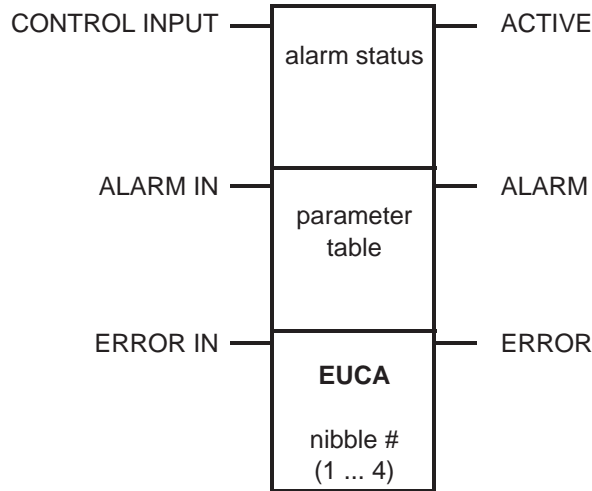


Representation: EUCA - Engineering Unit and Alarm

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON initiates the conversion
Middle input	0x, 1x	None	Alarm input
Bottom input	0x, 1x	None	Error input
alarm status (top node)	4x	INT, UINT	Alarm status for as many as four EUCA conversions (For more information please see <i>p. 492.</i>)
parameter table (middle node)	4x	INT, UINT,	First of nine contiguous holding registers in the EUCA parameter table (For more information please see <i>p. 493.</i>)
nibble # (1...4) (bottom node)		INT, UINT	Integer value, indicates which one of the four nibbles in the alarm status register to use
Top output	0x	None	Echoes the state of the top input
Middle output	0x	None	ON if the middle input is ON or if the result of the EUCA conversion crosses a warning level
Bottom output	0x	None	ON if the bottom input is ON or if a parameter is out of range

Parameter Description

Alarm Status (Top Node)

The 4x register entered in the top node displays the alarm status for as many as four EUCA conversions, which can be performed by the instruction. The register is segmented into four four-bit nibbles. Each four-bit nibble represents the four possible alarm conditions for an individual EUCA conversion.

The most significant nibble represents the first conversion, and the least significant nibble represents the fourth conversion:

HA1	HW1	LW1	LA1	HA2	HW2	LW2	LA2	HA3	HW3	LW3	LA3	HA4	HW4	LW4	LA4
Nibble 1 (first conversion)				Nibble 2 (second conversion)				Nibble 3 (third conversion)				Nibble 4 (fourth conversion)			

Alarm Setting

Condition of alarm setting

Alarm type	Condition
HA	An HA alarm is set when the SPV exceeds the user-defined high alarm value expressed in engineering units
HW	An HW alarm is set when SPV exceeds a user-defined high warning value expressed in engineering units
LW	An LW alarm is set when SPV is less than a user-defined low warning value expressed in engineering units
LA	An LA alarm is set when SPV is less than a user-defined low alarm value expressed in engineering units

Only one alarm condition can exist in any EUCA conversion at any given time. If the SPV exceeds the high warning level the HW bit will be set. If the HA is exceeded, the HW bit is cleared and the HA bit is set. The alarm bit will not change after returning to a less severe condition until the deadband (DB) area has also been exited.

**Parameter Table
(Middle Node)**

The 4x register entered in the middle node is the first of nine contiguous holding registers in the EUCA parameter table:

Register	Content	Range
Displayed	Binary value input by the user	0 ... 4 095
First implied	SPV calculated by the EUCA block	
Second implied	High engineering unit (HEU), maximum SPV required and set by the user (top of the scale)	$LEU < HEU \leq 99\,999$
Third implied	Low engineering unit (LEU), minimum SPV required and set by the user (bottom end of the scale)	$0 \leq LEU < HEU$
Fourth implied	DB area in SPV units, below HA levels and above LA levels that must be crossed before the alarm status bit will reset	$0 \leq DB < (HEU - LEU)$
Fifth implied	HA alarm value in SPV units	$HW < HA \leq HEU$
Sixth implied	HW alarm value in SPV units	$LW < HW < HA$
Seventh implied	LW alarm value in SPV units	$LA < LW < HW$
Eighth implied	LA alarm value in SPV units	$LEU \leq LA < LW$

Note: An error is generated if any value is out of the range defined above

Examples

Overview

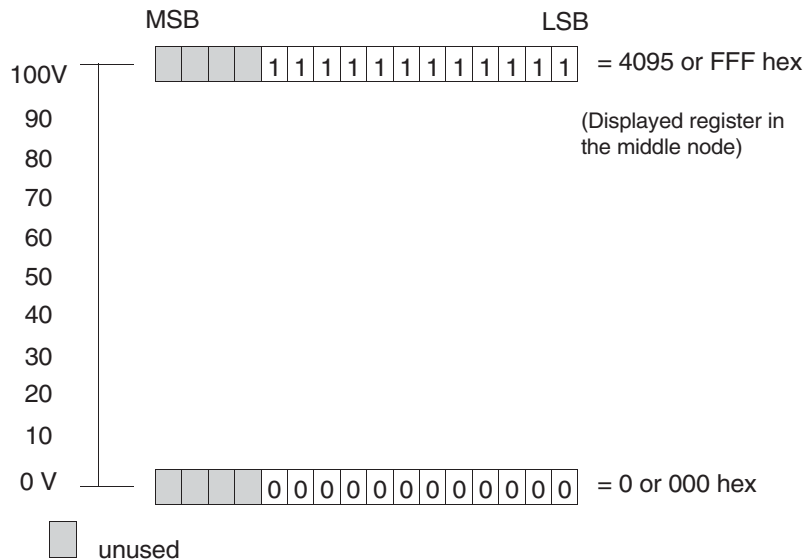
The following examples are shown.

