HollySys

LM MICRO SERIES PLC

Instruction Sets Reference Manual

Revision 1.0 ©2008, MAN-PLC-LM-2008-08-30001
HOLLYSYS (ASIA PACIFIC) PTE LTD
200 Pandan Loop, #08-01,
Pantech 21, Singapore 128388
Phone +65 6777-09507 Fax +65 6777-2730

Copyright Notice

This English manual is an authorized translation of HollySys (Asia Pacific) Pte Ltd from the original Chinese materials.

Permitted Uses and Restrictions on Use

Disclaimer

The materials contain in this manual are provided for general information purpose only. Whilst every care has been taken to ensure the accuracy of the information provided, we can accept no responsibilities for loss or damage which may arise from reliance on the information contained in this manual.

The text and charts of this manual have been checked to be consistent with all the hardware equipment mentioned in the content, however, errors are still unavoidable and completed consistence may not be guaranteed. All materials and content is subjected to changes without prior notifications.

Specifications, charts, simple programs and application examples in this manual, which are only raised for illustration purpose, have been tested. However, because of the update of software version and other various unforeseeable changes, the companies shall not undertake any responsibilities of applications according to this manual.

Trademarks

HollySys and the logos, trade names, and product names are trademarks or registered trademarks of Beijing HollySys Co., Ltd and HollySys (Asia Pacific) Pte Ltd in PRC and other countries.

Microsoft, Windows and Windows NT are trademarks or registered trademarks of Microsoft in the United States and/or other countries branches.

Other trademarks or registered trademarks in this manual belong to their respective owners.

Contact Us

HollySys (Asia Pacific) Pte Ltd
Address: 200 Pandan Loop, #08-01 Pantech 21, Singapore 128388
Tel: (+65) 6777-0950
Fax: (+65) 6777-2730
Http: //www.hollysys.com.sg
Technical Support: **PLC_support@hollysys.com.sg**
Revision 1.0, Edition: September 2008

Preface

LM Micro series PLC, a new generation of smart PLC products by Hollysys Co., Ltd, consists of various kinds of CPU modules and expansion modules. Because of their advantages such as stable performance, reliable quality and affordable price, the LM Micro series PLC products have achieved wide applications in the vast range of all automation industry and earned good reputation from our clients.

The PowerPro is a Windows-based programming tool with a well-structure instruction system specially developed for the LM Micro series PLC by HollySys. It is the standard software package for LM Micro series PLC hardware configuration and software programming.

Contents

"HollySys LM Micro PLC Instruction Set Reference Manual" is a technical manual designed to give a detailed introduction of all HollySys LM Micro PLC instructions, including:

- Overview of LM MICRO PLC instructions
- Operands and data types of LM MICRO PLC
- Detailed introduction of LM MICRO PLC instructions
- Examples of LM MICRO PLC application

Application Scope

All instructions introduced here are applicable to the following programming software versions:

- PowerPro1.0.x
- PowerPro2.0.x
- PowerPro2.1.x

How to Use

- Experienced users who are familiar with PowerPro software may refer to a certain instruction directly.
- For first-time users, a thorough reading of CHAPTER 1 "OVERVIEW OF THE INSTRUCTION" is recommended.
- For users who are not very familiar with the operands and data types of PLC, a thorough reading of CHAPTER 2 and CHAPTER 3 are necessary.
- Appendix A provides LM Instructions Fast-Check List, IEC Standard Instructions List, and Fast-Checklist of Instruction Associations and Conflicts, and Hardware Module Status information.
- Appendix B provides some application examples for the reference of users.

More References:

- LM Micro PLC Overview
- LM Micro PLC Selection Guide
- LM Micro PLC Hardware Manual
- LM Micro PLC Software Manual

Table of Contents

Chapter 1

Overview of Instruction

HollySys LM Micro series PLCs provides users plenty of programming instructions for users to do programming. It can be used on the programming software, PowerPro, with simple and convenient operations.

This chapter gives a detailed introduction of the HollySys LM Micro PLC instructions (here forth known as 'LM instructions' in short). Please note that PowerPro v2.1 supports all the instructions mentioned in this manual while the previous versions of PowerPro can support most of the instructions.

1.1 INTRODUCTION

In PLC, a command or a combination of commands that enables the CPU to execute an operation or achieve a certain function is called an instruction. The collection of instructions is known as instruction system. The instruction system is the bridge between PLC hardware and software and the base of PLC programming.

PowerPro provides plenty of instructions that, according to their different functions, can be divided into 51 types such as conversion instructions, comparison instructions, type conversion instructions, logical operators, external interrupts, bi-phase counters, etc. For the user to easily understand and memorized, the 51 types are further classified into two categories: one is basic instructions including all IEC standard instructions, mathematical instructions, etc. while the other is expansion instructions including external interrupt instructions, pulse outputs, bi-phase counters and the instructions related to hardware ports, etc. All the expansion instructions are presented as function blocks, and the names of their libraries are all started with "HollySys", ***see appendix A.1 for details.***

LM instructions can be implemented either as functions or function blocks in the programming environment. Functions and function blocks are POUs (Program Organization Unit) of PowerPro that are pre-designed to achieve certain functions. A function block may output one or more results and each function blocks application is related to an identifier (i.e. the application name). A function needs no identifier, and one function can has only one result (i.e. the function return value), ***see HollySys LM Micro PLC Software Manual for more details of function block and function.*** In Appendix A.1 Quick-Check List of LM Instructions, the implement mode of each instruction is indicated by 'FUN', which stands for instruction function implement, and 'FB', which stands for function block implement.

1.2 DEFINITION AND CLASSIFICATION OF INSTRUCTION LIBRARY

In the process of PLC programming, some library instructions, such as the string, trigger, counter, and PID controller instructions are used frequently. The instructions with related functions are stored together to build a special instruction library.

Note:
In the use of instructions implemented by 'FUN', declaration is not required.
In the use of instructions implemented by 'FB', declaration of the application name is required.
In LM PLC instruction system, some instructions can only be called after their related libraries have been added, while some others are not packaged in the library and can be called directly. Please refer to appendix A.1 for

The instruction libraries are the collections of LM PLC instruction codes, and each library has its respective library filename (library name.lib). It is necessary to load the corresponding library file before calling the library instructions.

According to different functions of instructions, the libraries are classified into two categories:

- **Basic instruction libraries** – the collection of common PLC basic instructions.
- **Expansion instruction libraries** – the collections of expansion instructions. The names of the libraries in this category all start with "Hollysys".

According to different positions of the instruction executive codes in the libraries, the instruction libraries are classified into three categories:

- **First category:** PowerPro build-in open instruction libraries. Instruction executive codes are stored in library file so that users can open the file by PowerPro to modify the instruction executive codes, and the users can design their own customized libraries. When downloaded into the PLC, the library will occupy larger memory space in the user program.
- **Second category:** PowerPro build-in proprietary instruction libraries. Instruction executive codes are stored in library file named "library name.hex". Users cannot open the file in PowerPro to modify the instruction codes. When using PowerPro software, users need to ensure that the hex file is named consistently with the lib file, and both of them are stored in the same directory. When downloaded into the PLC, the library will occupy larger memory space in the user program.
- **Third category**: PowerPro external instruction libraries. Instruction executive codes are stored in the base structure of PLC system and cannot be modified by users. When downloaded to the PLC, the library will occupy smaller memory space in the user program.

The above classification of instruction libraries will enhance the users' understanding of LM instruction system improving their programming skills using PowerPro software. In the following introductions of the basic instruction libraries and expansion instruction libraries, every library has been classified into the above mentioned three categories.

Tip:
To use the LM instructions, the related library file (library name.lib) shall be added.
Once the PowerPro instruction library(s) are added, it will occupy the user program memory space even no instruction from that library are used. Hence, it is suggested to add the necessary libraries that are being used only.

1.2.1 Basic Instruction Library

Standard Instruction Library (Standard.lib)

Standard.lib is a PowerPro external instruction library that is added automatically whenever a project is created. It is not needed to manually add this library again. The instructions included in Standard.lib are shown in figure 1-2-1.

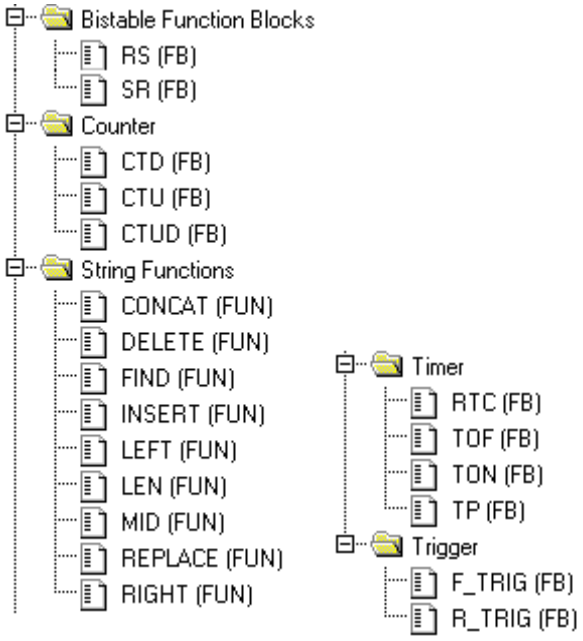


Figure 1-2-1 Standard Instruction Library

The definition of LM instructions in this library is shown in table 1-2-1.

Bistable Function Blocks	Counter
RS (Reset Dominant) SR (Set Dominant)	CTD (Counter Down) CTU (Counter Up) CTUD (Counter Up Down)
String Function	Timer
CONCAT (Concatenation of Two Strings) DELETE (Delete String) FIND (Search String) INSERT (Insert String) LEFT (Get String from Left) LEN (Get String Length) MID (Get the partial string from within a string) REPLACE (Replace String) RIGHT (Get string from right)	RTC (Real Time Clock) TOF (Timer Off Delay) TON (Timer On Delay) TP (Timer Pulse)
	Trigger
	F_TRIG (Falling Edge Detection) R_TRIG (Rising Edge Detection)

Table 1-2-1 Definition of Standard Instruction Library

Application Instruction Libraries (Util.lib and Util_no_Real.lib)

Util.lib and Util_no_Real.lib are PowerPro build-in open instruction libraries that shall be added manually by users before use. The instructions included in Util.lib are shown in figure 1-2-2.

Util.lib

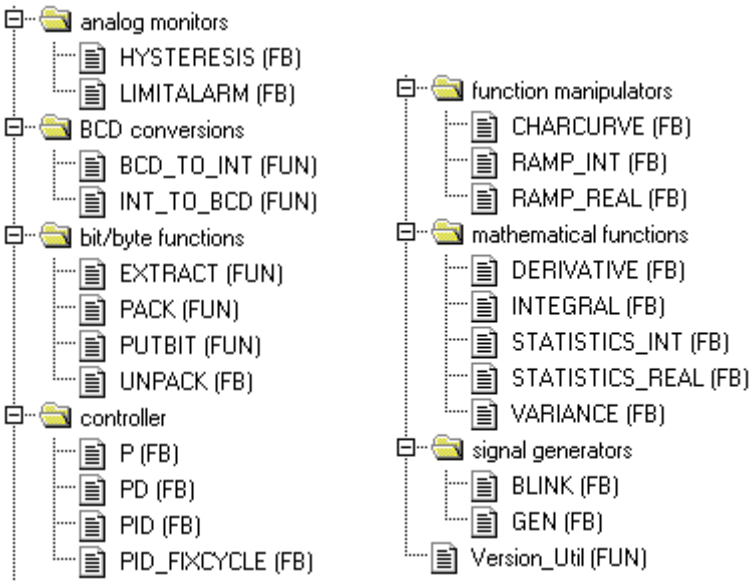


Figure 1-2-2 Application Instruction Libraries: Util.lib

The definition of LM instructions in this library is shown in table 1-2-2.

