiation between an instanced macro, i.e. a macro inserted into a section, and a conventionally created macro.
- Calling up DFBs in a macro
- Variable declaration
- Use of macro-own data structures
- Automatic acceptance of the variables declared in the macro
- Initial value for variables
- Multiple instancing of a macro in the whole program with different variables
- The section name, the variable name and the data structure name can contain up to 10 different exchange markings (@0 to @9).

MMI

Man Machine Interface

Multi element variables

Variables, one of which is assigned a Derived data type defined with STRUCT or ARRAY.
Distinctions are made between Field variables and structured variables.

N

Network A network is the connection of devices to a common data path, which communicate with each other via a common protocol.

Network node A node is a device with an address (164) on the Modbus Plus network.

Node address The node address serves a unique identifier for the network in the routing path. The address is set directly on the node, e.g. with a rotary switch on the back of the module.

O

Operand An operand is a Literal, a Variable, a Function call up or an Expression.

Operator An operator is a symbol for an arithmetic or Boolean operation to be executed.

Output parameters (Output) A parameter, with which the result(s) of the Evaluation of a FFB are returned.

Output/discretes (0x references) An output/marker bit can be used to control real output data via an output unit of the control system, or to define one or more outputs in the state RAM. Note: The x, which comes after the first figure of the reference type, represents a five figure storage location in the application data store, i.e. if the reference 000201 signifies an output or marker bit in the address 201 of the State RAM.

Output/marker words (4x references) An output/marker word can be used to save numerical data (binary or decimal) in the State RAM, or also to send data from the CPU to an output unit in the control system. Note: The x, which comes after the first figure of the reference type, represents a five figure storage location in the application data store, i.e. if the reference 400201 signifies a 16 bit output or marker word in the address 201 of the State RAM.

P

Peer processor The peer processor processes the token run and the flow of data between the Modbus Plus network and the PLC application logic.

PLC Programmable controller

Program The uppermost Program organization unit. A program is closed and loaded onto a single PLC.

Program cycle A program cycle consists of reading in the inputs, processing the program logic and the output of the outputs.

Program organization unit A Function, a Function block, or a Program. This term can refer to either a Type or an Item.

Programming device Hardware and software, which supports programming, configuring, testing, implementing and error searching in PLC applications as well as in remote system applications, to enable source documentation and archiving. The programming device could also be used for process visualization.

Programming redundancy system (Hot Standby) A redundancy system consists of two identically configured PLC devices, which communicate with each other via redundancy processors. In the case of the primary PLC failing, the secondary PLC takes over the control checks. Under normal conditions the secondary PLC does not take over any controlling functions, but instead checks the status information, to detect mistakes.

Project General identification of the uppermost level of a software tree structure, which specifies the parent project name of a PLC application. After specifying the project name, the system configuration and control program can be saved under this name. All data, which results during the creation of the configuration and the program, belongs to this parent project for this special automation.
General identification for the complete set of programming and configuring information in the Project data bank, which displays the source code that describes the automation of a system.

Project data bank The data bank in the Programming device, which contains the projection information for a Project.

Prototype data file (Concept EFB)

The prototype data file contains all prototypes of the assigned functions. Further, if available, a type definition of the internal

R

REAL

REAL stands for the data type "real". The input appears as Real literal or as Real literal with exponent. The length of the data element is 32 bit. The value range for variables of this data type reaches from 8.43E-37 to 3.36E+38.

Note: Depending on the mathematic processor type of the CPU, various areas within this valid value range cannot be represented. This is valid for values nearing ZERO and for values nearing INFINITY. In these cases, a number value is not shown in animation, instead NAN (**N**ot **A N**umber) oder INF (**INF**inite).

Real literal

Real literals function as the input of real values in the decimal system. Real literals are denoted by the input of the decimal point. The values may be preceded by the signs (+/-). Single underline signs (_) between figures are not significant.