- Principles of EUCA Operation (example 1)
 - Use in a Drive System (example 2)
 - Four EUCA conversions together (example 3)
-

Example 1

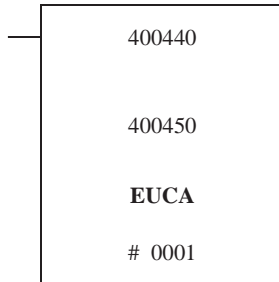
This example demonstrates the principles of EUCA operation. The binary value is manually input in the displayed register in the middle node, and the result is visually available in the SPV register (the first implied register in the middle node).

The illustration below shows an input range equivalent of a 0 ... 100 V measure, corresponding to the whole binary 12-bit range:



A range of 0 ... 100 V establishes 50 V for nominal operation. EUCA provides a margin on the nominal side of both warning and alarm levels (deadband). If an alarm threshold is exceeded, the alarm bit becomes active and stays active until the signal becomes greater (or less) than the DB setting -5 V in this example.

Programming the EUCA block is accomplished by selecting the EUCA loadable and writing in the data as illustrated in the figure below:

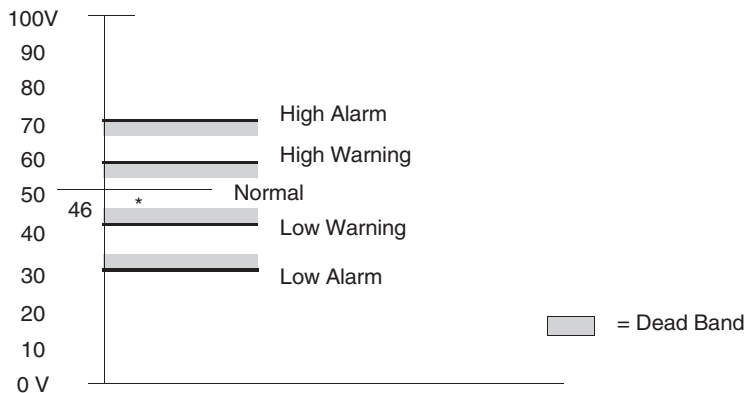


Reference Data

Register	Meaning	Content
400440	STATUS	0000000000000000
400450	INPUT	1871 DEC
400451	SPV	46 DEC
400452	HIGH_unit	100 DEC
400453	LOW_unit	0 DEC
400454	Dead_band	5 DEC
400455	HIGH_ALARM	70 DEC
400456	HIGH_WARN	60 DEC
400457	LOW_ALARM	40 DEC
400458	LOW_WARN	30 DEC

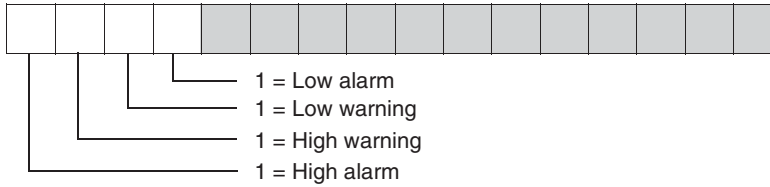
The nine middle-node registers are set using the reference data editor. DB is 5 V followed by 10 V increments of high and low warning. The actual high and low alarm is set at 20 V above and below nominal.

On a graph, the example looks like this:



Note: The example value shows a decimal 46, which is in the normal range. No alarm is set, i.e., register 400440 = 0.

You can now verify the instruction in a running PLC by entering values in register 400450 that fall into the defined ranges. The verification is done by observing the bit change in register 400440 where:



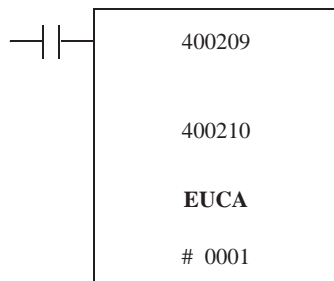
Example 2

If the input of 0 ... 4095 indicates the speed of a drive system of 0 ... 5000 rpm, you could set up a EUCA instruction as follows.

The binary value in 400210 results in an SPV of 4835 decimal, which exceeds the high absolute alarm level, sets the HA bit in 400209, and powers the EUCA alarm node.

Parameter	Speed
Maximum Speed	5 000 rpm
Minimum Speed	0 rpm
DB	100 rpm
HA Alarm	4 800 rpm
HW Alarm	4 450 rpm
LW Alarm	2 000 rpm
LA Alarm	1 200 rpm