Analog processing Instructions	BCD Conversion Instruction
HYSTERESIS (Hysteresis) LIMITALARM (Limit Alarm)	BCD_TO_INT (BCD to INT) INT_TO_BCD (INT to BCD)
PID Controller Instructions	Mathematical Instructions
P (Proportional) PD (Proportional and Derivative) PID (Proportional Integral and Derivative) PID_FIXCYCLE (PID, Fixcycle)	DERIVATIVE (Derivative) INTEGRAL (Integral) STATISTICS_INT (INT Statistics) STATISTICS_REAL (REAL Statistics) VARIANCE (Variance)
Bit Handling Instructions	Function manipulator Instructions
EXTRACT (Bit Extracted) PACK (Pack Eight Bits into 1 Byte) PUTBIT (Set Bit Value) UNPACK (Unpacks Byte into 8 Bits)	CHARCURVE (Characteristic Curve) RAMP_INT (Limit the Slope of a Value to a Certain Value) RAMP_REAL (Limit the Slope of a Value to a Certain Value)
Signal Generator Instructions	Version Check Instructions
BLINK (Pulse Signal Generator) GEN (Periodic Signal Generator)	Version_Util (Version Check)

Table 1-2-2 Definition of Application Instruction Library, Util.lib

Util_no_Real.lib

LM instructions included in Util_no_Real.lib are shown in figure 1-2-3.

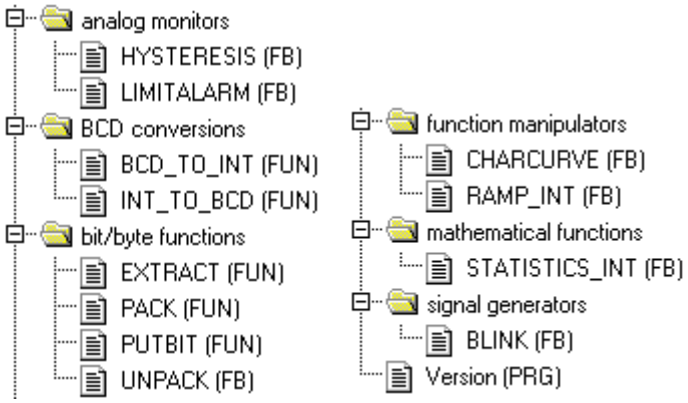


Figure 1-2-3 Application Instruction Libraries: Util_no_Real.lib

System Libraries (SysLibCallback.lib and SysLibC16x.lib)

SysLibCallback.lib and SysLibC16x.lib are PowerPro external libraries. LM instructions included are shown in figures 1-2-4 and 1-2-5. SysLibCallback.lib is added automatically when a new project is created while SysLibC16x.lib should be manually added by users when needed.

Tips:
The variables in Util.lib can be declared to the data type of REAL, while the variable in the Util_no_Real.lib can be declared to the data type other than REAL so that it will occupies lesser user program space.
Util.lib and Util_no_Real.lib cannot be utilized or added at the same time.

SysLibCallback.lib

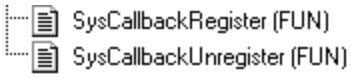


Figure 1-2-4

Functions include:

- Event call (SysCallbackRegister)
- Release event call (SysCallbackUnregistrer)

SysLibC16x.lib

Figure 1-2-5

Functions include:

- Get version information: SyslibGetVersion2300 can get version information of LM PLC.

Check Library (check.lib)

Check.lib is a PowerPro build-in open library. The LM instructions included in this library are shown in figure 1-2-6.

Tip:
SysLibCallback.lib is added automatically when a new project is created. Instructions in this library will be called automatically when system error occurs.

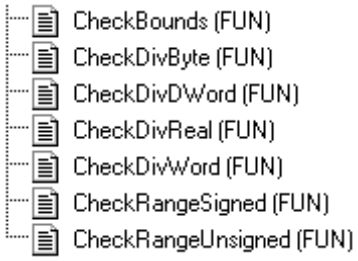


Figure 1-2-6

Functions include:

- Check whether the denominator is zero.
- Check boundary. .

IEC Action library (Iecsfc.lib)

Iecsfc.lib is a PowerPro build-in open library, and only one instruction is included in Iecsfc.lib, as shown in figure 1-2-7.

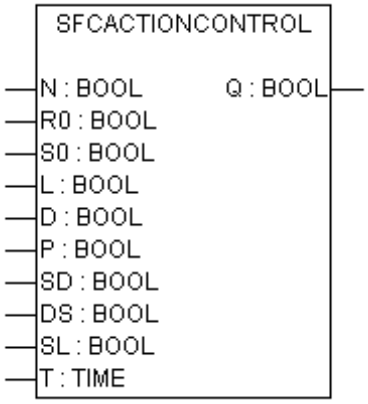


Figure 1-2-7

Functions include:

- SFCActionControl：IEC step associated action control of SFC language.

Tip:
Using IEC step associated action control, the Iecsfc.lib must be added first, otherwise error message will be prompt during compiling. System will call SFCActionControl automatically when there are associated actions. Refer to the Software Manual for more details.

1.2.2 Expansion Instruction Library

Introduction of expansion instruction libraries mainly contains the following three parts: supplementary library of Util.lib, Hollysys_PLC_***.lib and Hollysys_PLC_Ex_***.lib.

Supplementary Library of Util.lib

Hollysys_PLC_Util.lib and Hollysys_PLC_Modbus_CRC.lib are supplementary libraries of Util.lib that are classified as PowerPro build-in open libraries. Each includes only one LM instruction, as shown in figure 1-2-8.

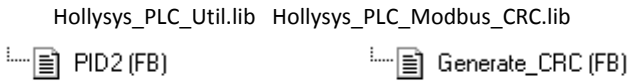


Figure 1-2-8

Hollysys_PLC_***.lib

Here *** stands for ANALOG, COMM, COMM2, DPSLAVE, Ethernet, HDRTC, HDRTCALM, HDRTCALM_N, PTOCtrl, Cnvt, Math and IO, totally 12 instruction libraries that are all PowerPro External libraries. LM instructions of each library are shown in table 1-2-3.

Hollysys_PLC_ANALOG.lib (analog module processing)	Hollysys_PLC_DPSLAVE.lib (DP module call)
Analog_IN (FB) Analog_OUT (FB)	DP_Slave (FB)
Hollysys_PLC_COMM.lib (RS232 communication)	Hollysys_PLC_COMM2.lib (RS485 communication)
COMM_RECEIVE (FB) COMM_SEND (FB) Reset_COMM_PRMT (FB) Set_COMM_PRMT (FB)	COMM2_RECEIVE (FB) COMM2_SEND (FB) Reset_COMM2_PRMT (FB) Set_COMM2_PRMT (FB)
Hollysys_PLC_HDRTC.lib (hardware real time clock)	Hollysys_PLC_HDRTCALM.lib (real time clock alarm)
Get_HD_RTC (FB) Set_HD_RTC (FB) Set_HD_RTC_X (FB)	Get_HDRTC_ALM (FB) Set_HDRTC_ALM (FB)
Hollysys_PLC_PTOCtrl.lib (mult-segment pulse output)	Hollysys_PLC_IO.lib (immediate output)
PTOCtrl_0 (FB) PTOCtrl_1 (FB)	OutPut_Bit (FB) Set_INT_OutPut (FB)
Hollysys_PLC_Cnvt.lib (EU conversion)	Hollysys_PLC_Math.lib (math)
E_H (FB) H_E (FB)	Rand (FB)

Hollysys_PLC_EtherNet.lib (ethernet module call)	Hollysys_PLC_HDRTCALM_N.lib (LM3104/5 real time clock alarm)
EtherNet_TCP (FB)	Get_HDRTC_ALM (FB) Set_HDRTC_ALM (FB)

Table 1-2-3

Hollysys_PLC_Ex_***.lib

Here *** includes CT, DCT, DCT32, TIMER, EXINT, PT, PTRun and Hollysys_PLC_Ex.lib, total 8 Nos. of instruction libraries; the libraries are PowerPro build-in proprietary libraries. LM instructions of each above mentioned library are shown in table 1-2-4.

Hollysys_PLC_Ex.lib (hardware instructions)	Hollysys_PLC_Ex_CT.lib (Mono-phase counters)
Get_Local_Address (FB) HD_WDT_Reset (FB) POT (FB) Set_Local_Address (FB)	HD_CTU_T7 (FB) HD_CTUD_T2 (FB) HD_CTUD_T3 (FB) HD_CTUD_T4 (FB)
Hollysys_PLC_Ex_DCT.lib (bi-phase counters)	**Hollysys_PLC_Ex_TIMER.lib (interrupt timers)**
HD_DCTUD_T2 (FB) HD_DCTUD_T3 (FB) HD_DCTUD_T4 (FB)	HD_Clear_T7 (FB) HD_STOP_T7 (FB) HD_TIMER_T7 (FB)
Hollysys_PLC_Ex_DCT32.lib (bi-phase 32 bit counters)	**Hollysys_PLC_Ex_ExINT.lib (external interrupt instructions)**
HD_DCTUD32_T3 (FB)	Fast_ExINT (FB) Fast_ExINT_E (FB)
Hollysys_PLC_Ex_PT.lib (pulse output instrctions)	**Hollysys_PLC_Ex_ PTRun.lib (pulse acceleration and deceleration output instructions)**
PTO_PWM0 (FB) PTO_PWM1 (FB)	PTO_PWM0_Run (FB) PTO_PWM1_Run (FB)

Table 1-2-4

1.3 INSTALL ADDITIONAL LIBRARY

To use a library, the file of this library shall be installed in the following directory: “PowerPro install directory\Library\”.

Start PowerPro, select “window/library manager” to open “library manager”, right-click, then select “additional library”, as shown in figure 1-3-1.

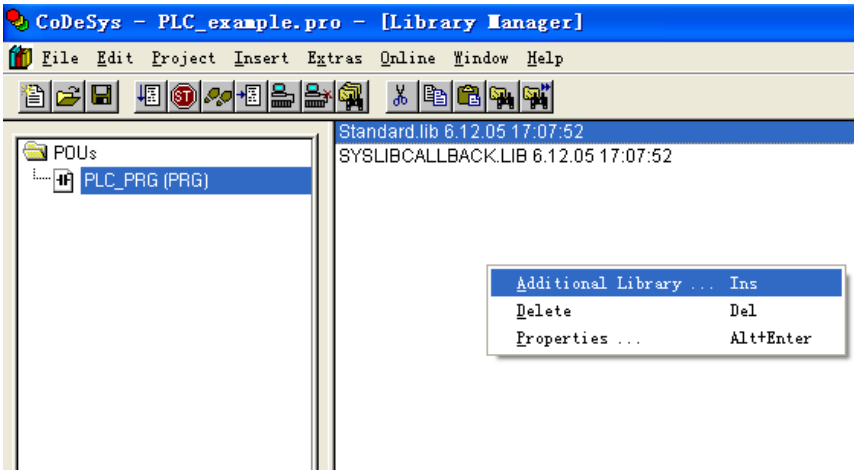


Figure 1-3-1

As shown in figure 1-3-2, select the needed file, and then click “open”. Open the corresponding *.lib file.

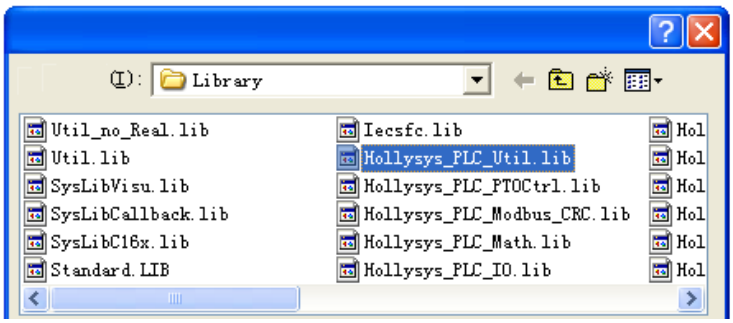


Figure 1-3-2

As shown in figure 1-3-3, the library selected is added to the list, the instructions contained in the library are shown in the window below.

