

# Chapter 3 Common element

---

3.1. Expression .....	3-1
3.2. Date types .....	3-4
3.3. Variable .....	3-8
3.4. Keywords.....	3-16
3.5. Program type.....	3-17



### 3. Common element

Program configuration element of G series PLC(program block, function, and function block) can be prepared by different languages such as IL, LD, SFC and etc. But, these languages contain common structure elements.

#### 3.1. Expression

##### 3.1.1. Identifiers

- . The combination of English alphabet or all literal starting with underline character(\_), number, underline character can be the identifier.
- . Identifier is used for the variable name.
- . Identifier shall not include the space.
- . Identifier is 16 characters of English literal in case of normal variable and 8 characters of English literal in case of I/O variable and instance name.
- . All English literal are acknowledged as the capital letter.

Feature description	Examples
Upper case and numbers	IW210, IW215Z, QX75, IDENT
Upper case, numbers, embedded underlines	LIM_SW_2, LIMSW5, ABCD, AB_CD
Upper case, numbers, leading or embedded underlines	_MAIN, _12V7, _ABCD

##### 3.1.2. Data expression

Numeric Literal, Character String, Time Literal and etc. are used for the data in G SERIES PLC.

Feature description	Examples
Integer literals	-12, 0, 123_456, +986
Real literals	-12.0, 0.0, 0.456, 3.14159_26
Real literals with exponents	-1.34E-12, 1.0E+6, 1.234E6
Base 2 literals	2#1111_1111, 2#1110_0000
Base 8 literals	8#377(Decimal 255) 8#340(Decimal 224)
Base 16 literals	16#FF(Decimal 255) 16#E0(Decimal 224)
Boolean zero (false) and one (true)	0, 1, TRUE, FALSE

#### 3.1.2.1. Numeric literals

- . There are two classes of numeric literals : integer literals and real literals.
- . Single underline characters( ) inserted between the digits of a numeric literal shall not be significant.
- . Decimal is expressed by general decimal number and, if it has decimal point, it is regarded as real literals.
- . Symbol of + and - can be used for the exponent. 'E' identifying the exponent does not have any difference between capital letter and small letter.
- . Real literals with exponent shall be described as below.  
**Ex)** 12E-5 ( x ) 12.0E-5 ( . )
- . 2,8,10, 16 can be used for the integer and binary # is marked before the numeric literals. If binary # is not marked, it is regarded as decimal.
- . 0 - 9, A - F are used for hexadecimal and a - f can be used also.
- . Symbol (+,-) shall not be used for the binary expression.
- . Boolean Data can be expressed by integer 0 and 1.

#### 3.1.2.2. Character string literals

- . All characters in the single quote character(') are the character string literals.
- . The length is restricted within 16 characters for character string constant and 30 characters for initialization.

**Example**

'CONVEYER'

#### 3.1.2.3. Time literals

- . Time literals is classified by Duration data for measuring or controlling the elapsed time of a control event time and Time of day and date data for synchronizing the beginning or end of a control event to an absolute time reference.

##### 3.1.2.3.1. Duration

- . Duration data shall be delimited on the left, by the keyword T# or t#.
- . The data shall be described in order of days(d), hours(h), minutes(m), seconds(s) and milli-seconds(ms) and can be start from any unit and ms do not need to be used but intermediate unit can not be omitted.
- . Underline literal( ) is not used.
- . The overflow is allowed at maximum unit and the unit can be described down to the decimal point except ms. However, the maximum can not exceed T#49d17h2m47s295ms. (32 bits of ms unit)
- . The place of decimal point is restricted to three points at the second unit(s).
- . The decimal point can not be used at ms unit.
- . Capital 'small letter are available for the unit letter.

Feature description	Examples
Duration(No underline)	T#14ms, T#14.7s, T#14.7m, T#14.7h t#14.7d, t#25h15m, t#5d14h12m18s356ms

### 3.1.2.3.2. Time of day and date

The date and time are expressed by three types of date, hour and date/hour as below.

Feature description	Prefix keyword
Date prefix	D#
Time of day prefix	TOD#
Date and time prefix	DT#

The date start with Jan. 1, 1984.

The expression of hour and date/hour is restricted and ms unit is available for three places down to decimal point. ( 1ms unit )

The overflow is not allowed at all units for hour and date/hour expression.

Feature description	Examples
Date literals	D#1984-06-25 d#1984-06-25
Time of day literals	TOD#15:36:55.36 tod#15:36:55.369
Date and time literals	DT#1984-06-25-15:36:55.36 dt#1984-06-25-15:36:55.369

## 3.2. Data types

Data has the data type expressing unique property.