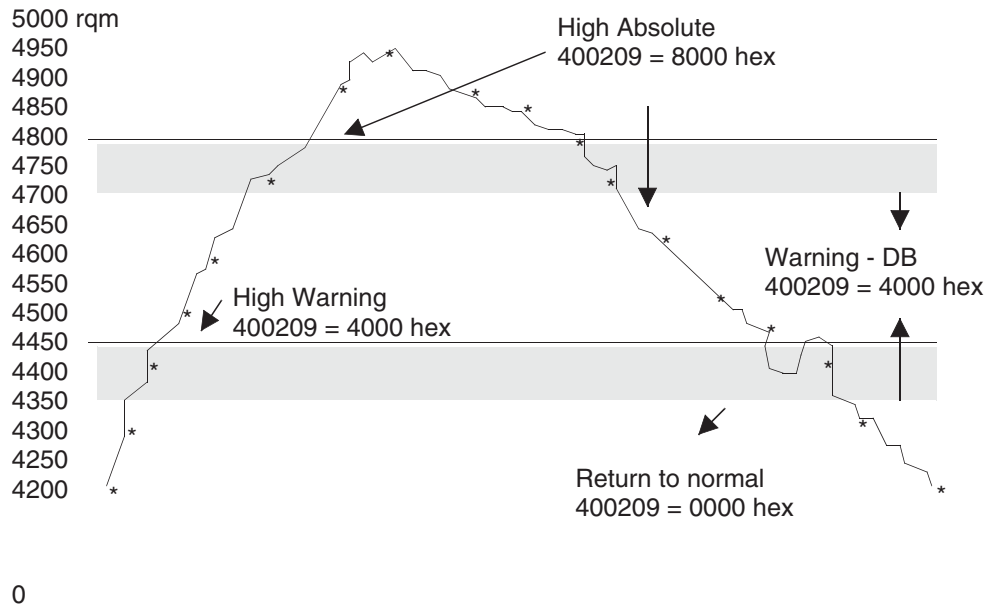
Instruction



Reference Data

Register	Meaning	Content
400209	STATUS	1000000000000000
400210	INPUT	3960 DEC
400211	SPV	4835 DEC
400212	MAX_SPEED	5000 DEC
400213	MIN_SPEED	0 DEC
400214	Dead_band	100 DEC
400215	HIGH_ALARM	4800 DEC
400216	HIGH_WARN	4450 DEC
400217	LOW_ALARM	2000 DEC
400218	LOW_WARN	1200 DEC

The N.O. contact is used to suppress alarm checks when the drive system is shutdown, or during initial start up allowing the system to get above the Low alarm RPM level.



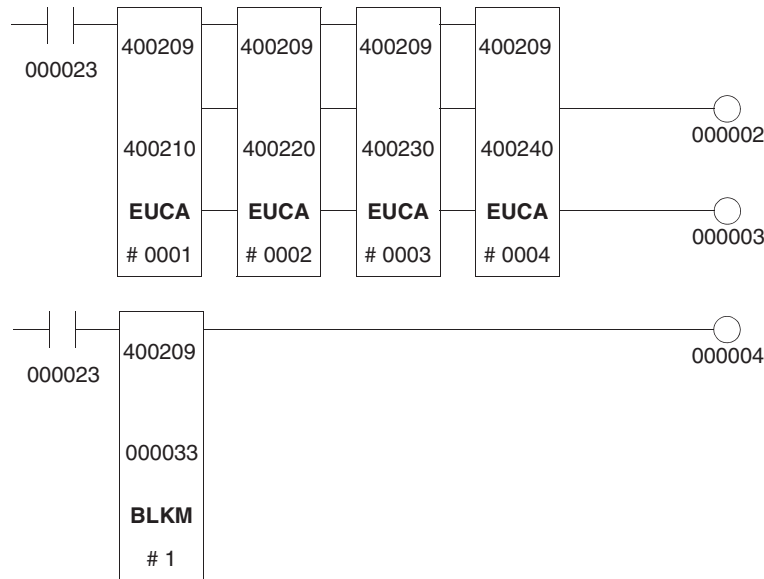
Varying the binary value in register 400210 would cause the bits in nibble 1 of register 400209 to correspond with the changes illustrated above. The DB becomes effective when the alarm or warning has been set, then the signal falls into the DB zone.

The alarm is maintained, thus taking what would be a switch chatter condition out of a marginal signal level. This point is exemplified in the chart above, where after setting the HA alarm and returning to the warning level at 4700 the signal crosses in and out of DB at the warning level (4450) but the warning bit in 400209 stays ON.

The same action would be seen if the signal were generated through the low settings.

Example 3

You can chain up to four EUCA conversions together to make one alarm status register. Each conversion writes to the nibble defined in the block bottom node. In the program example below, each EUCA block writes it's status (based on the table values for that block) into a four bit (nibble) of the status register 400209.



Reference Data

Register	Meaning	Content
400209	STATUS	0000001001001000

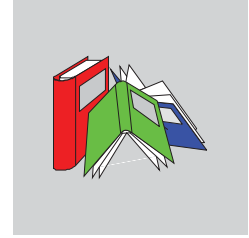
The status register can then be transferred using a BLKM instruction to a group of discrettes wired to illuminate lamps in an alarm enunciator panel.

As you observe the status content of register 400209 you see: no alarm in block 1, an LW alarm in block 2, an HW alarm in Block 3, and an HA alarm in block 4.

The alarm conditions for the four blocks can be represented with the following table settings:

	Conversion 1	Conversion 2	Conversion 3	Conversion 4
Input	400210 = 2048	400220 = 1220	400230 = 3022	400240 = 3920
Scaled #	400211 = 2501	400221 = 1124	400231 = 7379	400241 = 0770
HEU	400212 = 5000	400222 = 3300	400232 = 9999	400242 = 0800
LEU	400213 = 0000	400223 = 0200	400233 = 0000	400243 = 0100
DB	400214 = 0015	400224 = 0022	400234 = 0100	400244 = 0006
Hi Alarm	400215 = 40000	400225 = 2900	400235 = 8090	400245 = 0768
Hi Warn	400216 = 3500	400226 = 2300	400236 = 7100	400246 = 0680
Lo Warn	400217 = 2000	400227 = 1200	400237 = 3200	400247 = 0280
Lo Alarm	400218 = 1200	400228 = 0430	400238 = 0992	400248 = 0230

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 8#. 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 2#. 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 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)	<p>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).</p> <p>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.</p> <p>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.</p>

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: <ul style="list-style-type: none">● %0x12345 = %Q12345● %1x12345 = %I12345● %3x12345 = %IW12345● %4x12345 = %QW12345
IEC name conventions (identifier)	<p>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.</p> <p>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.</p> <p>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.</p>
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 I/O status of input bits is controlled via the process data, which reaches the CPU from an entry device.
	<div style="border: 1px solid black; padding: 5px;"><p>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.</p></div>
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)	<p>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.</p> <p>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.</p> <p>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.</p>
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	<p>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.</p> <p>EFB libraries can be subdivided into Groups.</p>
Literals	<p>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.</p> <p>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.</p>
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	<p>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.</p> <p>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.</p>

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	<p>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.</p> <p>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.</p>
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 status structure is given.