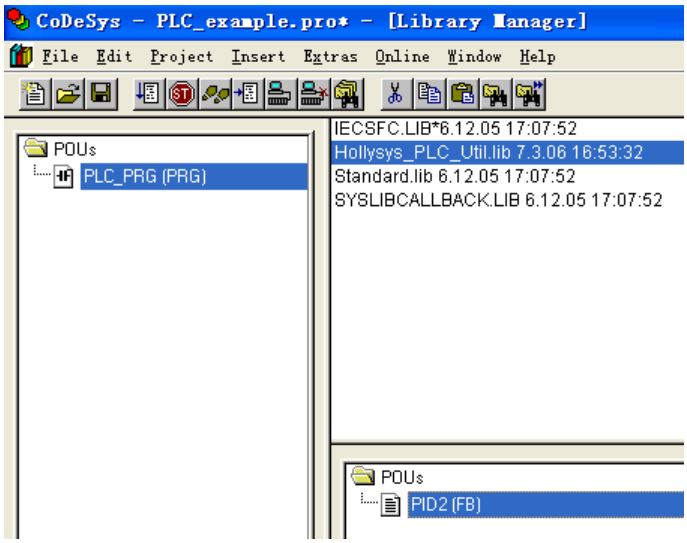


Figure 1-3-3

1.4 NOTES FOR USING INSTRUCTION SYSTEM

- Rising edge enable ---- Whenever the signal state changes from 0 to 1 and remains as 1, the related instructions will be executed.
- One-time-effect instruction after power-on/download ---- When the signal state changes from 0 to 1 and remains as 1, the instructions will be executed only once even if in the next cycle the rising edge change will be detected again. The instruction can be re-executed only after the re-download/re-power-on of the PLCs.
- %MB0～%MB99 is the reserved memory area for the system that users are allowed to read but not write, otherwise program will shows error.
- When using the Analog_IN, Analog_OUT and DP_Slave instructions, address setting must be consisted with the corresponding module node id that is configured in PLC, and they will execute when high signal state on EN.
- Free port communication parameters are set by Set_COMM_PRMT; to restore it to download/debug mode in the original program, set "RUN/STOP" switch to STOP and then login to programming system.
- When using Modbus slave setting instruction SET_LOCAL_ADDRESS, if the communication parameters are not 38400bps, 8 bits and no check, Reset_COMM_PRMT instruction will be called to set communication parameters, then SET_LOCAL_ADDRESS instruction will be called to set Modbus slave address. Set “RUN/STOP” switch to STOP before downloading; set "RUN/STOP" switch to RUN after downloading.
- Only after the setting of RS232 free port parameters by the Set_COMM_PRMT instruction, the use of COMM_SEND/COMM_RECEIVE instructions are allowed to send/receive data.
- Only after the setting of RS485 free port parameters by the Set_COMM2_PRMT instruction, the use of the COMM2_SEND/COMM2_RECEIVE instructions are allowed to send/receive data.
- Addresses of Word variables (%MW) must be defined in even number, such as %MW200, %MW202, %MW204.
- Addresses of Double-Word variables (%MD) must be defined in even number, such as %MD300, %MD304, %MD308.
- When using mathematical instructions, if the range defined for output data is less than the actual output result, then significant bit will be lost.
- Due to the hardware conflict, some instructions cannot be used at the same time. See Fast-Check List OF Associated instruction conflict for details.
- If the Instant Name of an instruction is declared into RETAIN area, all inputs and outputs that belong to this instruction will be stored in RETAIN memory. Therefore, it is suggested not to declare too many variables in RETAIN area as there is not enough memory.
- In Ladder Diagram (LD) programming environment, “Insert Box with EN" and “Insert "Function Block" (FB) are two different instructions modes. If selecting “Insert Box with EN” option, then the corresponding instruction codes will not be scanned when the signal state is 0; if selecting "Insert Function Block", the corresponding instructions will be scanned regardless of the signal status.

Chapter 2

Operands

In PLCs, instruction system is the bridge between the PLC hardware and software, and the base of PLC programming.

Similar to computer operation instructions, the basic format of PLC instruction is composed of operation codes and operands. Operation codes indicate the operation type and the operation instruction that are executed by CPU. Operands indicate the objects and purposes that are operated by CPU. Constants, variables, addresses and return values of functions all belong to operands.

2.1 CONSTANTS

2.1.1 Boolean Constants

Boolean constants: TRUE (1) and FALSE (0).

2.1.2 Clock Constants

Clock constant is used to operate the clock. Consisted of "T#" (or "t#") and "clock value", the clock constants contain units such as day (d), hour (h), minute (m), second (s) and millisecond (ms). The correct order of these units is d, h, m, and s, ms，for example “T #12h38m16s stands for 12 hours 38 minutes 16 seconds".

The correct clock constants:

```
T#18ms                  (*18ms*)
T#100s12ms              (*100s 12ms, limit exceeding is allowed for larger unit*)
T#12h34m15s             (*12h 34m 15s*)
```

The incorrect clock constants:

```
T#5m68s                 (*limit exceeding is not allowed for smaller unit*)
15ms                    (*need prefix T#*)
T#4ms13d                (*incorrect sequence*)
```

2.1.3 Date Constants

A Date constant consists of "D#" ("d#", "DATE#" or "date#") and "date value". For example:

```
DATE#2007-1-06          (*date constant 2007-1-6*)
D#1980-09-22            (*date constant 1980-9-22*)
```

2.1.4 Time Constants

A Time constant consists of "TOD#" ("tod#", "TIME_OF_DAY#" or "time_of_day#") and “time value”.

The format of time value is hour: minute: second (second can be REAL). For example:

```
TOD#00:00:00                (*time constant 0h 0m 0s*)
TIME_OF_DAY#15:36:30.123    (*time constant 15h 36m 30.123s*)
```

2.1.5 Date-Time Constants

A Date-Time constant is a combination of date constant and time constant that consists of "DT#" ("dt#", "DATE_AND_TIME#" or "date_and_time#") and "date-time value". For example:

```
DT#1980-09-22-15:45:18          (*date-time constant 1980y 9m 22d 15h 45m 18s*)
date_and_time#2001-03-09-00:00:00   (*date-time constant 2001y 3m 9d 0h 0m 0s*)
```

2.1.6 Numeric Constants

The value of numeric constants can be binary, decimal, octal and hexadecimal. If the integer value is not decimal, the “base” and "#" can be added in front of the integer. 10 to 15 in decimal correspond to A to F in hexadecimal. "_" can be used in Digital constant.

The data type of numeric constant can be BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, and UDINT.

By default, "larger" data type is not allowed to use as “smaller” data type. For example, DWORD data cannot be directly used as INT data, the solution is to execute a data type conversion before use. For example:

```
14              (*decimal 14*)
2#1001_0011     (*binary 1001_0011*)
8#67            (*octal 67*)
16#AE           (*hexadecimal AE*)
```

2.1.7 Real Constants

Real constants are expressed in decimal fractions and exponents, following the standard scientific format. The data type of real constant is REAL. For example：

```
7.4             (*real 7.4*)
1.64e+009       (*real 1.64e+009*)
```

2.1.8 String Constants

String constants are contained between two single quotation marks that allow spaces and special characters. For example:

```
'Abby and Craig'    (*string Abby and Craig*)
':-)'               (*string :-)*)
```

When the compound characters starting with $ are included in a string constant, the meanings are as follows:

String	Meaning
$$	Character $
$’	Single quotation marks

$L or $l	Input line
$N or $n	New line
$P or $p	Input page
$R or $r	Return
$T or $t	Tab key

2.2 VARIABLE

Variables may be declared in the variable list of POU or in the global variable list. Please note the follows in application:

- Variable names shall not include spaces or special characters; they shall not be declared more than once or use the same names with key words.
- Variable names are not case-insensitive. (e.g.： VAR1, Var1 and var1 are the same variables.)
- "_" are allowed in variable names. (e.g.: "A_BCD" and "AB_CD" are different variables.)
- Variable name does not support two continuous underline ("_ _"). (e.g.: "A_ _B" is a invalid variable name.)

2.2.1 System Identifiers

System identifiers are declared as implicit variables, and they're different in each system. Open auxiliary dialog box using command "insert/declaration keywords", and select the type of system variables to find the available system identifiers in the system.

2.2.2 Syntax of Variable access

- Access the elements of two-dimension array: <segment name>[Index1, Index2]
- Access structure variable: <structure name>.<variable name>
- Access function block and program variable: <instant name>.<variable name>

2.2.3 Access the Variable Bit

For INT variables, each bit of the variables is allowed to be accessed.

- The index number should place after the variable, and are separated from variables by using ".". The index number begins with 0.
- The data types which can be access by bit are: SINT, INT, DINT, USINT, UINT, UDINT, BYTE, WORD and DWORD.
- For those variables defined as VAR_IN_OUT, index accessing is invalid.

For example：

```
a : INT;              (*Type of variable a is Integer*)
b : BOOL;             (*Type of variable b is Boolean*)
...
a.2 := b;             (*Assign Boolean value b to the 2nd index of variable a*)
...
```

Errors:

If the index value is exceed the index limit of particular variable (for example a.16 := b), then errors message:

- Index '<n>' outside the valid range for variable '<var>'!

- Because the index range of INT variable is fixed as 0—15; a.16 exceeded upper limit.

Certain types of variable is not allow to be accessed by bit (for example a : REAL). It will give errors message:

- Invalid data type '<type>' for direct indexing
- (Index accessing is invalid for REAL data type.)

2.3 ADDRESS

2.3.1 Address Format

The memory address format is '%' + 'memory area' + 'data format' + 'address'.

Memory Area		Data Format	
I	Input	X	bit
Q	Output	B	byte (8 bits)
M	Memory location	W	word (16 bits)
		D	double word (32 bits)

For Example:

Address Format	Address
%QX7.5	the 5th BIT of output address 7
%IW4	1 WORD start from input address 4
%QB7	1 BYTE start from output address 7
%MD48	1 DWORD start from memory address 48

2.3.2 Memory Location

In PowerPro, the permutation of memory addresses is in the order of bytes, start from 0 and memory size is determined by PLC types. For example, the address definition of M is shown in table 2-1-1.

Address	Byte	Word	Double Word
0	%MB0	%MW0	%MD0
1	%MB1		
2	%MB2	%MW2	
3	%MB3		
4	%MB4	%MW4	%MD4
5	%MB5		
6	%MB6	%MW6	
7	%MB7		
……	……	……	……
4n	%MB4n	%MW4n	%MD4n
4n+1	%MB4n+1		
4n+2	%MB4n+2	%MW4n+2	
4n+3	%MB4n+3		

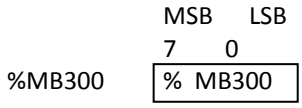
Table 2-1-1

Note:
WORD variables (%MW) must be defined into even addresses, such as %MW0, %MW2, %MW4, %MW6……%MW2n. Each WORD variable contains 2 bytes.
Double WORD variables (%MD) must be defined into even addresses, such as %MD0, %MD4……%MD4n. Each double WORD variable contains 4 bytes or 2 words.

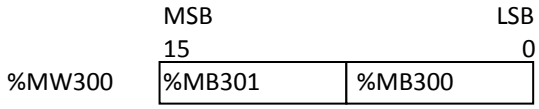
2.3.3 Data Storage Format

M memory area is taken as an example here to illustrate the data storage format in PowerPro. As shown below, MSB stands for Most Significant Bit, and LSB stands for Least Significant Bit:

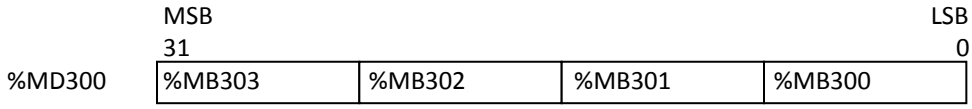
Byte



Word



Double Word



2.4 FUNCTION RETURN VALUE

In ST language, return values of function can be directly used as operands. For example:

```
Result:= Fct(7) + 3;  (*the function return value Fct plus 3, then assign it to Result *).
```

Chapter

3

Data Types

Standard data types and user-defined data types can be used in programming. The data types are indicated by identifiers, which specify the sizes of memory space the data use and which kind of data they are.

Standard data types include BOOL, INT, REAL, STRING and TIME, and they can be mutually converted using data type conversion instructions provided by PowerPro.

The user-defined data types include ARRAY, POINTER, ENUMERATION and STRUCT.

3.1 STANDARD DATA TYPES

3.1.1 Boolean Data Type

The identifier of Boolean variables is BOOL, the values are “TRUE”(1) and “FALSE”(0).

3.1.2 Integer Data Types

The identifiers of integer data types include BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, and UDINT. The range of each data type may be different. The ranges are as shown in table 3-1-1.