### 3.2.1. Elementary data types

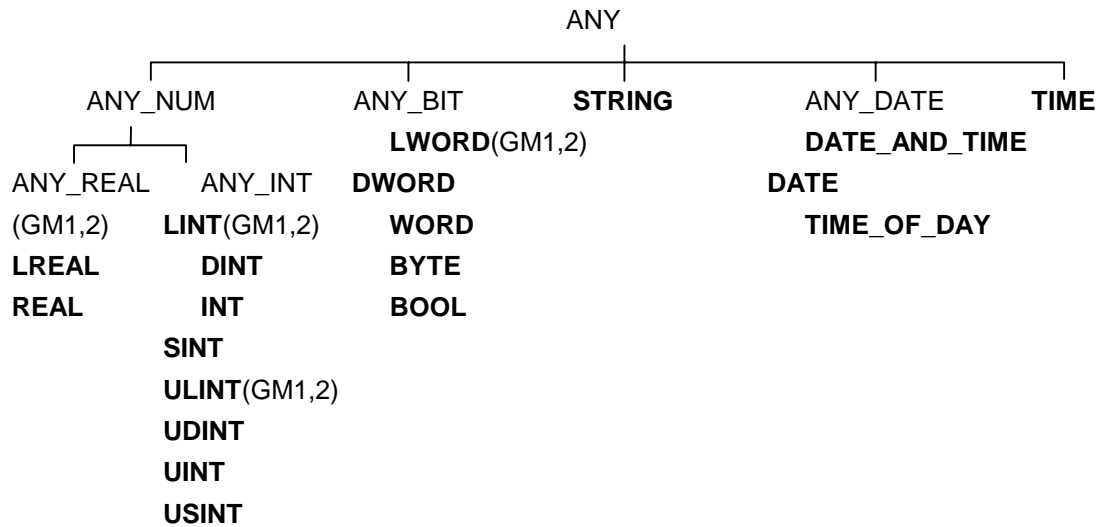
G SERIES PLC supports the basic data type as below.

No.	Keyword	Data type	Size (Bit)	Range
1	SINT	Short integer	8	-128 ~ 127
2	INT	Integer	16	-32768 ~ 32767
3	DINT	Double integer	32	-2147483648 ~ 2147483647
4	LINT	Long integer	64	$-2^{63} \sim 2^{63}-1$
5	USINT	Unsigned short integer	8	0 ~ 255
6	UINT	Unsigned integer	16	0 ~ 65535
7	UDINT	Unsigned double integer	32	0 ~ 4294967295
8	ULINT	Unsigned long integer	64	$0 \sim 2^{64}-1$
9	REAL	Real numbers	32	-3.402823E38 ~ -1.401298E-45 1.401298E-45 ~ 3.402823E38
10	LREAL	Long reals	64	-1.7976931E308 ~ -4.9406564E-324 4.9406564E-324 ~ 1.7976931E308
11	TIME	Duration	32	T#0S ~ T#49D17H2M47S295MS
12	DATE	Date	16	D#1984-01-01 ~ D#2163-6-6
13	TIME_OF_DAY	Time of day	32	TOD#00:00:00 ~ TOD#23:59:59.999
14	DATE_AND_TIME	Date and time of day	64	DT#1984-01-01-00:00:00 ~ DT#2163-12-31-23:59:59.999
15	STRING	Character string	30*8	-
16	BOOL	Boolean	1	0,1
17	BYTE	Bit string of length 8	8	16#0 ~ 16#FF
18	WORD	Bit string of length 16	16	16#0 ~ 16#FFFF
19	DWORD	Bit string of length 32	32	16#0 ~ 16#FFFFFFFF
20	LWORD	Bit string of length 64	64	16#0 ~ 16#FFFFFFFFFFFFFFFF

**Note** LINT, ULINT, REAL, LREAL and LWORD is **NOT** supported in the G6 and G4

### 3.2.2. Data type hierarchy

Below data type is used in G SERIES PLC.



- ANY\_REAL(LREAL, REAL), LINT, ULINT and LWORD are applied in GM1 and GM2.
- If ANY\_NUM is displayed in the data type, LREAL, REAL, LINT, DINT, INT, SINT, ULINT, UDINT, UINT and USINT are included as the hierarchy.
- For example, if the type is expressed as ANY\_BIT in GM3, one of DWORD, WORD, BYTE and BOOL can be used.