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 (**Not A Number**) oder INF (**INFinite**).

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 point 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	<p>If the data type for the literal is to be automatically determined, use the following construction: 'Data type name' #'Literal value'.</p> <p>Example INT#15 (Data type: Integer, value: 15), BYTE#00001111 (data type: Byte, value: 00001111) REAL#23.0 (Data type: Real, value: 23.0)</p> <p>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.</p>
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, discretets, 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	<p>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.</p> <p>The automatically generated step name always has the structure: S_n_m</p> <p>S = Step n = Section number (number running) m = Number of steps in the section (number running)</p>
Structured text (ST)	<p>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.</p>
Structured variables	<p>Variables, one of which is assigned a Derived data type defined with STRUCT (structure).</p> <p>A structure is a collection of data elements with generally differing data types (Elementary data types and/or derived data types).</p>
SY/MAX	<p>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.</p>
Symbol (Icon)	<p>Graphic display of various objects in Windows, e.g. drives, user programs and Document windows.</p>

T

Template data file (Concept EFB)	<p>The template data file is an ASCII data file with a layout information for the Concept FBD editor, and the parameters for code generation.</p>
TIME	<p>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 $2^{\text{exp}(32)}-1$. The unit for the data type TIME is 1 ms.</p>
Time span literals	<p>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.</p>

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 $2^{\text{exp}(32)}-1$.
- 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 $(2^{\text{exp}16})-1$.
- 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 display...** . 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



Numerics

3x or 4x register
entering in mathematical equation, 57

A

ABS, 64
AD16, 109
ADD, 113
Add 16 Bit, 109
Addition, 113
 AD16, 109
 ADD, 113
Advanced Calculations, 782
algebraic expression
 equation network, 54
algebraic notation
 equation network, 51
Analog Input, 787
Analog Output, 799
Analog Values, 71
AND, 117
ARCCOS, 64
ARCSIN, 64
ARCTAN, 64
argument
 equation network, 64
 limits, 65
arithmetic operator, 59
ASCII Functions
 READ, 937
 WRIT, 1091

assignment operator, 59
Average Weighted Inputs Calculate, 803

B

Base 10 Antilogarithm, 279
Base 10 Logarithm, 383
BCD, 123
benchmark performance
 equation network, 69
Binary to Binary Code, 123
Bit Control, 755
Bit pattern comparison
 CMPR, 167
Bit Rotate, 139
bitwise operator, 59
BLKM, 127
BLKT, 131
Block Move, 127
Block Move with Interrupts Disabled, 135
Block to Table, 131
BMDI, 135
boolean, 56
BROT, 139

C

Calculated preset formula, 809
Central Alarm Handler, 793
Changing the Sign of a Floating Point
Number, 301
Check Sum, 163

CHS, 157
CKSM, 163
Closed Loop Control, 71
CMPR, 167
coil
 equation network, 53
 error messages, 53
Coils, 91
Communications
 MSTR, 701
COMP, 179
Compare Register, 167
Complement a Matrix, 179
Comprehensive ISA Non Interacting PID,
831
conditional expression
 equation network, 51, 61
conditional operator, 59
Configure Hot Standby, 157
constant
 equation network, 51
constant data
 floating point, 57
 long (32-bit), 57
 LSB (least significant byte), 57
 mathematical equation, 57
Contacts, 91
Conversion
 BCD to binary, 123
 binary to BCD, 123
COS, 64
COSD, 64
Counters / Timers
 T.01 Timer, 1049
 T0.1 Timer, 1053
 T1.0 Timer, 1057
 T1MS Timer, 1061
 UCTR, 1077
Counters/Timers
 DCTR, 203
creating equation network, 52

D

data
 mathematical equation, 56
data conversions
 equation network, 66
Data Logging for PCMCIA Read/Write
Support, 223
DCTR, 203
Derivative Rate Calculation over a Specified
Time, 883
DIOH, 207
discrete reference
 mathematical equation, 56
Distributed I/O Health, 207
DIV, 217
Divide, 217
Divide 16 Bit, 243
DLOG, 223
Double Precision Addition, 265
Double Precision Division, 347
Double Precision Multiplication, 395
Double Precision Subtraction, 447
Down Counter, 203
DRUM, 237
DRUM Sequencer, 237
DV16, 243