Identifier	Type	Minum Value	Maximum Value	Storage Space
BYTE	byte	0	255	8 Bit
WORD	word	0	65535	16 Bit
DWORD	double word	0	4294967295	32 Bit
SINT	short integer	-128	127	8 Bit
USINT	unsigned short integer	0	255	8 Bit
INT	integer	-32768	32767	16 Bit
UINT	unsigned integer	0	65535	16 Bit
DINT	double integer	-2147483648	2147483647	32 Bit
UDINT	unsigned double integer	0	4294967295	32 Bit

Table 3-1-1

3.1.3 Real Data Type

The real data type is also known as the floating-point data type, expressing the rational number. The identifier of the real data type is REAL. A REAL variable occupies 32 bits, or 4 bytes, of memory space.

Note:
When data of a larger data type is converted into a smaller data type, the higher bits will be lost.

3.1.4 String Data Type

A string can contain any number of characters, and the identifier is STRING. The size give at declaration of a string variable determines the size of memory space needed to store the variable, which is the numbers of characters in the string. If the size is not given, the default size 80 will be used.

For Example:

```
str:STRING(35):='This is a String';  (*declare a string with 35 characters *)
```

3.1.5 Time Data Type

The time data type is used for time processing, and the identifier is TIME (T for short), TIME_OF_DAY (TOD for short), DATE (D for short) and DATE_AND_TIME (DT for short).

- TIME is a time value in milliseconds, and the initial value is 0.
- TOD represents the amount of time in milliseconds that has elapsed from the beginning of the day, and the initial value is 00:00.
- DATE represents the current date, measured in seconds, and the initial value is January 1, 1970.
- DT represents a date and a time in seconds, and the initial value is 00:00 on January 1, 1970.

3.2 USER-DEFINED DATA TYPES

3.2.1 Arrays

One-dimensional, two-dimensional and three-dimensional arrays are basic data types. Arrays can be declared in variable lists or global variable lists of POUs. The identifier of arrays is ARRAY.

Syntax of Array Declaration

```
<array name>:ARRAY [<L1>..<U1>,<L2>..<U2>,<L3>..<U3>] OF <basic data type>;
```

L1, L2 and L3 are the minimum values of the fields, and U1, U2 and U3 are the maximum values. The ranges of the fields in an array must be integers.

For example:

```
Card_game: ARRAY [1..13, 1..4] OF INT;  (*Card_game is defined as a two-dimensional integer array*)
```

Array Initialization

The elements of an array can be either initialized or not initialized when the array is declared.

Example 1: fully initialized arrays

```
Arr1:ARRAY [1..5] OF BYTE:= 1,2,3,4,5;
Arr2:ARRAY [1..2,3..4] OF INT:= 1,3(7) ;        (* short for 1,7,7,7* )
Arr3:ARRAY [1..2,2..3,3..4] OF INT:= 2(0),4(4),2,3;   (* short for 0,0,4,4,4,4,2,3* )
```

Example 2: initialization of arrays of structures

```
TYPE STRUCT1:
STRUCT
p1:int;
p2:int;
p3:dword;
END_STRUCT
END_ TYPE
ARRAY[1..3] OF STRUCT1:=(p1:=1,p2:=10,p3:= 3),(p1:=2,p2:=0,p3:=2),
(p1:=4,p2:=5,p3:=1);
```

Example 3: partial initialization of arrays

```
Arr1:ARRAY [1..10] OF BYTE:= 1,2;
```

The elements not initialized will have the default initial value. In example 3, the elements from [3] to [10] are initialized to 0.

Array Access

Use the syntax below to access two-dimensional arrays:

```
<Field_Name>[Index1,Index2]
```

For example:

```
Array access:
Card_game[9,2]
```

Note:

If CheckBounds is used to define functions, the program can automatically detect array index out of bound error. The keyword of functions must be CheckBounds, the program of CheckBounds is shown in figure 3-2-1.

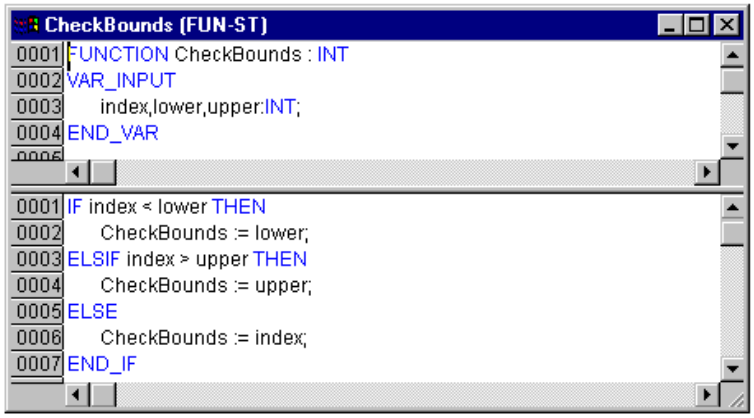


Figure 3-2-1

If there is no CheckBounds program in the projects, A[B] in figure 3-2-2 become A[10], and it exceeds the maximum index 7 of array A, which will result in errors during compilation.

But if CheckBounds is defined in the project, A[B] will be considered as A[7], A Boolean value of TRUE will be assigned to A[7], and there will be no error during compilation.

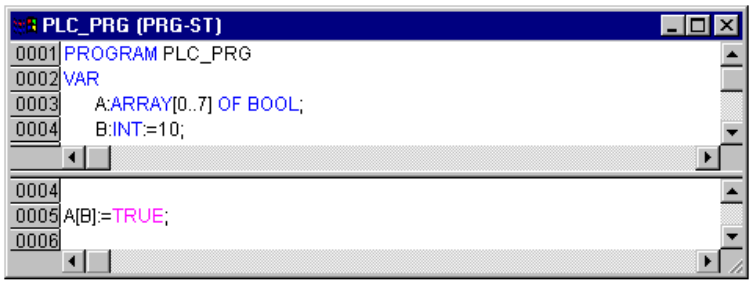


Figure 3-2-2

3.2.2 Pointers

During the program executing, the addresses of variables/function blocks can be obtained by pointers. Pointers can point to any type of data or function blocks, including user-defined data types. The syntax of pointer declaration is:

<pointer name>： POINTER TO <data type/function block type>

The address of a variable or instruction instance can be extracted by the address extraction instruction ADR. The data stored in a particular address can be retrieved by adding a symbol "^" after the pointer that points to the address.

Example	
Program	Meaning
pt:POINTER TO INT; Var_int1:INT:= 5; Var_int2:INT; pt:= ADR(Var_int1); Var_int2:= pt^;	(*define a pointer of pt as INT*) (*define variable Var_int1 as INT, set it to 5*) (*define variable Var_int2 as INT*) (*get the address of Var_int1, assign it to pointer pt*) (*assign the data value stored in address pt to Var_int2, Var_int2=5*)

3.2.3 Enumerations

An enumeration is a customized numeric type composed of a list of constants, and these constants are called enumeration values.

The enumeration values can be identified in the entire program as long as it is declared in a POU. It is suggested to create an enumeration in "Data types" tab  at lower left corner of PowerPro, and create a new enumeration value by adding an object (Add Object) in the Data types tab. .

Enumeration starts with the keyword TYPE, and ends with the keyword END_TYPE. The syntax of the Enumeration declaration is:

```
TYPE<Identifier>:(<Enum_0>,<Enum_1>,…,<Enum_n>);
END_TYPE
```

Note
The enumeration values may include identifiers. During initialization, identifiers will be initialized as one of Enumeration values. Any number can be assigned to enumeration values. If the enumeration values are not initialized, the initial values will be incremented starting from 0. The defined enumeration values are compatible with other data types, which can be used as if they are of INT data types.

Example	
Program	Comments

TYPE TRAFFIC_SIGNAL: (Red, Yellow, Green:=10);
END_TYPE
 	
TRAFFIC_SIGNAL1: TRAFFIC_SIGNAL;
TRAFFIC_SIGNAL1:=0;
FOR i:= Red TO Green DO
i:= i + 1;
END_FOR
	(*the initial values for each color is: Red=0, Yellow=1, Green=10*)
	
	(*the value of traffic signal is Red*)

The same enumeration values cannot be re-assigned.

Example:

```
TRAFFIC_SIGNAL: (red, yellow, green);
COLOR: (blue, white, red);
```

Error: red cannot be used by TRAFFIC_SIGNAL and COLOR at the same time.

3.2.4 Structures

Structures can be created in the "Data types" tab  at the lower left corner of PowerPro by adding a new object of the data type ("Add Object").

A structure starts with the keywords TYPE and STRUCT, and ends with END_STRUCT and END_TYPE.

Syntax of Structure Declaration

```
TYPE <structure name>:
STRUCT
<Variable declaration1>
 :
< Variable declaration n>
END_STRUCT
END_TYPE
```

<structure name> is a data type that can be identified within the whole project, and can be quoted like a standard data type, however, the variable address of STRUCT cannot be specified ("AT" command cannot be used to specify the address of the variable after variable names.)

Example (a STRUCT named Polygonline):

```
TYPE Polygonline:
STRUCT
Start:ARRAY [1..2] OF INT;
Point1:ARRAY [1..2] OF INT;
Point2:ARRAY [1..2] OF INT;
Point3:ARRAY [1..2] OF INT;
Point4:ARRAY [1..2] OF INT;
End:ARRAY [1..2] OF INT;
END_STRUCT
END_TYPE
```

Example (initialization STRUCT):

```
Poly_1:polygonline:= (Start:=3,3,Point1 =5,2,Point2:=7,3,Point3:=8,5,
Point4:=5,7, End :=3,5);
```

Accessing members of a structure

```
<struct name>.<name of struct member>
```

Example:

If in a structure "Week", one of the members is "Monday", it can be accessed as follows:

```
Week.Monday
```

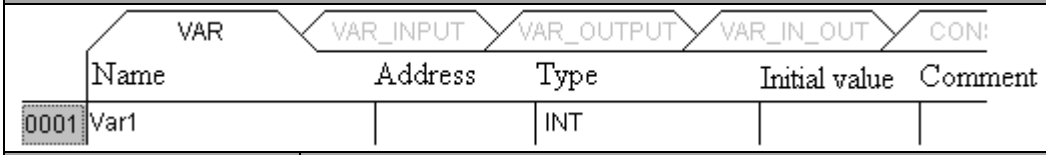
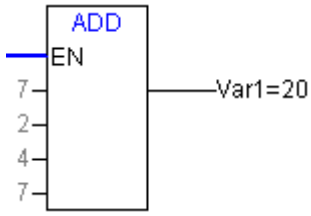
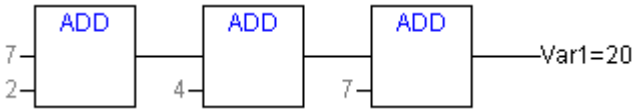
Chapter 4

Basic Instructions

4.1 ARITHMETIC OPERATORS

4.1.1 ADD—Addition

- Function: Addition of two (or more) variables or constants.
- Input/output data types: BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, TIME.