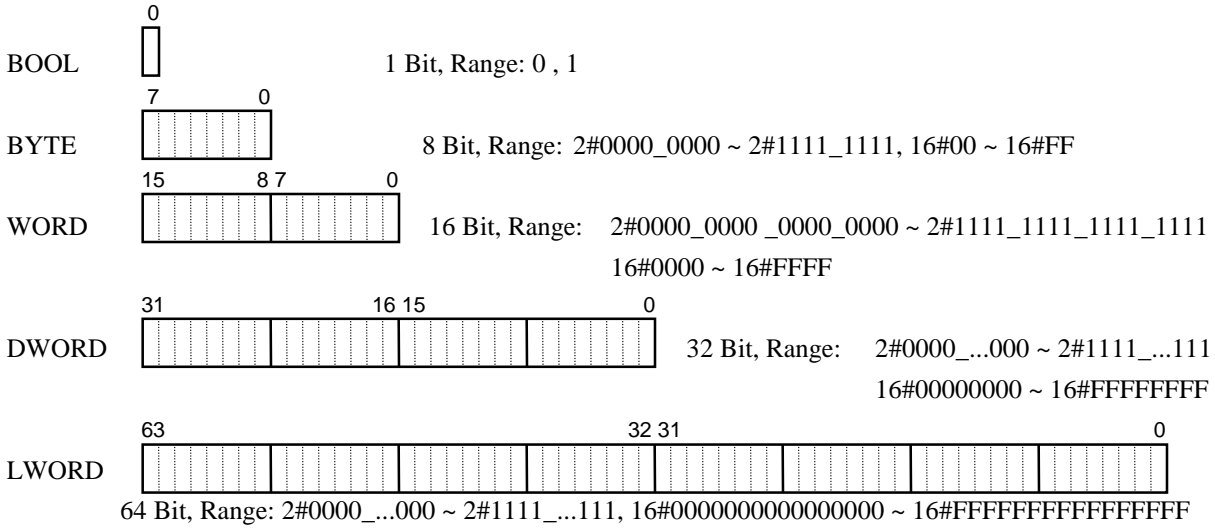
### 3.2.3. Initial value

If initial value of data is not assigned, the data will be assigned automatically as below.

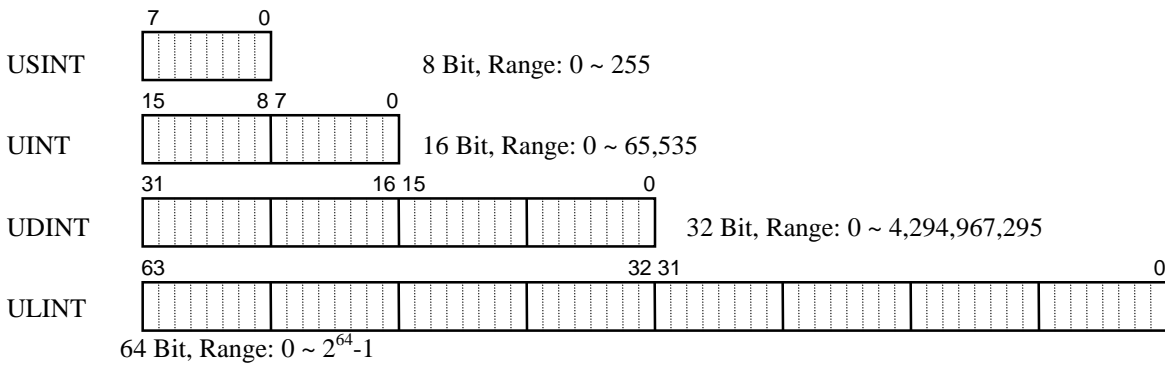
Data type(s)	Initial value
SINT, INT, DINT, LINT	0
USINT, UINT, UDINT, ULINT	0
BOOL, BYTE, WORD, DWORD, LWORD	0
REAL, LREAL	0.0
TIME	T#0s
DATE	D#1984-01-01
TIME_OF_DAY	TOD#00:00:00
DATE_AND_TIME	DT#1984-01-01-00:00:00
STRING	" (empty string)