E

EMTH, 259

EMTH Subfunction

EMTH-ADDDP, 265

EMTH-ADDFP, 271, 275

EMTH-ANLOG, 279

EMTH-ARCOS, 285

EMTH-ARSIN, 291

EMTH-ARTAN, 295

EMTH-CHSIN, 301

EMTH-CMPFP, 307

EMTH-CMPIF, 313

EMTH-CNVDR, 319

EMTH-CNVFI, 325

EMTH-CNVIF, 331

EMTH-CNVRD, 337

EMTH-COS, 343

EMTH-DIVDP, 347

EMTH-DIVFI, 353

EMTH-DIVFP, 357

EMTH-DIVIF, 361

EMTH-ERLOG, 365

EMTH-EXP, 371

EMTH-LNFP, 377

EMTH-LOG, 383

EMTH-LOGFP, 389

EMTH-MULDP, 395

EMTH-MULFP, 401

EMTH-MULIF, 405

EMTH-PI, 411

EMTH-POW, 417

EMTH-SINE, 423

EMTH-SQRFP, 429

EMTH-SQRT, 435

EMTH-SQRTP, 441

EMTH-SUBDP, 447

EMTH-SUBFI, 453

EMTH-SUBFP, 457

EMTH-SUBIF, 461

EMTH-TAN, 465

EMTH-ADDDP, 265

EMTH-ADDFP, 271

EMTH-ADDIF, 275

EMTH-ANLOG, 279

EMTH-ARCOS, 285

EMTH-ARSIN, 291

EMTH-ARTAN, 295

EMTH-CHSIN, 301

EMTH-CMPFP, 307

EMTH-CMPIF, 313

EMTH-CNVDR, 319

EMTH-CNVFI, 325

EMTH-CNVIF, 331

EMTH-CNVRD, 337

EMTH-COS, 343

EMTH-DIVDP, 347

EMTH-DIVFI, 353

EMTH-DIVFP, 357

EMTH-DIVIF, 361

EMTH-ERLOG, 365

EMTH-EXP, 371

EMTH-LNFP, 377

EMTH-LOG, 383

EMTH-LOGFP, 389

EMTHMULDP, 395

EMTH-MULFP, 401

EMTH-MULIF, 405

EMTH-PI, 411

EMTH-POW, 417

EMTH-SINE, 423

EMTH-SQRFP, 429

EMTH-SQRT, 435

EMTH-SQRTP, 441

EMTH-SUBDP, 447

EMTH-SUBFI, 453

EMTH-SUBFP, 457

EMTH-SUBIF, 461

EMTH-TAN, 465

enable contact

horizontal open, 53

horizontal short, 53

normally closed, 53

normally open, 53

Engineering Unit Conversion
and Alarms, 489

equation network

- ABS, 64
- algebraic expression, 54
- algebraic notation, 51
- ARCCOS, 64
- ARCSIN, 64
- ARCTAN, 64
- argument, 64
- argument limits, 65
- arithmetic operator, 59
- assignment operator, 59
- benchmark performance, 69
- bitwise operator, 59
- conditional expression, 51, 61
- conditional operator, 59
- constant, 51
- content, 54
- COS, 64
- COSD, 64
- creating, 52
- data conversions, 66
- enable contact, 53
- entering function, 64
- entering parentheses, 63
- EXP, 64
- exponentiation operator, 59
- FIX, 64
- FLOAT, 64
- group expressions in nested layers of

- parentheses, 51
- LN, 64
- LOG, 64
- logic editor, 51
- logical expression, 51
- math operator, 51
- mathematical, 55
- mathematical function, 64
- mathematical operation, 59
- nested parentheses, 63
- operator precedence, 62
- output coil, 53
- overview, 51
- parentheses, 59, 63
- relational operator, 59
- result, 54
- roundoff differences, 68
- SIN, 64
- SIND, 64
- single expression, 61
- size, 54
- SQRT, 64
- TAN, 64
- TAND, 64
- unary operator, 59
- variable, 51
- words consumed, 54

ESI, 469

EUCA, 489

Exclusive OR, 1145

EXP, 64

exponential notation

- mathematical equation, 58

exponentiation operator, 59

expression

- equation network, 61

Extended Math, 259

Extended Memory Read, 1133

Extended Memory Write, 1139

F

Fast I/O Instructions
 BMDI, 135
 ID, 617
 IE, 621
 IMIO, 625
 IMOD, 631
 ITMR, 639
FIN, 503
First In, 503
First Out, 507
First-order Lead/Lag Filter, 851
FIX, 64
FLOAT, 64
Floating Point - Integer Subtraction, 453
Floating Point Addition, 271
Floating Point Arc Cosine of an Angle (in Radians), 285
Floating Point Arc Tangent of an Angle (in Radians), 295
Floating Point Arcsine of an Angle (in Radians), 291
Floating Point Common Logarithm, 389
Floating Point Comparison, 307
Floating Point Conversion of Degrees to Radians, 319
Floating Point Conversion of Radians to Degrees, 337
Floating Point Cosine of an Angle (in Radians), 343
Floating Point Divided by Integer, 353
Floating Point Division, 357
Floating Point Error Report Log, 365
Floating Point Exponential Function, 371
Floating Point Multiplication, 401
Floating Point Natural Logarithm, 377
Floating Point Sine of an Angle (in Radians), 423
Floating Point Square Root, 429, 435
Floating Point Subtraction, 457
Floating Point Tangent of an Angle (in Radians), 465
Floating Point to Integer, 513
Floating Point to Integer Conversion, 325
floating point variable, 56

Formatted Equation Calculator, 821
Formatting Messages, 83
Four Station Ratio Controller, 887
FOUT, 507
FTOI, 513
function
 ABS, 64
 ARCCOS, 64
 ARCSIN, 64
 ARCTAN, 64
 argument, 64
 argument limits, 65
 COS, 64
 COSD, 64
 entering in equation network, 64
 EXP, 64
 FIX, 64
 FLOAT, 64
 LN, 64
 LOG, 64
 SIN, 64
 SIND, 64
 SQRT, 64
 TAN, 64
 TAND, 64