<table>
<tr><td colspan="2">Application Example</td></tr>
<tr><td colspan="2">Variable Definition</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>Programming Language</td><td>Program</td></tr>
<tr><td>Ladder Digram (LD)</td><td> 
</td></tr>
<tr><td>Structured Text (ST)</td><td>Var1:=7+2+4+7; (*result Var1is 20*)</td></tr>
<tr><td>Instruction List (IL)</td><td>LD 7
ADD 2,4,7
ST Var1 (* result Var1 is 20*)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td> 
</td></tr>
</table>

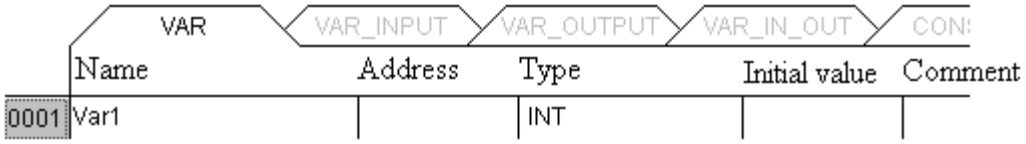
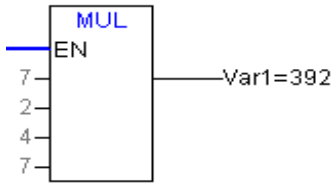
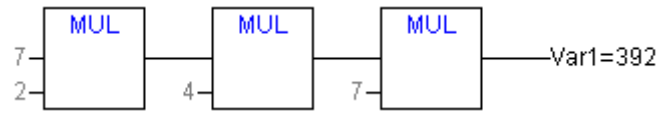
Note:

- Addition operation can also apply to TIME , for example t#45s + t#50s = t#1m35 s.
- The selected output type shall be consistent with the output result, otherwise may cause data error; it is the same for MUL, SUB, DIV instructions

4.1.2 MUL—Multiplication

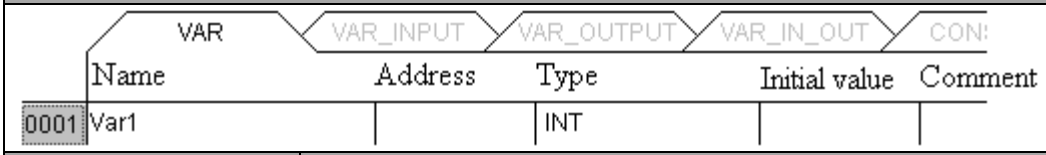
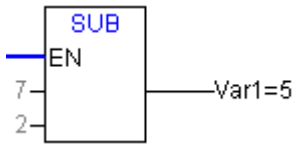
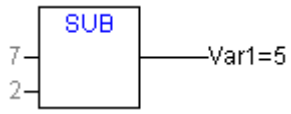
- Function: Multiplication of two (or more) variables or constants.

- Input/output data types: BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL.

<table>
<tr><td colspan="2">Application Example</td></tr>
<tr><td colspan="2">Variable Declaration</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>Programming Language</td><td>Program</td></tr>
<tr><td>Ladder Diagram (LD)</td><td> 
</td></tr>
<tr><td>Structured Text (ST)</td><td>Var1:=7*2*4*7; (*result Var1 is 392*)</td></tr>
<tr><td>Instruction List (IL)</td><td>LD 7
MUL 2,4,7
ST Var1 (* result Var1 is 392*)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td> 
</td></tr>
</table>

4.1.3 SUB—Subtraction

- Function: Subtraction of two (or more) variables or constants.
- Input/output data types: BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, TOD.

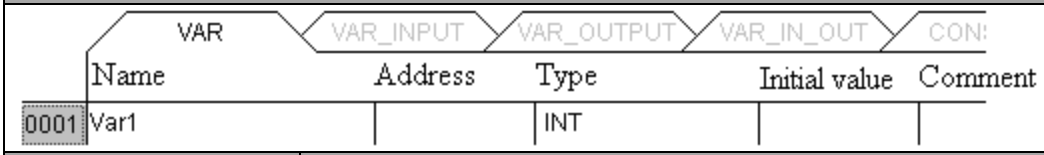
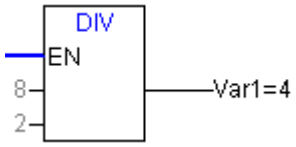
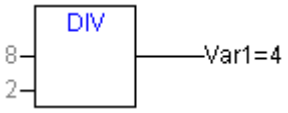
<table>
<tr><td colspan="2">APPLICATION EXAMPLE</td></tr>
<tr><td colspan="2">Variable Declaration</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>Programming Language</td><td>Program</td></tr>
<tr><td>Ladder Diagram (LD)</td><td> 
</td></tr>
<tr><td>Structured Text (ST)</td><td>Var1:=7-2; (*result Var1 is 5*)</td></tr>
<tr><td>Instruction List (IL)</td><td>LD 7
SUB 2
ST Var1 (*result Var1 is 5*)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td> 
</td></tr>
</table>

Note:

- SUB can also apply to TIME variables, for example t#1m35s - t#50s = t#45s, but the result shall not be negative.
- SUB can also apply to TOD variables, for example TOD#23:40:30- TOD#00:30:20=T#1390m10s0ms, but the result shall not be negative.

4.1.4 DIV—Division

- Function: Division of one variable or constant by another.
- Input/output data types: BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL.

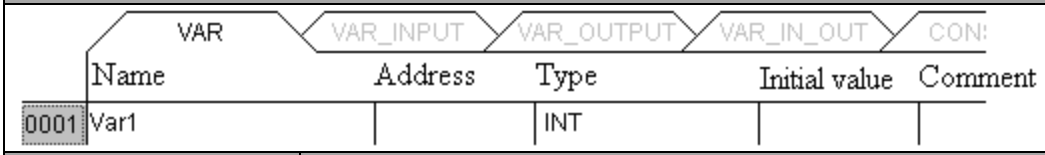
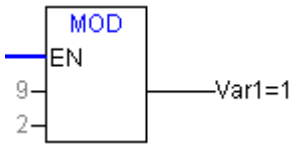
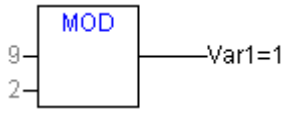
<table>
<tr><td colspan="2">APPLICATION EXAMPLE</td></tr>
<tr><td colspan="2">Variable Declaration</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>Programming Language</td><td>Program</td></tr>
<tr><td>Ladder Diagram (LD)</td><td> 
</td></tr>
<tr><td>Structured Text (ST)</td><td>Var1:=8/2; (*result Var1 is 4*)</td></tr>
<tr><td>Instruction List (IL)</td><td>LD 8
DIV 2
ST Var1 (* result Var1 is 4*)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td> 
</td></tr>
</table>

Note:

- When using DIV in projects, CheckDivByte, CheckDivWord, CheckDivDWord, CheckDivRealand instructions (see 4.15) may be used to check if the divisor is zero to avoid this phenomenon.

4.1.5 MOD—Modulo Division

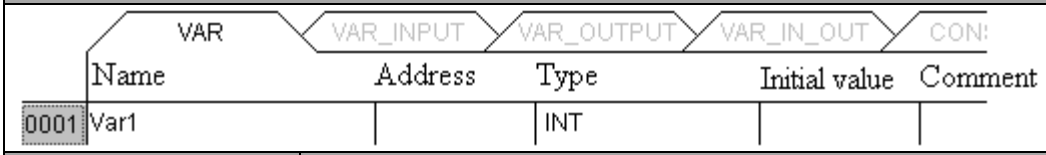
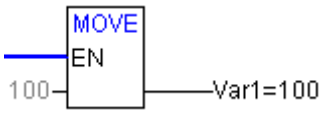
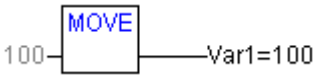
- Function: Modulo Division of one variable or constants by another, get the remainder. The result shall be an integer.
- Input/output data types: BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT.

<table>
<tr><td colspan="2">APPLICATION EXAMPLE</td></tr>
<tr><td colspan="2">Variable Declaration</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>Programming Language</td><td>Program</td></tr>
<tr><td>Ladder Diagram (LD)</td><td> 
</td></tr>
<tr><td>Structured Text (ST)</td><td>Var1:=9 MOD 2; (*result Var1 is 1*)</td></tr>
<tr><td>Instruction List (IL)</td><td>LD 9
MOD 2
ST Var1 (*result Var1 is 1*)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td> 
</td></tr>
</table>

4.2 MOVE

4.2.1 MOVE–Move Instruction

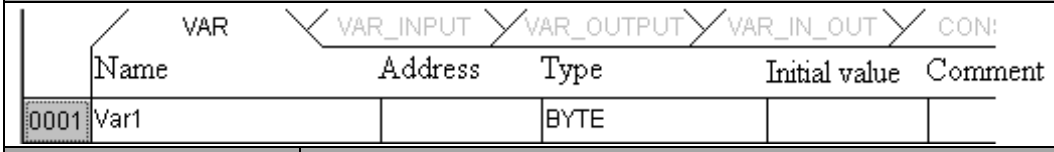
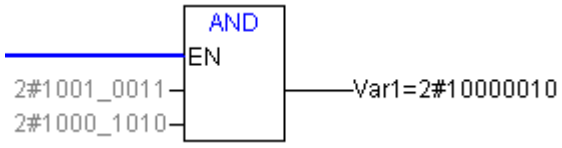
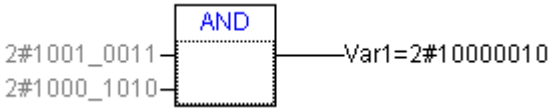
- Function: Assignment of a variable or constant to another variable.
- Input/output data types: BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, TIME, DT, BOOL, STRING, ARRAY.

<table>
<tr><td colspan="2">APPLICATION EXAMPLE</td></tr>
<tr><td colspan="2">Variable Declaration</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>Programming Language</td><td>Program</td></tr>
<tr><td>Ladder Diagram (LD)</td><td> 
</td></tr>
<tr><td>Structured Text (ST)</td><td>Var1:=100; (*result Var1 is 100*)</td></tr>
<tr><td>Instruction List (IL)</td><td>LD 100
MOVE
ST Var1 (* result Var1 is 100*)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td> 
</td></tr>
</table>

4.3 LOGICAL OPERATORS

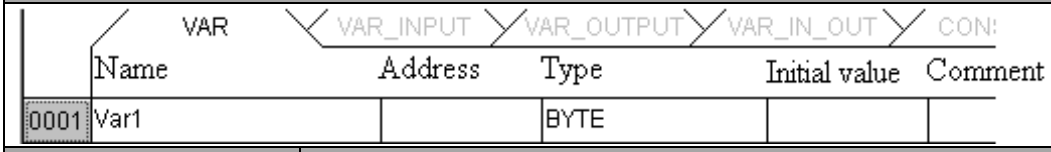
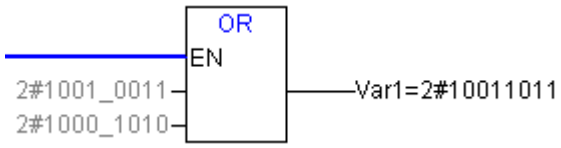
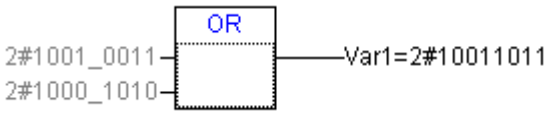
4.3.1 AND—Logical AND of Bit Operands

- Function: AND operation of variables or constants.
- Input/output data types: BOOL, BYTE, WORD and DWORD.

<table>
<tr><td colspan="2">APPLICATION EXAMPLE</td></tr>
<tr><td colspan="2">Variable Declaration</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>Programming Language</td><td>Program</td></tr>
<tr><td>Ladder Diagram (LD)</td><td> 
</td></tr>
<tr><td>Structured Text (ST)</td><td>Var1:=2#1001_0011 AND 2#1000_1010;
(*result Var1 is 2#10000010*)</td></tr>
<tr><td>Instruction List (IL)</td><td>LD 2#1001_0011
AND 2#1000_1010
ST Var1 (*result Var1 is 2#10000010*)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td> 
</td></tr>
</table>

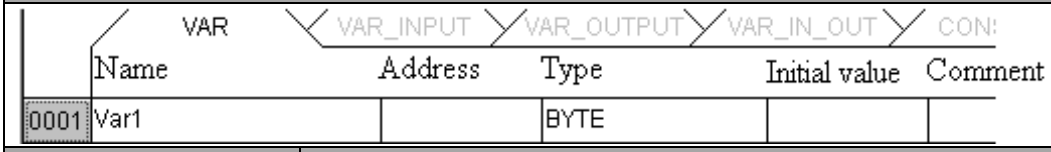
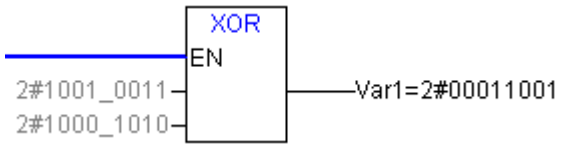
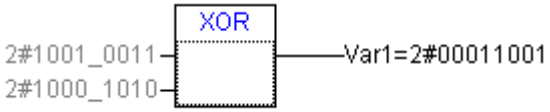
4.3.2 OR—Logical OR of Bit Operands

- Function: OR operation of variables or constants.
- Input/output data types: BOOL, BYTE, WORD and DWORD.