### 3.2.4. Data type structure

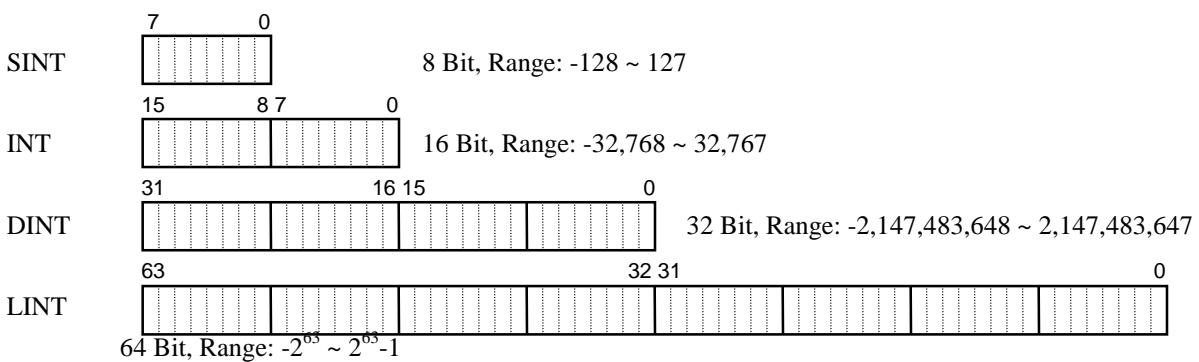
# Bit String



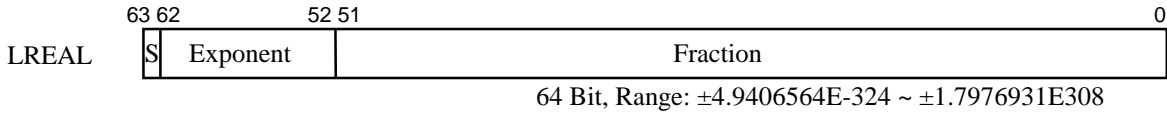
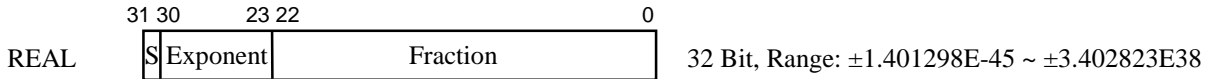
# Unsigned Integer



# Integer (Negative is expressed by 2' Complement)



# Real (Based on IEEE Standard 754-1984)

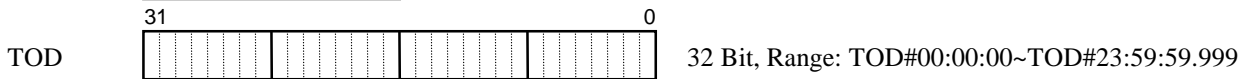
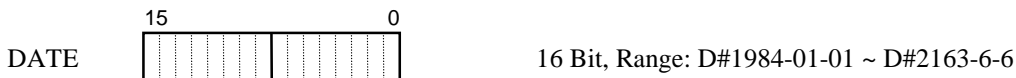
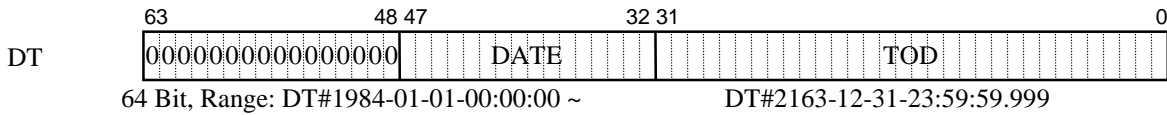


- S: symbol mark ( 0: Positive, 1: Negative)
- Exponent: 2' exponent (  $2^{e-127}$ :  $e=b_{30}b_{29}...b_{23}$ ,  $e=b_{62}b_{61}...b_{52}$  )
- Fraction: value down to the decimal point ( Fraction:  $f=b_{22}b_{21}...b_0$ ,  $e=b_{51}b_{52}...b_0$  )

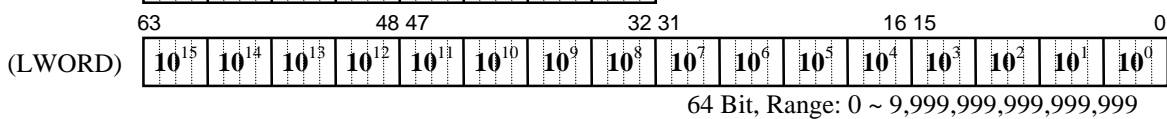
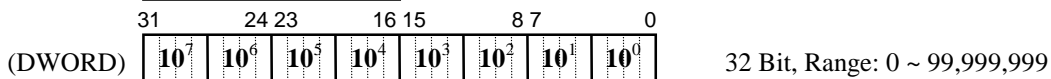
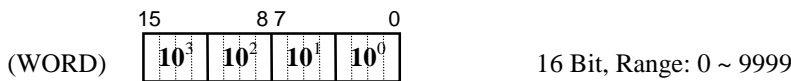
# Time



# Date



#BCD



### 3.3. Variable

The variable includes the data value in the program. The variable indicates PLC I/O, internal memory and etc.

#### 3.3.1. Representation

- The variable is expressed by two types; one is by assigning name with identifier and the other is by expressing PLC I/O or physical or logical place in the memory in the data element directly.
- The variable by identifier shall be unique in the scope(program configuration region declared as variable) to identify other variable.
- Direct variable can be expressed by the order of prefix starting with % letter and data. The prefix is described as below.