Example
-12.0, 0.0, +0.456, 3.14159_26

Real literal with exponent

Real literals with exponent function as the input of real values in the decimal system. Real literals with exponent are denoted by the input of the decimal point. The exponent sets the key potency, by which the preceding number is multiplied to get to the value to be displayed. The basis may be preceded by a negative sign (-). The exponent may be preceded by a positive or negative sign (+/-). Single underline signs (_) between figures are not significant. (Only between numbers, not before or after the decimal poiont and not before or after "E", "E+" or "E-")

Example
-1.34E-12 or -1.34e-12
1.0E+6 or 1.0e+6
1.234E6 or 1.234e6

Reference

Each direct address is a reference, which starts with an ID, specifying whether it concerns an input or an output and whether it concerns a bit or a word. References, which start with the code 6, display the register in the extended memory of the state RAM.
0x area = Discrete outputs
1x area = Input bits
3x area = Input words
4x area = Output bits/Marker words
6x area = Register in the extended memory

Note: The x, which comes after the first figure of each reference type, represents a five figure storage location in the application data store, i.e. if the reference 400201 signifies a 16 bit output or marker word in the address 201 of the State RAM.

Register in the extended memory (6x reference)

6x references are marker words in the extended memory of the PLC. Only LL984 user programs and CPU 213 04 or CPU 424 02 can be used.

RIO (Remote I/O)

Remote I/O provides a physical location of the I/O coordinate setting device in relation to the processor to be controlled. Remote inputs/outputs are connected to the consumer control via a wired communication cable.

RP (PROFIBUS)

RP = Remote Peripheral

RTU mode

Remote Terminal Unit
The RTU mode is used for communication between the PLC and an IBM compatible personal computer. RTU works with 8 data bits.

Rum-time error

Error, which occurs during program processing on the PLC, with SFC objects (i.e. steps) or FFBs. These are, for example, over-runs of value ranges with figures, or time errors with steps.

S

SA85 module The SA85 module is a Modbus Plus adapter for an IBM-AT or compatible computer.

Section A section can be used, for example, to describe the functioning method of a technological unit, such as a motor.
A Program or DFB consist of one or more sections. Sections can be programmed with the IEC programming languages FBD and SFC. Only one of the named programming languages can be used within a section.
Each section has its own Document window in Concept. For reasons of clarity, it is recommended to subdivide a very large section into several small ones. The scroll bar serves to assist scrolling in a section.

Separator format (4:00001) The first figure (the Reference) is separated from the ensuing five figure address by a colon (:).

Sequence language (SFC) The SFC Language elements enable the subdivision of a PLC program organizational unit in a number of Steps and Transitions, which are connected horizontally by aligned Connections. A number of actions belong to each step, and a transition condition is linked to a transition.

Serial ports With serial ports (COM) the information is transferred bit by bit.

Source code data file (Concept EFB) The source code data file is a usual C++ source file. After execution of the menu command **Library** → **Generate data files** this file contains an EFB code framework, in which a specific code must be entered for the selected EFB. To do this, click on the menu command **Objects** → **Source**.

Standard format (400001) The five figure address is located directly after the first figure (the reference).

Standardized literals If the data type for the literal is to be automatically determined, use the following construction: 'Data type name'#'Literal value'.

Example
INT#15 (Data type: Integer, value: 15),
BYTE#00001111 (data type: Byte, value: 00001111)
REAL#23.0 (Data type: Real, value: 23.0)

For the assignment of REAL data types, there is also the possibility to enter the value in the following way: 23.0.
Entering a comma will automatically assign the data type REAL.

State RAM The state RAM is the storage for all sizes, which are addressed in the user program via References (Direct display). For example, input bits, discretes, input words, and discrete words are located in the state RAM.

Statement (ST) Instructions are "commands" of the ST programming language. Instructions must be terminated with semicolons. Several instructions (separated by semi-colons) can occupy the same line.

Status bits There is a status bit for every node with a global input or specific input/output of Peer Cop data. If a defined group of data was successfully transferred within the set time out, the corresponding status bit is set to 1. Alternatively, this bit is set to 0 and all data belonging to this group (of 0) is deleted.

Step SFC Language element: Situations, in which the Program behavior follows in relation to the inputs and outputs of the same operations, which are defined by the associated actions of the step.