<table>
<tr><td colspan="2">APPLICATION EXAMPLE</td></tr>
<tr><td colspan="2">Variable Declaration</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>Programming Language</td><td>Program</td></tr>
<tr><td>Ladder Diagram (LD)</td><td> 
</td></tr>
<tr><td>Structured Text (ST)</td><td>Var1:=2#1001_0011 OR 2#1000_1010 ;
(*result Var1 is 2#10011011*)</td></tr>
<tr><td>Instruction List (IL)</td><td>LD 2#1001_0011
OR 2#1000_1010
ST Var1 (*result Var1 is 2#10011011*)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td> 
</td></tr>
</table>

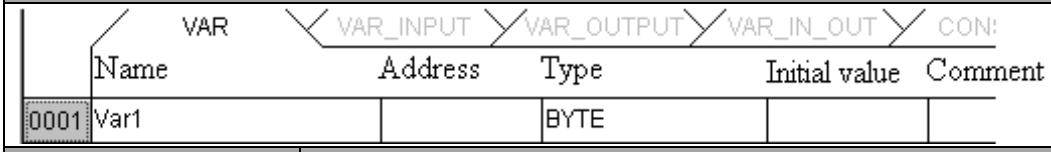
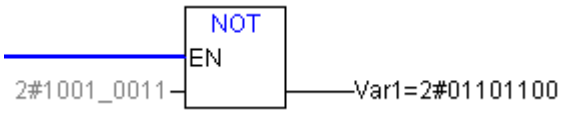
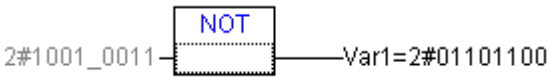
4.3.3 XOR—Logical XOR of Bit Operands

- Function: XOR operation of variables or constants.
- Input/output data types: BOOL, BYTE, WORD and DWORD.

<table>
<tr><td colspan="2">APPLICATION EXAMPLE</td></tr>
<tr><td colspan="2">Variable Declaration</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>Programming Language</td><td>Program</td></tr>
<tr><td>Ladder Diagram (LD)</td><td> 
</td></tr>
<tr><td>Structured Text (ST)</td><td>Var1:=2#1001_0011 XOR 2#1000_1010 ;
(*result Var1 is 2#00011001*)</td></tr>
<tr><td>Instruction List (IL)</td><td>LD 2#1001_0011
XOR 2#1000_1010
ST Var1 (*result Var1 is 2#00011001*)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td> 
</td></tr>
</table>

4.3.4 NOT—Logical NOT of Bit Operands

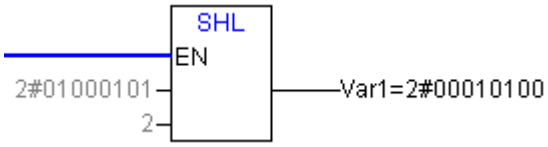
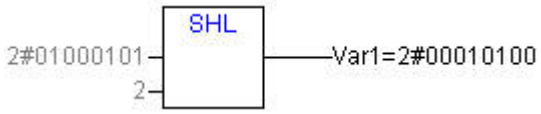
- Function: Logical NOT operation of variables or constant. Bitwise NOT.
- Input/output data types: BOOL, BYTE, WORD and DWORD.

<table>
<tr><td colspan="2">APPLICATION EXAMPLE</td></tr>
<tr><td colspan="2">Variable Declaration</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>Programming Language</td><td>Program</td></tr>
<tr><td>Ladder Diagram (LD)</td><td> 
</td></tr>
<tr><td>Structured Text (ST)</td><td>Var1:= NOT 2#1001_0011 ;
(*result Var1 is 2#01101100*)</td></tr>
<tr><td>Instruction List (IL)</td><td>LD 2#1001_0011
NOT
ST Var1 (*result Var1 is 2#01101100*)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td> 
</td></tr>
</table>

4.4 BIT-SHIFT OPERATORS

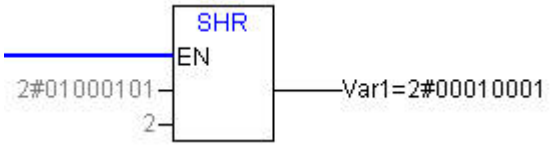
4.4.1 SHL—Left-shift

- Function: Bitwise left-shift of an operand, lose the left bits and fill the right bits with 0.
- Input/output data types: BYTE, INT, WORD, DWORD, SINT and UINT.

<table>
<tr><td colspan="2">APPLICATION EXAMPLE</td></tr>
<tr><td colspan="2">Variable Declaration</td></tr>
<tr><td colspan="2">
VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CON
<table>
<tr><th></th><th>Name</th><th>Address</th><th>Type</th><th>Initial value</th><th>Comment</th></tr>
<tr><td>0001</td><td>Var1</td><td></td><td>BYTE</td><td></td><td></td></tr>
<tr><td>0002</td><td>Var2</td><td></td><td>INT</td><td></td><td></td></tr>
</table>
</td></tr>
<tr><td>Programming Language</td><td>Program</td></tr>
<tr><td>Ladder Diagram (LD)</td><td> 
</td></tr>
<tr><td>Structured Text (ST)</td><td>Var1:=SHL(16#45,2); (* result Var1 is 16#14*)
Var2:=SHL(16#45,2); (* result Var2 is 16#0114*)
Note: in the example above, although the values of inputs are the same, but the types of outputs are different, so the results Var1 and Var2 are different.</td></tr>
<tr><td>Instruction List (IL)</td><td>LD 16#45
SHL 2
ST Var1 (* resultVar1 is 16#14*)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td> 
</td></tr>
</table>

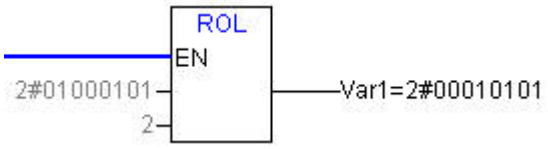
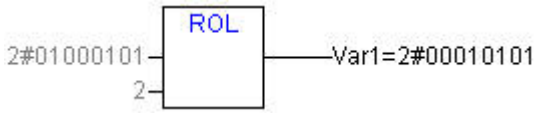
4.4.2 SHR—Right-shift

- Function: Bitwise right-shift of an operand, lose the right bits and fill the left bits with 0.
- Input/output data types: BYTE, INT, WORD, DWORD, SINT and UINT.

<table>
<tr><td colspan="2">APPLICATION EXAMPLE</td></tr>
<tr><td colspan="2">Variable Declaration</td></tr>
<tr><td colspan="2">
VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CON
<table>
<tr><th></th><th>Name</th><th>Address</th><th>Type</th><th>Initial value</th><th>Comment</th></tr>
<tr><td>0001</td><td>Var1</td><td></td><td>BYTE</td><td></td><td></td></tr>
<tr><td>0002</td><td>Var2</td><td></td><td>INT</td><td></td><td></td></tr>
</table>
</td></tr>
<tr><td>Programming Language</td><td>Program</td></tr>
<tr><td>Ladder Diagram (LD)</td><td> 
</td></tr>
<tr><td>Structured Text (ST)</td><td>Var1:=SHR(16#45,2); (* result Var1 is 16#11*)
Var2:=SHR(16#45,2); (* result Var2 is 16#0011*)</td></tr>
<tr><td>Instruction List (IL)</td><td>LD 16#45
SHR 2
ST Var1 (* result Var1 is 16#11*)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td> 
</td></tr>
</table>

4.4.3 ROL—Rotation to the Left

- Function: Bitwise rotation of an operand to the left, the left bit moved out to fill the most right bit.
- Input/output data types: BYTE, INT, WORD, DWORD, SINT and UINT.

<table>
<tr><td colspan="2">APPLICATION EXAMPLE</td></tr>
<tr><td colspan="2">Variable Declaration</td></tr>
<tr><td colspan="2">
VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CON:
<table>
<tr><th></th><th>Name</th><th>Address</th><th>Type</th><th>Initial value</th><th>Comment</th></tr>
<tr><td>0001</td><td>Var1</td><td></td><td>BYTE</td><td></td><td></td></tr>
<tr><td>0002</td><td>Var2</td><td></td><td>INT</td><td></td><td></td></tr>
</table>
</td></tr>
<tr><td>Programming Language</td><td>Program</td></tr>
<tr><td>Ladder Diagram (LD)</td><td> 
</td></tr>
<tr><td>Structured Text (ST)</td><td>Var1:=ROL(16#45,2); (* result Var1 is 16#15*)
Var2:= ROL (16#45,2); (* result Var2 is 16#0114*)
Note: in the process of ROL, although the values of inputs are the same, but the types of outputs are different, so the results Var1 and Var2 are different.</td></tr>
<tr><td>Instruction List (IL)</td><td>LD 16#45
ROL 2
ST Var1 (* result Var1 is 16#15*)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td> 
</td></tr>
</table>

4.4.4 ROR—Rotation to the Right

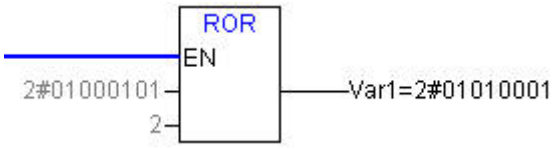
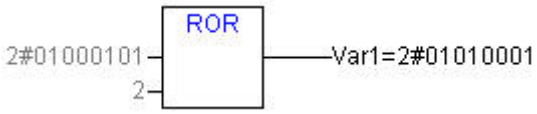
- Function: Bitwise rotation of an operand to the right, the right bit moved out to fill the most left most.
- Input/output data types: BYTE, INT, WORD, DWORD, SINT and UINT.

APPLICATION EXAMPLE

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CON

	Name	Address	Type	Initial value	Comment
0001	Var1		BYTE		
0002	Var2		INT		

Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	Var1:=ROR(16#45,2) (* result Var1 is 16#51*) Var2:= ROR (16#45,2) (* result Var2 is 16#4011*) Note: in the process of ROR, although the values of inputs are the same, but the types of outputs are different, so the results Var1 and Var2 are different.
Instruction List (IL)	LD 16#45 ROR 2 ST Var1 (* result Var1 is 16#51*)
Function Block Diagram (FBD)	

4.5 SELECTION OPERATORS

All selection instructions can be applied performed with variables. However, in order to give a clearer explanation of the operators, the following examples only use constant operators. The selected input data type shall not have longer length than the selected output data type.

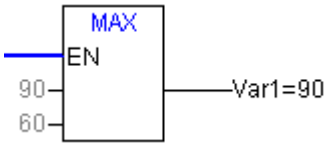
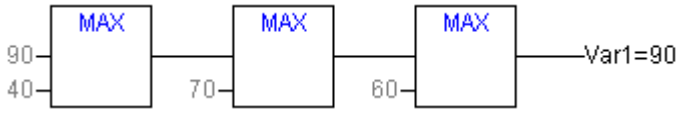
4.5.1 SEL—Binary Selection

- Function: Select one of the inputs as the output by selection switch, output the first data when selection switch is FALSE, and output the second data when selection switch is TRUE.
- Instruction format: OUT := SEL(G, IN0, IN1), G is the selection switch, IN0 is the first data and IN1 is the second data.
- Input/output data types:
- G must be BOOL, IN0, IN1and the output data can be any type

<table>
<tr><td colspan="2">APPLICATION EXAMPLE</td></tr>
<tr><td colspan="2">Variable Declaration</td></tr>
<tr><td colspan="2">VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CON
Name | Address | Type | Initial value | Comment
0001 Var1 | | INT | |
0002 Var2 | | INT | |</td></tr>
<tr><td>Programming Language</td><td>Program</td></tr>
<tr><td>Ladder Diagram (LD)</td><td>SEL
EN
TRUE
2
6
Var1=6</td></tr>
<tr><td>Structured Text (ST)</td><td>Var1:=SEL(TRUE,3,4); (*result Var1 is 4 *)</td></tr>
<tr><td>Instruction List (IL)</td><td>LD TRUE
SEL 2,6
ST Var1 (*result Var1 is 6*)
LD FALSE
SEL 2,6
ST Var2 (*result Var2 is 2*)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td>SEL
TRUE
2
6
Var1=6</td></tr>
</table>

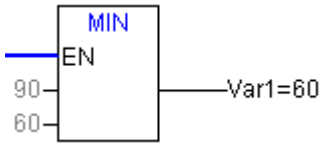
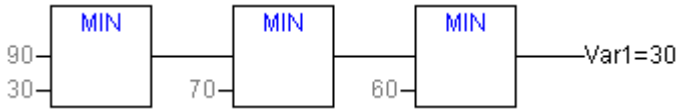
4.5.2 MAX—Maximum

- Function: Select the maximum from a number of input data as the output .
- Instruction format: OUT:=MAX(IN0, IN1), IN0 stands for the first input, IN1 stands for the second input, and OUT stands for the output.
- Input/output data types: IN0, IN1 and OUT can be any type of variables.