G

group expressions in nested layers of parentheses
 equation network, 51

H

History and Status Matrices, 583
HLTH, 583
horizontal open
 equation network, 53
horizontal short
 equation network, 53
Hot standby
 CHS, 157

I

IBKR, 603
IBKW, 607
ICMP, 611
ID, 617
IE, 621
IMIO, 625
Immediate I/O, 625
IMOD, 631
Indirect Block Read, 603
Indirect Block Write, 607
infix notation
 equation network, 52
Input Compare, 611
Input Selection, 897
Installation of DX Loadables, 101
Instruction
 Coils, Contacts and Interconnects, 91
Instruction Groups, 37
 ASCII Communication Instructions, 39
 Coils, Contacts and Interconnects, 50
 Counters and Timers Instructions, 40
 Fast I/O Instructions, 41
 Loadable DX, 42
 Math Instructions, 43
 Matrix Instructions, 45
 Miscellaneous, 46
 Move Instructions, 47
 Overview, 38
 Skips/Specials, 48
 Special Instructions, 49
Integer - Floating Point Subtraction, 461
Integer + Floating Point Addition, 275
Integer Divided by Floating Point, 361
Integer to Floating Point, 645
Integer x Floating Point Multiplication, 405
Integer-Floating Point Comparison, 313
Integer-to-Floating Point Conversion, 331
Integrate Input at Specified Interval, 827
Interconnects, 91
Interrupt Disable, 617
Interrupt Enable, 621
Interrupt Handling, 97
Interrupt Module Instruction, 631
Interrupt Timer, 639

ISA Non Interacting PI, 865
ITMR, 639
ITOF, 645

J

JSR, 649
Jump to Subroutine, 649

L

LAB, 653
Label for a Subroutine, 653
Limiter for the Pv, 837

-
- LL984
- AD16, 109
 - ADD, 113
 - AND, 117
 - BCD, 123
 - BLKM, 127
 - BLKT, 131
 - BMDI, 135
 - BROT, 139
 - CHS, 157
 - CKSM, 163
 - Closed Loop Control / Analog Values, 71
 - CMPR, 167
 - Coils, Contacts and Interconnects, 91
 - COMP, 179
 - DCTR, 203
 - DIOH, 207
 - DIV, 217
 - DLOG, 223
 - DRUM, 237
 - DV16, 243
 - EMTH, 259
 - EMTH-ADDDP, 265
 - EMTH-ADDFP, 271
 - EMTH-ADDIF, 275
 - EMTH-ANLOG, 279
 - EMTH-ARCOS, 285
 - EMTH-ARSIN, 291
 - EMTH-ARTAN, 295
 - EMTH-CHSIN, 301
 - EMTH-CMPFP, 307
 - EMTH-CMPIF, 313
 - EMTH-CNVDR, 319
 - EMTH-CNVFI, 325
 - EMTH-CNVIF, 331
 - EMTH-CNVRD, 337
 - EMTH-COS, 343
 - EMTH-DIVDP, 347
 - EMTH-DIVFI, 353
 - EMTH-DIVFP, 357
 - EMTH-DIVIF, 361
 - EMTH-ERLOG, 365
 - EMTH-EXP, 371
 - EMTH-LNFP, 377
 - EMTH-LOG, 383
 - EMTH-LOGFP, 389
 - EMTH-MULD, 395
 - EMTH-MULFP, 401
 - EMTH-MULIF, 405
 - EMTH-PI, 411
 - EMTH-POW, 417
 - EMTH-SINE, 423
 - EMTH-SQRFP, 429
 - EMTH-SQRT, 435
 - EMTH-SQRTP, 441
 - EMTH-SUBDP, 447
 - EMTH-SUBFI, 453
 - EMTH-SUBFP, 457
 - EMTH-SUBIF, 461
 - EMTH-TAN, 465
 - ESI, 469
 - EUCA, 489
 - FIN, 503
 - Formatting Messages for ASCII READ/

WRIT Operations, 83
FOUT, 507
FTOI, 513
HLTH, 583
IBKR, 603
IBKW, 607
ICMP, 611
ID, 617
IE, 621
IMIO, 625
IMOD, 631
Interrupt Handling, 97
ITMR, 639
ITOF, 645
JSR, 649
LAB, 653
LOAD, 657
MAP 3, 661
MBIT, 677
MBUS, 681
MRTM, 691
MSTR, 701
MU16, 747
MUL, 751
NBIT, 755
NCBT, 759
NOBT, 763
NOL, 767
OR, 775
PCFL, 781
PCFL-AIN, 787
PCFL-ALARM, 793
PCFL-AOUT, 799
PCFL-AVER, 803
PCFL-CALC, 809
PCFL-DELAY, 815
PCFL-EQN, 821
PCFL-INTEG, 827
PCFL-KPID, 831
PCFL-LIMIT, 837
PCFL-LIMV, 841
PCFL-LKUP, 845
PCFL-LLAG, 851
PCFL-MODE, 855
PCFL-ONOFF, 859
PCFL-PI, 865
PCFL-PID, 871
PCFL-RAMP, 877
PCFL-RATE, 883
PCFL-RATIO, 887
PCFL-RMPLN, 893
PCFL-SEL, 897
PCFL-TOTAL, 903
PEER, 909
PID2, 913
R --> T, 929
RBIT, 933
READ, 937
RET, 943
SAVE, 961
SBIT, 965
SCIF, 969
SENS, 975
SRCH, 987
STAT, 993
SU16, 1021
SUB, 1025
Subroutine Handling, 99
T.01 Timer, 1049
T-->R, 1037
T-->T, 1043
T0.1 Timer, 1053
T1.0 Timer, 1057
T1MS Timer, 1061
TBLK, 1067
TEST, 1073
UCTR, 1077
WRIT, 1091
XMRD, 1133
XMWT, 1139
XOR, 1145
LN, 64
LOAD, 657
Load Flash, 657
Load the Floating Point Value of "Pi", 411