Location prefix

No.	Prefix	Meaning
1	I	Input Location
2	Q	Output Location
3	M	Memory Location

Size prefix

No.	Prefix	Meaning
1	X	Single bit size
2	None	Single bit size
3	B	Byte(8 Bits) size
4	W	Word(16 bits) size
5	D	Double Word(32 Bits) size
6	L	Long Word(64 Bits) size

Expression type

**%[Location prefix][Size prefix] n1.n2.n3**

No.	I, Q	M
n1	Base number(start from 0)	n1 data according to [Size prefix] (start from 0)
n2	Slot number(start from 0)	n2 bit on n1 data (start from 0) : can be omitted.
n3	n3 data according to [Size prefix] (start from 0)	Not used.

**Ex)**

%QX3.1.4 or %Q3.1.4	No. 4 output (1 Bit) of No. 1 slot of No. 3 base
%IW2.4.1	No.1 input by the word unit of No. 4 slot of No. 2 base (16 Bits)
%MD48	Memory of double word unit located at 48
%MW40.3	No.3 bit in memory on word unit located at 40

(There is no concept of base, slot and etc. internal memory.)

- . Small letter can not be used for the prefix.
- . If size prefix is not used, the variable is processed as 1 bit.
- . Direct variable can be used without declaration.

**3.3.2. Variable declaration**

- . Program configuration element(i.e. programm block, function, function block) has the declaration, which can declare the variable in configuration element.
- . The variable shall be declared in order to use it in the program configuration element.
- . Bellows shall be set for the variable declaration.

1) Variable type : set how to declare the variable.

Variable type	Description
VAR	General variable readable and writable
VAR_RETAIN	Retentive variable
VAR_CONSTANT	Variable readable only
VAR_EXTERNAL	Declaration to use VAR_GLOBAL variable

**Reference**

Resource global variable and configuration global variable are declared by VAR\_GLOBAL, VAR\_GLOBAL\_RETAIN, VAR\_GLOBAL\_CONSTANT and VAR\_EXTERNAL can not be declared.

- 2) Data type : Assign the data type of variable.
- 3) Memory allocation : Allocate the memory for variable.
  - Automatic ----The compiler assigns the variable location automatically  
(Automatic allocation variable).
  - User definition(AT) ----The user assigns the location by direct variable.  
(Direct variable)

#### **Reference**

The location of automatic allocation variable is not fixed. For example, if VAL1 variable is declared as BOOL data type, the variable is not fixed in a specific location in the data region. The compiler and linker define the location after programming. In case of compiling again after correcting the program, the location can be changed.

The benefit of automatic allocation of a variable is that the user does not need to check the location of the internal variable. A variable declared by a different name is not duplicated in the data memory.

If the variable location is set for direct variable, the memory is allocated implicitly by %I, %Q and %M. It is possible to duplicate the memory allocated so care must be taken

- . Initial Value allocation : Allocate the initial value of variable. If no, the value of clause 3.2.3. is allocated as initial value.

#### **Reference**

The initial value can not be allocated if VAR\_EXTERNAL is declared.

The initial value can not be allocated if %I %Q, and %M are allocated.

- . After PLC is switched off, the variable, which requires the data, can be declared by VAR\_RETAIN supplying retention function under below regulation.
  - 1) Retention variable is retained at the warm restart.
  - 2) During cold restart of system, it is initialized to initial value defined by user or basic initial value.
- . Variable not declared by VAR\_RETAIN is initialized to initial value defined by user or basic initial value during cold restart or warm restart.

#### **Reference**

The variable allocated to %I,%Q and %M can not be declared to VAR\_RETAIN, VAR\_CONSTANT.

- . The variable can be used by declaring basic data type to array. To declare array variable, the type and array size to be used for parameter shall be declared.

However, string data type in basic data type can not be set to parameter.
- . Scope of variable declaration, i.e., region to use variable, is restricted to the program configuration element that the variable is declared. Therefore, the variable declared in other program configuration element can not be used. The variable declared as global variable can be accessed in all location by declaring VAR\_EXTERNAL.

The configuration global variable can be used all program configuration element of all resources and the resource global variable can be used in all program configuration element.

**Example of variable declaration**

Name	Type	Data type	Initial value	Memory allocation
I_VAL	VAR	INT	1234	Automatic
BIPOLAR	VAR_RETAIN	REAL		Automatic
LIMIT_SW	VAR	BOOL		%IX1.0.2
GLO_SW	VAR_EXTERNAL	DWORD		Automatic
READ_BUF	VAR	ARRAY OF INT[10]		Automatic

**3.3.3. Reserve variable**

- . The reserved variable is declared in the system in advance. These variables are used for special purpose and the user can not declare them as variable name.
- . Use the reserved variable without declaration.
- . Refer to 'CPU user's manual' for details.