<table>
<tr><td colspan="2">APPLICATION EXAMPLE</td></tr>
<tr><td colspan="2">Variable Declaration</td></tr>
<tr><td colspan="2">
VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CON
<table>
<tr><th></th><th>Name</th><th>Address</th><th>Type</th><th>Initial value</th><th>Comment</th></tr>
<tr><td>0001</td><td>Var1</td><td></td><td>INT</td><td></td><td></td></tr>
<tr><td>0002</td><td>Var2</td><td></td><td>INT</td><td></td><td></td></tr>
</table>
</td></tr>
<tr><td>Programming Language</td><td>Program</td></tr>
<tr><td>Ladder Diagram (LD)</td><td> 
</td></tr>
<tr><td>Structured Text (ST)</td><td>Var1:=MAX(90,60); (*result Var1 is 90 *)
Var2:=MAX(40,MAX(90,60)); (*result Var2 is 90 *)</td></tr>
<tr><td>Instruction List (IL)</td><td>LD 90
MAX 40
MAX 70
MAX 60
ST Var1 (* result Var1 is 90*)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td> 
</td></tr>
</table>

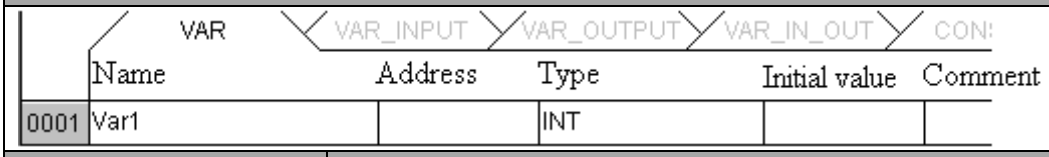
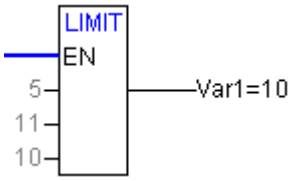
4.5.3 MIN—Minimum

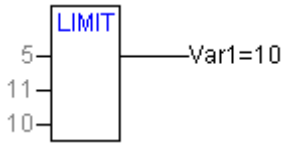
- Function: Select the minimum from a number of input data as the output.
- Instruction format: OUT:=MIN(IN0, IN1), IN0 stands for the first input, IN1 stands for the second input, and OUT stands for the output.
- Input/output data types: IN0, IN1 and OUT can be any type of variables.

<table>
<tr><td colspan="2">APPLICATION EXAMPLE</td></tr>
<tr><td colspan="2">Variable Declaration</td></tr>
<tr><td colspan="2">
VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CON
<table>
<tr><th></th><th>Name</th><th>Address</th><th>Type</th><th>Initial value</th><th>Comment</th></tr>
<tr><td>0001</td><td>Var1</td><td></td><td>INT</td><td></td><td></td></tr>
<tr><td>0002</td><td>Var2</td><td></td><td>INT</td><td></td><td></td></tr>
</table>
</td></tr>
<tr><td>Programming Language</td><td>Program</td></tr>
<tr><td>Ladder Diagram (LD)</td><td> 
</td></tr>
<tr><td>Structured Text (ST)</td><td>Var1:=MIN(90,30); (*result Var1 is 30 *)
Var2:=MIN(MIN(90,30),60); (*result Var2 is 30 *)</td></tr>
<tr><td>Instruction List (IL)</td><td>LD 90
MIN 30
MIN 40
MIN 70
ST Var1 (* result Var1 is 30*)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td> 
</td></tr>
</table>

4.5.4 LIMIT—Limiting

- Function: Check whether the input is within the range between the Max and the Min. If the input is within the range, LIMIT will directly return the value as the output. Should the input value exceed the upper limit Max, LIMIT will return the Max as the output. Should the input value fall below the lower limit Min, the output will be Min.
- Instruction format: OUT := LIMIT(Min, IN, Max)
- Input/output data types: IN and OUT can be any type of variables.

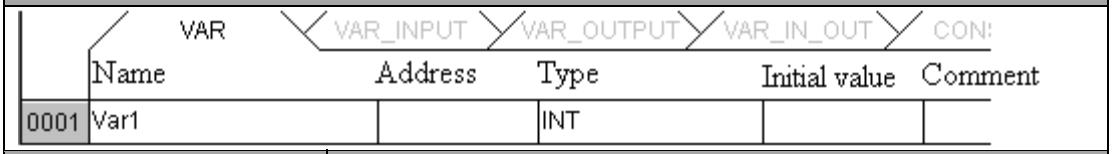
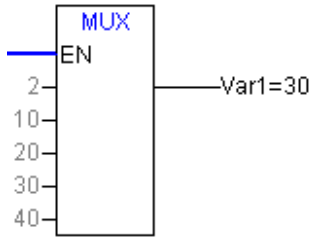
<table>
<tr><td colspan="2">APPLICATION EXAMPLE</td></tr>
<tr><td colspan="2">Variable Declaration</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>Programming Language</td><td>Program</td></tr>
<tr><td>Ladder Diagram (LD)</td><td> 

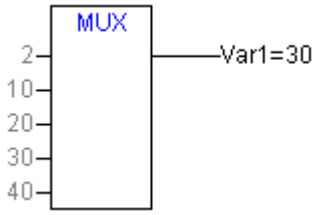
(*11 is input, 5 is minimum, 10 is maximum*)</td></tr>
<tr><td>Structured Text (ST)</td><td>Var1:=LIMIT(30,90,80); (*result Var1 is 80 *)</td></tr>
<tr><td>Instruction List (IL)</td><td>LD 90
LIMIT 30，80
ST Var1 (*result Var1 is 80*)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td> 

(*11 is input, 5 is minimum, 10 is maximum *)</td></tr>
</table>

4.5.5 MUX—Multiplexer

- Function: MUX selects a value from a group of values by the controlling number K.
- Instruction format: OUT:=MUX(K,IN0,…,INn), K is controlling number, IN0,…,INn are inputs, OUT is output. Select INk as output when controlling number is K.
- Input/output data types: IN0,…, INn and OUT can be any type of variables, K must be BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT or UDINT.

<table>
<tr><td colspan="2">APPLICATION EXAMPLE</td></tr>
<tr><td colspan="2">Variable Declaration
 
</td></tr>
<tr><td>Programming Language</td><td>Program</td></tr>
<tr><td>Ladder Diagram (LD)</td><td> 

(*2 is controlling data, corresponds to 30, so output 30 *)</td></tr>
<tr><td>Structured Text (ST)</td><td>Var1:=MUX(0,30,40,50,60,70,80); (* result Var1 is 30*)</td></tr>
<tr><td>Instruction List (IL)</td><td>LD 0
MUX 30，40，50，60，70，80
ST Var1 (* result Var1 is 30*)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td> 

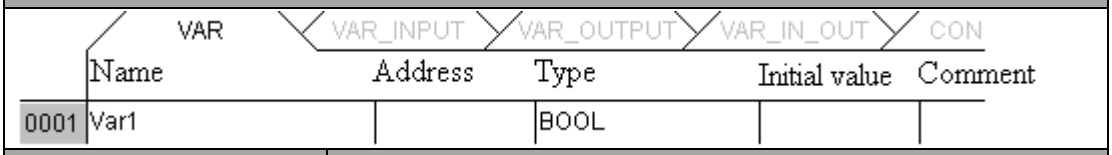
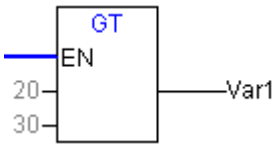
(*2 is control data, corresponds to 30, so output 30 *)</td></tr>
</table>

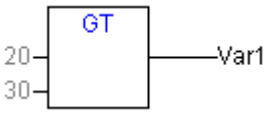
4.6 COMPARISON OPERATORS

All comparison operations can also be executed with variables. However, in order to give a clearer explanation of the operators, the following examples only use constant operators..

4.6.1 GT—Greater Than

- Function: Compare the value of two operands, output TRUE when the first value is greater than the second one, otherwise output FALSE.
- Input/output data types:
- Input data type: BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, TIME, DATE, TOD, DT and STRING;
- Output data type: BOOL.

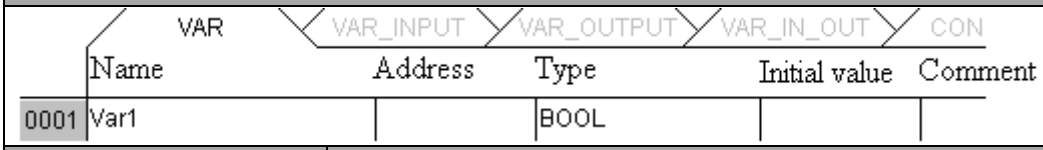
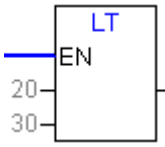
<table>
<tr><td colspan="2">APPLICATION EXAMPLE</td></tr>
<tr><td colspan="2">Variable Declaration</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>Programming Language</td><td>Program</td></tr>
<tr><td>Ladder Diagram (LD)</td><td> 

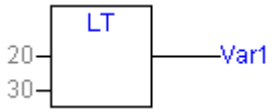
(* result Var1 is FALSE *)</td></tr>
<tr><td>Structured Text (ST)</td><td>Var1:=20>30;</td></tr>
<tr><td>Instruction List (IL)</td><td>LD 20
GT 30
ST Var1 (* result Var1 is FALSE *)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td> 

(* result Var1 is FALSE *)</td></tr>
</table>

4.6.2 LT—Less Than

- Function: Output TRUE when the first value is less than the second one, otherwise output FALSE.
- Input/output data types:
- Input data type: BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, TIME, DATE, TOD, DT and STRING;
- Output data type: BOOL.

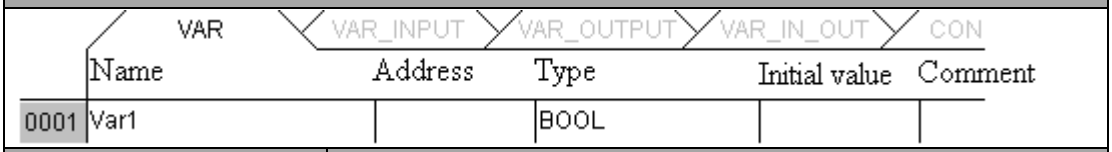
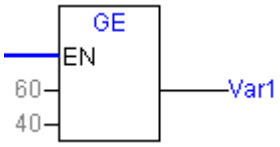
<table>
<tr><td colspan="2">APPLICATION EXAMPLE</td></tr>
<tr><td colspan="2">Variable Declaration</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>Programming Language</td><td>Program</td></tr>
<tr><td>Ladder Diagram (LD)</td><td> 

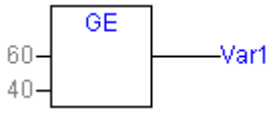
(* result Var1 is TRUE *)</td></tr>
<tr><td>Structured Text (ST)</td><td>VAR1:=20<30; (* result Var1 is TRUE *)</td></tr>
<tr><td>Instruction List (IL)</td><td>LD 20
LT 30
ST Var1 (* result Var1 is TRUE *)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td> 

(* result Var1 is TRUE *)</td></tr>
</table>

4.6.3 GE—Greater or Equal

- Function: Output TRUE when the first value is greater than or equal to the second one, otherwise output FALSE.
- Input/output data types:
- Input data type: BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, TIME, DATE, TOD, DT and STRING;
- Output data type: BOOL.