- Loadable DX
 - CHS, 157
 - DRUM, 237
 - ESI, 469
 - EUCA, 489
 - HLTH, 583
 - ICMP, 611
 - Installation, 101
 - MAP 3, 661
 - MBUS, 681
 - MRTM, 691
 - NOL, 767
 - PEER, 909
- LOG, 64
- Logarithmic Ramp to Set Point, 893
- logic editor
 - equation network, 51, 52
- Logical And, 117
- logical expression
 - equation network, 51
- Logical OR, 775
- Look-up Table, 845
- LSB (least significant byte)
 - constant data, 57
- M**
- MAP 3, 661
- MAP Transaction, 661
- Master, 701
- Math
 - AD16, 109
 - ADD, 113
 - BCD, 123
 - DIV, 217
 - DV16, 243
 - FTOI, 513
 - ITOF, 645
 - MU16, 747
 - MUL, 751
 - SU16, 1021
 - SUB, 1025
 - TEST, 1073
- math coprocessor
 - roundoff differences, 68
- math operator
 - equation network, 51
- mathematical equation
 - constant data, 57
 - exponential notation, 58
 - values and data types, 55
- mathematical function
 - ABS, 64
 - ARCCOS, 64
 - ARCSIN, 64
 - ARCTAN, 64
 - argument, 64
 - argument limits, 65
 - COS, 64
 - COSD, 64
 - entering in equation network, 64
 - equation network, 64
 - EXP, 64
 - FIX, 64
 - FLOAT, 64
 - LN, 64
 - LOG, 64
 - SIN, 64
 - SIND, 64
 - SQRT, 64
 - TAN, 64
 - TAND, 64
- mathematical operation
 - arithmetic operator, 59
 - assignment operator, 59
 - bitwise operator, 59
 - conditional operator, 59
 - equation network, 59
 - exponentiation operator, 59
 - parentheses, 59
 - relational operator, 59
 - unary operator, 59

Matrix

AND, 117
BROT, 139
CMPR, 167
COMP, 179
MBIT, 677
NBIT, 755
NCBT, 759, 763
OR, 775
RBIT, 933
SBIT, 965
SENS, 975
XOR, 1145
MBIT, 677
MBUS, 681
MBUS Transaction, 681

Miscellaneous

CKSM, 163
DLOG, 223
EMTH, 259
EMTH-ADDDP, 265
EMTH-ADDFP, 271
EMTH-ADDIF, 275
EMTH-ANLOG, 279
EMTH-ARCOS, 285, 343
EMTH-ARSIN, 291
EMTH-ARTAN, 295
EMTH-CHSIN, 301
EMTH-CMPFP, 307
EMTH-CMPIF, 313
EMTH-CNVDR, 319
EMTH-CNVFI, 325
EMTH-CNVIF, 331
EMTH-CNVRD, 337
EMTH-DIVDP, 347
EMTH-DIVFI, 353
EMTH-DIVFP, 357
EMTH-DIVIF, 361
EMTH-ERLOG, 365
EMTH-EXP, 371
EMTH-LNFP, 377
EMTH-LOG, 383
EMTH-LOGFP, 389
EMTH-MULDP, 395
EMTH-MULFP, 401
EMTH-MULIF, 405
EMTH-PI, 411
EMTH-POW, 417
EMTH-SINE, 423
EMTH-SQRFP, 429
EMTH-SQRT, 435
EMTH-SQRTP, 441
EMTH-SUBDP, 447
EMTH-SUBFI, 453
EMTH-SUBFP, 457
EMTH-SUBIF, 461
EMTH-TAN, 465
LOAD, 657
MSTR, 701
SAVE, 961
SCIF, 969
XMRD, 1133

XMWT, 1139
mixed data types
 equation network, 66
Modbus Functions, 1099
Modbus Plus
 MSTR, 701
Modbus Plus Network Statistics
 MSTR, 732
Modify Bit, 677
Move
 BLKM, 127
 BLKT, 131
 FIN, 503
 FOUT, 507
 IBKR, 603
 IBKW, 607
 R --> T, 929
 SRCH, 987
 T-->R, 1037
 T-->T, 1043
 TBLK, 1067
MRTM, 691
MSTR, 701
 Clear Local Statistics, 716
 Clear Remote Statistics, 722
 CTE Error Codes for SY/MAX and TCP/
 IP Ethernet, 746
 Get Local Statistics, 714
 Get Remote Statistics, 720
 Modbus Plus and SY/MAX Ethernet
 Error Codes, 739
 Modbus Plus Network Statistics, 732
 Peer Cop Health, 724
 Read CTE (Config Extension Table), 728
 Read Global Data, 719
 Reset Option Module, 727
 SY/MAX-specific Error Codes, 741
 TCP/IP Ethernet Error Codes, 743
 TCP/IP Ethernet Statistics, 737
 Write CTE (Config Extension Table), 730
 Write Global Data, 718
MU16, 747
MUL, 751
Multiply, 751
Multiply 16 Bit, 747
Multi-Register Transfer Module, 691

N

NBIT, 755
NCBT, 759
nested layer
 parentheses, 51
nested parentheses
 equation network, 63
Network Option Module for Lonworks, 767
NOBT, 763
NOL, 767
Normally Closed Bit, 759
normally closed contact
 equation network, 53
Normally Open Bit, 763
normally open contact
 equation network, 53