## 1) User flag

Reserve variable	Data type	Description
_ERR	BOOL	Operation error contact
_LER	BOOL	Operation error latch contact
_T20MS	BOOL	20 ms Clock contact
_T100MS	BOOL	100 ms Clock contact
_T200MS	BOOL	200 ms Clock contact
_T1S	BOOL	1 sec. Clock contact
_T2S	BOOL	2 sec. Clock contact
_T10S	BOOL	10 sec. Clock contact
_T20S	BOOL	20 sec. Clock contact
_T60S	BOOL	60 sec. Clock contact
_ON	BOOL	All time On contact
_OFF	BOOL	All time Off contact
_1ON	BOOL	1 scan On contact
_1OFF	BOOL	1 scan Off contact
_STOG	BOOL	Reversal at every scanning
_INIT_DONE	BOOL	Initial program completion
_RTC_DATE	DATE	Current date of RTC
_RTC_TOD	TOD	Current time of RTC
_RTC_WEEK	UINT	Current day of RTC

### 3. Common element

---

#### 2) System error flag

Reserve variable	Data type	Content
_CNF_ER	WORD	System error(heavy trouble)
_CPU_ER	BOOL	CPU configuration error
_IO_TYER	BOOL	Module type inconsistency error
_IO_DEER	BOOL	Module installation error
_FUSE_ER	BOOL	Fuse shortage error
_IO_RWER	BOOL	I/O module read/write error(trouble)
_SP_IFER	BOOL	Special/Communication module interface error(trouble)
_ANNUN_ER	BOOL	Heavy trouble detection error of external device
_WD_ER	BOOL	Scan Watch-Dog error
_CODE_ER	BOOL	Program code error
_STACK_ER	BOOL	Stack Overflow error
_P_BCK_ER	BOOL	Program error

#### 3) System error release flag

Reserve variable	Data type	Description
_CNF_ER_M	BYTE	System error(heavy trouble) release
_IO_DEER_M	BOOL	Module installation error release
_FUSE_ER_M	BOOL	Fuse shortage error release
_IO_RWER_M	BOOL	I/O module read/write error release
_SP_IFER_M	BOOL	Special/Communication module interface error release
_ANNUN_ER_M	BOOL	Heavy trouble detection error release of external device

#### 4) System alarm flag

Reserve variable	Data type	Description
_CNF_WAR	WORD	System alarm(Alarm message)
_RTC_ERR	BOOL	RTC data error
_D_BCK_ER	BOOL	Data back-up error
_H_BCK_ER	BOOL	Hot restart unable error
_AB_SD_ER	BOOL	Abnormal Shutdown
_TASK_ERR	BOOL	Task conflict (Normal cycle, external task)
_BAT_ERR	BOOL	Battery error
_ANNUN_WR	BOOL	Light trouble detection of external device

Reserve variable	Data type	Description
_HSPMT1_ER	BOOL	Over high-speed link parameter 1
_HSPMT2_ER	BOOL	Over high-speed link parameter 2
_HSPMT3_ER	BOOL	Over high-speed link parameter 3
_HSPMT4_ER	BOOL	Over high-speed link parameter 4

## 5) System error details flag

Reserve variable	Data type	Description
_IO_TYER_N	UINT	Module type inconsistency slot number
_IO_TYERR	ARRAY OF BYTE	Module type inconsistency location
_IO_DEER_N	UINT	Module installation slot number
_IO_DEERR	ARRAY OF BYTE	Module installation location
_FUSE_ER_N	UINT	Fuse shortage slot number
_FUSE_ERR	ARRAY OF BYTE	Fuse shortage slot location
_IO_RWER_N	UINT	I/O module read/write error slot number
_IO_RWERR	ARRAY OF BYTE	I/O module read/write error slot location
_IP_IFER_N	UINT	Special/Link module interface error slot number
_IP_IFERR	ARRAY OF BYTE	Special/Link module interface error slot location
_ANC_ERR	ARRAY OF UINT	Heavy trouble detection of external device
_ANC_WAR	ARRAY OF UINT	Light trouble detection of external device
_ANC_WB	ARRAY OF BIT	Light trouble detection bit map of external device
_TC_BMAP	ARRAY OF BYTE	Task conflict mark
_TC_CNT	UINT	Task conflict counter
_BAT_ER_TM	DT	Battery voltage drop-down time
_AC_F_CNT	UINT	Shutdown counter
_AC_F_TM	ARRAY OF DT	Instantaneous power failure history