Step name The step name functions as the unique flag of a step in a Program organization unit. The step name is automatically generated, but can be edited. The step name must be unique throughout the whole program organization unit, otherwise an Error message appears.
The automatically generated step name always has the structure: S_n_m

S = Step
n = Section number (number running)
m = Number of steps in the section (number running)

Structured text (ST) ST is a text language according to IEC 1131, in which operations, e.g. call up of Function blocks and Functions, conditional execution of instructions, repetition of instructions etc. are displayed through instructions.

Structured variables Variables, one of which is assigned a Derived data type defined with STRUCT (structure).
A structure is a collection of data elements with generally differing data types (Elementary data types and/or derived data types).

SY/MAX In Quantum control devices, Concept closes the mounting on the I/O population SY/MAX I/O modules for RIO control via the Quantum PLC with on. The SY/MAX remote subrack has a remote I/O adapter in slot 1, which communicates via a Modicon S908 R I/O system. The SY/MAX I/O modules are performed when highlighting and including in the I/O population of the Concept configuration.

Symbol (Icon) Graphic display of various objects in Windows, e.g. drives, user programs and Document windows.

T

Template data file (Concept EFB)

The template data file is an ASCII data file with a layout information for the Concept FBD editor, and the parameters for code generation.

TIME

TIME stands for the data type "Time span". The input appears as Time span literal. The length of the data element is 32 bit. The value range for variables of this type stretches from 0 to 2exp(32)-1. The unit for the data type TIME is 1 ms.

Time span literals

Permitted units for time spans (TIME) are days (D), hours (H), minutes (M), seconds (S) and milliseconds (MS) or a combination thereof. The time span must be denoted by the prefix t#, T#, time# or TIME#. An "overrun" of the highest ranking unit is permitted, i.e. the input T#25H15M is permitted.

Example
t#14MS, T#14.7S, time#18M, TIME#19.9H, t#20.4D, T#25H15M, time#5D14H12M18S3.5MS

Token

The network "Token" controls the temporary property of the transfer rights via a single node. The token runs through the node in a circulating (rising) address sequence. All nodes track the Token run through and can contain all possible data sent with it.

Traffic Cop

The Traffic Cop is a component list, which is compiled from the user component list. The Traffic Cop is managed in the PLC and in addition contains the user component list e.g. Status information of the I/O stations and modules.

Transition

The condition with which the control of one or more Previous steps transfers to one or more ensuing steps along a directional Link.

U

UDEFB
User defined elementary functions/function blocks
Functions or Function blocks, which were created in the programming language C, and are available in Concept Libraries.

UDINT
UDINT stands for the data type "unsigned double integer". The input appears as Integer literal, Base 2 literal, Base 8 literal or Base 16 literal. The length of the data element is 32 bit. The value range for variables of this type stretches from 0 to 2exp(32)-1.

UINT
UINT stands for the data type "unsigned integer". The input appears as Integer literal, Base 2 literal, Base 8 literal or Base 16 literal. The length of the data element is 16 bit. The value range for variables of this type stretches from 0 to (2exp16)-1.

Unlocated variable
Unlocated variables are not assigned any state RAM addresses. They therefore do not occupy any state RAM addresses. The value of these variables is saved in the system and can be altered with the reference data editor. These variables are only addressed by symbolic names.

Signals requiring no peripheral access, e.g. intermediate results, system tags etc, should primarily be declared as unlocated variables.

V

Variables
Variables function as a data exchange within sections between several sections and between the Program and the PLC.
Variables consist of at least a variable name and a Data type.
Should a variable be assigned a direct Address (Reference), it is referred to as a Located variable. Should a variable not be assigned a direct address, it is referred to as an unlocated variable. If the variable is assigned a Derived data type, it is referred to as a Multi-element variable.
Otherwise there are Constants and Literals.

Vertical format
Vertical format means that the page is higher than it is wide when looking at the printed text.

W

Warning When processing a FFB or a Step a critical status is detected (e.g. critical input value or a time out), a warning appears, which can be viewed with the menu command **Online** → **Event viewer...** . With FFBs the ENO output remains at "1".

WORD WORD stands for the data type "Bit sequence 16". The input appears as Base 2 literal, Base 8 literal or Base 1 16 literal. The length of the data element is 16 bit. A numerical range of values cannot be assigned to this data type.

Index

I

M

O

P

R

S

T

U