O

ON/OFF Values for Deadband, 859
One Hundredth Second Timer, 1049
One Millisecond Timer, 1061
One Second Timer, 1057
One Tenth Second Timer, 1053
operator combinations
 equation network, 66
operator precedence
 equation network, 62
OR, 775
output coil
 equation network, 53

P

parentheses
 entering in equation network, 63
 equation network, 51
 nested, 63
 nested layer, 51
 using in equation network, 63
PCFL, 781
PCFL Subfunctions
 General, 73
PCFL-AIN, 787
PCFL-ALARM, 793

PCFL-AOUT, 799
PCFL-AVER, 803
PCFL-CALC, 809
PCFL-DELAY, 815
PCFL-EQN, 821
PCFL-INTEG, 827
PCFL-KPID, 831
PCFL-LIMIT, 837
PCFL-LIMV, 841
PCFL-LKUP, 845
PCFL-LLAG, 851
PCFL-MODE, 855
PCFL-ONOFF, 859
PCFL-PI, 865
PCFL-PID, 871
PCFL-RAMP, 877
PCFL-RATE, 883
PCFL-RATIO, 887
PCFL-RMPLN, 893
PCFL-SEL, 897
PCFL-Subfunction
 PCFL-AIN, 787
 PCFL-ALARM, 793
 PCFL-AOUT, 799
 PCFL-AVER, 803
 PCFL-CALC, 809
 PCFL-DELAY, 815
 PCFL-EQN, 821
 PCFL-INTEG, 827
 PCFL-KPID, 831
 PCFL-LIMIT, 837
 PCFL-LIMV, 841
 PCFL-LKUP, 845
 PCFL-LLAG, 851
 PCFL-MODE, 855
 PCFL-ONOFF, 859
 PCFL-PI, 865
 PCFL-PID, 871
 PCFL-RAMP, 877
 PCFL-RATE, 883
 PCFL-RATIO, 887
 PCFL-RMPLN, 893
 PCFL-SEL, 897
 PCFL-TOTAL, 903
PCFL-TOTAL, 903
PEER, 909

PEER Transaction, 909
PID Algorithms, 871
PID Example, 77
PID2, 913
PID2 Level Control Example, 80
PLCs
 roundoff differences, 68
 scan time, 69
precedence
 equation network, 62
Process Control Function Library, 781
Process Square Root, 441
Process Variable, 72
Proportional Integral Derivative, 913
Put Input in Auto or Manual Mode, 855

Q

Quantum PLCs
 roundoff differences, 68

R

R --> T, 929
Raising a Floating Point Number to an Integer Power, 417
Ramp to Set Point at a Constant Rate, 877
RBIT, 933
READ, 937
 MSTR, 712
Read, 937
READ/WRITE Operations, 83
Register to Table, 929
registers consumed
 mathematical equation, 56
Regulatory Control, 782
relational operator, 59
Reset Bit, 933
result
 equation network, 54
RET, 943
Return from a Subroutine, 943
roundoff differences
 equation network, 68

S

SAVE, 961
Save Flash, 961
SBIT, 965
SCIF, 969
Search, 987
SENS, 975
Sense, 975
Sequential Control Interfaces, 969
Set Bit, 965
Set Point Variable, 72
signed 16-bit variable, 56
signed long (32-bit) variable, 56
SIN, 64
SIND, 64
single expression
 equation network, 61
Skips / Specials
 RET, 943
Skips/Specials
 JSR, 649
 LAB, 653

Special

DIOH, 207
PCFL, 781
PCFL-, 799
PCFL-AIN, 787
PCFL-ALARM, 793
PCFL-AVER, 803
PCFL-CALC, 809
PCFL-DELAY, 815
PCFL-EQN, 821
PCFL-KPID, 831
PCFL-LIMIT, 837
PCFL-LIMV, 841
PCFL-LKUP, 845
PCFL-LLAG, 851
PCFL-MODE, 855
PCFL-ONOFF, 859
PCFL-PI, 865
PCFL-PID, 871
PCFL-RAMP, 877
PCFL-RATE, 883
PCFL-RATIO, 887
PCFL-RMPLN, 893
PCFL-SEL, 897
PCFL-TOTAL, 903
PCPCFL-INTEGFL, 827
PID2, 913
STAT, 993
SQRT, 64
SRCH, 987
STAT, 993
Status, 993
SU16, 1021
SUB, 1025
Subroutine Handling, 99
Subtract 16 Bit, 1021
Subtraction, 1025
Support of the ESI Module, 469

T

T.01 Timer, 1049
T-->R, 1037
T-->T, 1043
T0.1 Timer, 1053
T1.0 Timer, 1057
T1MS Timer, 1061
Table to Block, 1067
Table to Register, 1037
Table to Table, 1043
TAN, 64
TAND, 64
TBLK, 1067
TCP/IP Ethernet Statistics
 MSTR, 737
TEST, 1073
Test of 2 Values, 1073
Time Delay Queue, 815
Totalizer for Metering Flow, 903

U

UCTR, 1077
unary operator, 59
unsigned 16-bit variable, 56
unsigned long (32-bit) variable, 56
Up Counter, 1077

V

values and data types
 mathematical equation, 55
variable
 equation network, 51
variable data
 mathematical equation, 56
Velocity Limiter for Changes in the Pv, 841

W

word
 maximum in an equation network, 54
words consumed
 constant data, 57
 mathematical equation, 56

WRIT, 1091
Write, 1091
 MSTR, 710

X

XMRD, 1133
XMWT, 1139
XOR, 1145