### 3. Common element

#### 6) System operation status

Reserve variable	Data type	Description
_CPU_TYPE	UINT	System type
_VER_NUM	UINT	PLC O/S Ver. No.
_MEM_TYPE	UINT	Memory module type
_SYS_STATE	WORD	PLC mode and status
_GMWIN_CNF	BYTE	PADT connection status
_RST_TY	BYTE	Restart mode information
_INIT_RUN	BIT	Initializing
_SCAN_MAX	UINT	Max. scan time(ms)
_SCAN_MIN	UINT	Min. scan time(ms)
_SCAN_CUR	UINT	Current scan time(ms)
_STSK_NUM	UINT	Task number requiring execution time check
_STSK_MAX	UINT	Max. task execution time(ms)
_STSK_MIN	UINT	Min. task execution time(ms)
_STSK_CUR	UINT	Current task execution time(ms)
_RTC_TIME	ARRAY OF BYTE	Current time
_SYS_ERR	UINT	Error type

#### 7) Communication module information flag

[n corresponds to the slot number which the communication module is installed(n = 0 - 7)]

Reserve variable	Data type	Description
_CnVERNO	UINT	Communication module Ver. No.
_CnSTNOH _CnSTNOL	UINT	Communication module station number
_CnTXECNT	UINT	Communication transmit error
_CnRXECNT	UINT	Communication receive error
_CnSVCFCNT	UINT	Communication service process error
_CnSCANMX	UINT	Max. communication scan time(1ms unit)
_CnSCANAV	UINT	Average communication scan time(1ms unit)
_CnSCANMN	UINT	Min. communication scan time(1ms unit)
_CnLINF	UINT	Communication module system information
_CnCRDER	BOOL	Communication module system error(Error = 1)
_CnSVBSY	BOOL	Lack of common RAM resource(Lack=1)
_CnIFERR	BOOL	Interface error(Error = 1)
_CnINRING	BOOL	Communication in ring(IN_RING = 1)

8) Remote I/O control flag

[m correspond to the slot number which the communication module is installed(m = 0 - 7)].

Reserve variable	Data type	Description
_FSMm_reset	BOOL(Writable)	Remote I/O station reset control(reset=1)
_FSMm_io_reset	BOOL(Writable)	Output reset control of remote I/O station (reset=1)
_FSMm_st_no	USINT(Writable)	Station number of corresponding remote I/O station

9) HS(High-speed) link information detail flag

[m corresponds to the high-speed link parameter number(m = 1,2,3,4)]

Reserve variable	Data type	Description
_HSmRLINK	BIT	RUN_LINK information of HS link
_HSmLTRBL	BIT	Abnormal information of HS(Link Trouble)
_HSmSTATE	ARRAY OF BIT	General communication status information of k data block at HS link parameter
_HSmMOD	ARRAY OF BIT	Station mode information of k data block at HS link parameter (Run = 1, Others = 0)
_HSmTRX	ARRAY OF BIT	Communication status information of k data block at HS link parameter (Normal = 1, Abnormal = 0)
_HSmERR	ARRAY OF BIT	Station status information of k data block at HS link parameter (Normal = 0, Error = 1)

## 3.4. Keywords

The keywords are the words defined in advance for the system. Therefore, the identifier can not be used for the keywords.

Keywords
ACTION ... END_ACTION
ARRAY ... OF
AT
CASE ... OF ... ELSE ... END_CASE CONFIGURATION ... END_CONFIGURATION
Data type name
DATE#, D# DATE_AND_TIME#, DT#
EXIT
FOR ... TO ... BY ... DO ... END_FOR
FUNCTION ... END_FUNCTION
FUNCTION_BLOCK ... END_FUNCTION_BLOCK
Function block names
IF ... THEN ... ELSIF ... ELSE ... END_IF
OK
Operator (IL language) Operator (ST language)
PROGRAM
PROGRAM ... END_PROGRAM
REPEAT ... UNTIL ... END_REPEAT
RESOURCE ... END_RESOURCE
RETAIN
RETURN
STEP ... END_STEP
STRUCTURE ... END_STRUCTURE
T#
TASK ... WITH
TIME_OF_DAY#, TOD#
TRANSITION ... FROM... TO ... END_TRANSITION
TYPE ... END_TYPE
VAR ... END_VAR VAR_INPUT ... END_VAR VAR_OUTPUT ... END_VAR VAR_IN_OUT ... END_VAR VAR_EXTERNAL ... END_VAR
VAR_ACCESS ... END_VAR
VAR_GLOBAL ... END_VAR
WHILE ... DO ... END_WHILE
WITH

### 3.5. Program type

- . The program is divided by function, function block and program.
- . The program can not call its program itself.(Recursive calling prohibit)

#### 3.5.1. Functions

- . Function has one output.

##### **Example**

If A function adds input IN1 and IN2 and then adds the result with 100, i.e.,  $output\ 1 \leq IN1 + IN2 + 100$ , this function is correct. But, if  $output\ 2 \leq IN1 + IN2 * 100$  at another output, this can not be the function since the output is two, output 1 and output 2.

- . Functions shall contain no internal state information, i.e., invocation of a function with the same arguments (input parameters) shall always yield the same value (output).

##### **Example**

If B function is

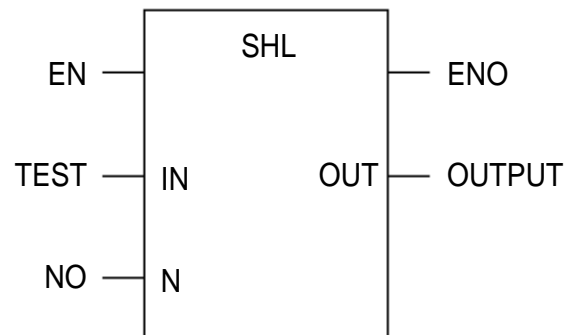
Output 1  $\leq$  IN1 + IN2 + Val

Val  $\leq$  Output 1 (Val is internal variable),

This can not be function since internal variable Val exists. Internal variable causes different output though the input is same. In above sample, output 1 can be changed due to Val variable though IN1 and IN2 are fixed. Comparing A function, if IN1 is 20 and IN2 is 30 in A function, output 1 is always 150. If the input is fixed, the output is also fixed.

- . Internal variable of function can not get initial value.
- . The variable can not be declared as VAR\_EXTERNAL.
- . Direct variable shall not be used in the function.
- . The function is called from the program configuration element.
- . In the configuration element of program calling function, the data transfer from function is executed through the input.

#### Example



SHL function is elementary function to output the result after left-shifting data if IN input to N times. The program configuration element calling SHL function assigns and calls TEST variable to IN input and N input to NO variable. The function result is stored to OUTPUT variable.

- . The function is used in the library.
- . The function block or program can not be called in the function.
- . The function is same to the function name and has the variable that data type is same to the function result. This variable is automatically generated during creating the function and outputs the result in this variable.

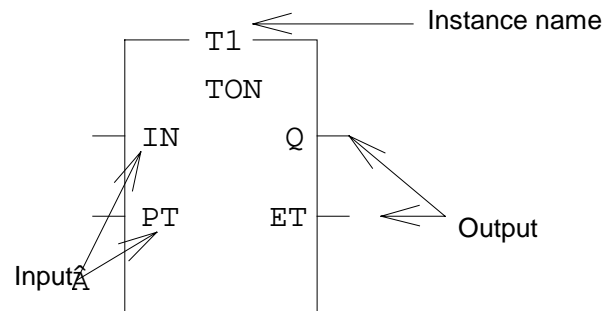
#### Example

If the function name is WEIGH and the result is WORD data type, internal function name becomes WEIGH and variable of WORD data type is automatically generated. The user inputs the result to WEIGH variable and outputs it.

ST WEIGH (Example at IL)

### 3.5.2. Function blocks

- . The function block output can be several.
- . The function block can have internal data. The function block shall declare the instance as if the variable is declared before using. The instance is the assembly of variable used in function block. The function block, as instance, shall have data memory since it stores the variable and output values. The program can be kind of function block and shall declare the instance. But, the program can not be used in the program or function block.
- . The output of function block puts the period(.) behind the instance name and then, write output name.

**Example**

General type of function block is the timer and counter. As on-delay timer function block is TON, declaring T1 and T2 as instance and calling them in the program operates on-delay time. The timer's output contact or elapsed time is used by putting the period(.) between instance name and output name. For the timer function block, as the output contact name is Q and the output contact of each instance is T1.Q, T2.Q and elapsed time name is ET, T1.ET and T2.ET describe elapsed time. Comparing the function output, the output is the return value calling function and the function block output is defined at each instance.

- . Direct variable can not be used in the function block. But the direct variable declared as global variable and allocated to user definition(AT) can be used as declaration of VAR\_EXTERNAL.
- . Function block is used by the library.
- . The program can not be called in the function block.

**3.5.3. Program blocks**

- . The program is declared as instance like function block.
- . Direct variable can be used in the program.
- . Input/Output variable is not used in the program.

The program call is defined in the resource.

MEMO