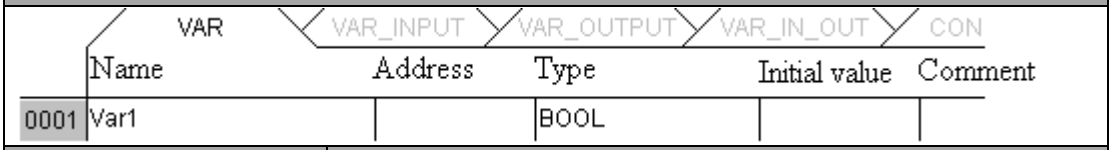
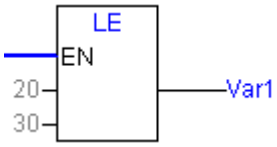
<table>
<tr><td colspan="2">APPLICATION EXAMPLE</td></tr>
<tr><td colspan="2">Variable Declaration</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>Programming Language</td><td>Program</td></tr>
<tr><td>Ladder Diagram (LD)</td><td> 

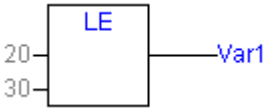
(*result Var1 is TRUE*)</td></tr>
<tr><td>Structured Text (ST)</td><td>VAR1:=60>=40; (*result Var1 is TRUE*)</td></tr>
<tr><td>Instruction List (IL)</td><td>LD 60
GE 40
ST Var1 (*result Var1 is TRUE*)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td> 

(*result Var1 is TRUE*)</td></tr>
</table>

4.6.4 LE—Less or Equal

- Function: Output TRUE when the first value is less than or equal to the second one, otherwise output FALSE.
- Input/output data types:
- Input data type: BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, TIME, DATE, TOD, DT and STRING;
- Output data type: BOOL.

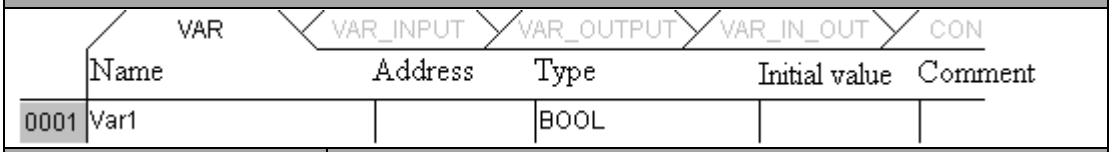
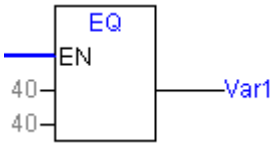
<table>
<tr><td colspan="2">APPLICATION EXAMPLE</td></tr>
<tr><td colspan="2">Variable Declaration</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>Programming Language</td><td>Program</td></tr>
<tr><td>Ladder Diagram (LD)</td><td> 

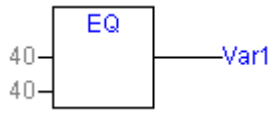
(*result Var1 is TRUE*)</td></tr>
<tr><td>Structured Text (ST)</td><td>VAR1:=20<=30; (*result Var1 is TRUE*)</td></tr>
<tr><td>Instruction List (IL)</td><td>LD 20
LE 30
ST Var1 (*result Var1 is TRUE*)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td> 

(*result Var1 is TRUE*)</td></tr>
</table>

4.6.5 EQ—Equal

- Function: Output TRUE when the first value is equal to the second one, otherwise output FALSE.
- Input/output data types:
- Input data type: BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, TIME, DATE, TOD, DT and STRING;
- Output data type: BOOL.

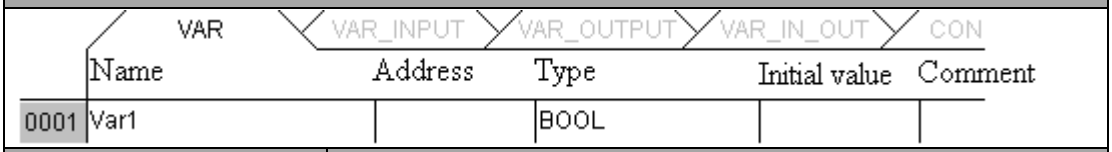
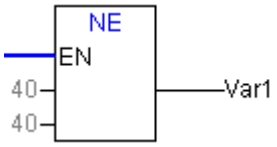
<table>
<tr><td colspan="2">APPLICATION EXAMPLE</td></tr>
<tr><td colspan="2">Variable Declaration</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>Programming Language</td><td>Program</td></tr>
<tr><td>Ladder Diagram (LD)</td><td> 

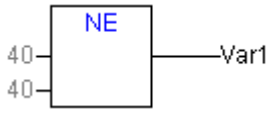
(*result Var1 is TRUE*)</td></tr>
<tr><td>Structured Text (ST)</td><td>VAR1:=40=40; (*result Var1 is TRUE*)</td></tr>
<tr><td>Instruction List (IL)</td><td>LD 40
EQ 40
ST Var1 (*result Var1 is TRUE*)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td> 

(*result Var1 is TRUE*)</td></tr>
</table>

4.6.6 NE—Not Equal

- Function: Output TRUE when the first value is not equal to the second one, otherwise output FALSE.
- Input/output data types:
- Input data type: BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, TIME, DATE, TOD, DT and STRING;
- Output data type: BOOL.

<table>
<tr><td colspan="2">APPLICATION EXAMPLE</td></tr>
<tr><td colspan="2">Variable Declaration</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>Programming Language</td><td>Program</td></tr>
<tr><td>Ladder Diagram (LD)</td><td> 

(*result Var1 is FALSE*)</td></tr>
<tr><td>Structured Text (ST)</td><td>VAR1:=40<>40; (*result Var1 is FALSE*)</td></tr>
<tr><td>Instruction List (IL)</td><td>LD 40
NE 40
ST Var1 (*result Var1 is FALSE*)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td> 

(*result Var1 is FALSE*)</td></tr>
</table>

4.7 DATA TYPE CONVERSION

PowerPro provides 240 data type conversion instructions for the conversions between different data types.

Syntax：<TYPE1>_TO_<TYPE2>

- It is disabled to convert a “larger” data type into a “smaller” data type. Information may be lost when converting a “larger” data type into a “smaller” data type.
- The high byte will be ignored if the value to be converted is over the range of target data type. For example, INT_TO_BYTE or DINT_TO_WORD.
- In <TYPE>_TO_STRING, the string is generated from the left. If the length of string is smaller than the length of <TYPE>, the right part will be cut.

List of Data Type Conversion Instructions

- All data type conversion instructions are listed in table 4-7-1.

BOOL_TO_<TYPE>	BYTE_TO_<TYPE>	DATE_TO_<TYPE>	DINT_TO_<TYPE>
BOOL_TO_BYTE	BYTE_TO_BOOL	DATE_TO_BOOL	DINT_TO_BOOL
BOOL_TO_DATE	BYTE_TO_DATE	DATE_TO_BYTE	DINT_TO_BYTE
BOOL_TO_DINT	BYTE_TO_DINT	DATE_TO_DINT	DINT_TO_DATE
BOOL_TO_DT	BYTE_TO_DT	DATE_TO_DT	DINT_TO_DT
BOOL_TO_DWORD	BYTE_TO_DWORD	DATE_TO_DWORD	DINT_TO_DWORD
BOOL_TO_INT	BYTE_TO_INT	DATE_TO_INT	DINT_TO_INT
BOOL_TO_REAL	BYTE_TO_REAL	DATE_TO_REAL	DINT_TO_REAL
BOOL_TO_SINT	BYTE_TO_SINT	DATE_TO_SINT	DINT_TO_SINT
BOOL_TO_STRING	BYTE_TO_STRING	DATE_TO_STRING	DINT_TO_STRING
BOOL_TO_TIME	BYTE_TO_TIME	DATE_TO_TIME	DINT_TO_TIME
BOOL_TO_TOD	BYTE_TO_TOD	DATE_TO_TOD	DINT_TO_TOD
BOOL_TO_UDINT	BYTE_TO_UDINT	DATE_TO_UDINT	DINT_TO_UDINT
BOOL_TO_UINT	BYTE_TO_UINT	DATE_TO_UINT	DINT_TO_UINT
BOOL_TO_USINT	BYTE_TO_USINT	DATE_TO_USINT	DINT_TO_USINT
BOOL_TO_WORD	BYTE_TO_WORD	DATE_TO_WORD	DINT_TO_WORD
DT_TO_<TYPE>	DWORD_TO_<TYPE>	INT_TO_<TYPE>	WORD_TO_<TYPE>
DT_TO_BOOL	DWORD_TO_BOOL	INT_TO_BOOL	WORD_TO_BOOL
DT_TO_BYTE	DWORD_TO_BYTE	INT_TO_BYTE	WORD_TO_BYTE
DT_TO_DATE	DWORD_TO_DATE	INT_TO_DATE	WORD_TO_DATE
DT_TO_DINT	DWORD_TO_DINT	INT_TO_DINT	WORD_TO_DINT
DT_TO_DWORD	DWORD_TO_DT	INT_TO_DT	WORD_TO_DT
DT_TO_INT	DWORD_TO_INT	INT_TO_DWORD	WORD_TO_DWORD
DT_TO_REAL	DWORD_TO_REAL	INT_TO_REAL	WORD_TO_INT
DT_TO_SINT	DWORD_TO_SINT	INT_TO_SINT	WORD_TO_REAL
DT_TO_STRING	DWORD_TO_STRING	INT_TO_STRING	WORD_TO_SINT
DT_TO_TIME	DWORD_TO_TIME	INT_TO_TIME	WORD_TO_STRING
DT_TO_TOD	DWORD_TO_TOD	INT_TO_TOD	WORD_TO_TIME
DT_TO_UDINT	DWORD_TO_UDINT	INT_TO_UDINT	WORD_TO_TOD
DT_TO_UINT	DWORD_TO_UINT	INT_TO_UINT	WORD_TO_UDINT
DT_TO_USINT	DWORD_TO_USINT	INT_TO_USINT	WORD_TO_UINT
DT_TO_WORD	DWORD_TO_WORD	INT_TO_WORD	WORD_TO_USINT
REAL_TO_<TYPE>	SINT_TO_<TYPE>	STRING_TO_<TYPE>	TIME_TO_<TYPE>
REAL_TO_BOOL	SINT_TO_BOOL	STRING_TO_BOOL	TIME_TO_BOOL
REAL_TO_BYTE	SINT_TO_BYTE	STRING_TO_BYTE	TIME_TO_BYTE
REAL_TO_DATE	SINT_TO_DATE	STRING_TO_DATE	TIME_TO_DATE
REAL_TO_DINT	SINT_TO_DINT	STRING_TO_DINT	TIME_TO_DINT
REAL_TO_DT	SINT_TO_DT	STRING_TO_DT	TIME_TO_DT
REAL_TO_DWORD	SINT_TO_DWORD	STRING_TO_DWORD	TIME_TO_DWORD
REAL_TO_INT	SINT_TO_INT	STRING_TO_INT	TIME_TO_INT
REAL_TO_SINT	SINT_TO_REAL	STRING_TO_REAL	TIME_TO_REAL
REAL_TO_STRING	SINT_TO_STRING	STRING_TO_SINT	TIME_TO_SINT
REAL_TO_TIME	SINT_TO_TIME	STRING_TO_TIME	TIME_TO_STRING
REAL_TO_TOD	SINT_TO_TOD	STRING_TO_TOD	TIME_TO_TOD
REAL_TO_UDINT	SINT_TO_UDINT	STRING_TO_UDINT	TIME_TO_UDINT
REAL_TO_UINT	SINT_TO_UINT	STRING_TO_UINT	TIME_TO_UINT
REAL_TO_USINT	SINT_TO_USINT	STRING_TO_USINT	TIME_TO_USINT
REAL_TO_WORD	SINT_TO_WORD	STRING_TO_WORD	TIME_TO_WORD
TOD_TO_<TYPE>	UDINT_TO_<TYPE>	UINT_TO_<TYPE>	USINT_TO_<TYPE>

TOD_TO_BOOL	UDINT_TO_BOOL	UINT_TO_BOOL	USINT_TO_BOOL
TOD_TO_BYTE	UDINT_TO_BYTE	UINT_TO_BYTE	USINT_TO_BYTE
TOD_TO_DATE	UDINT_TO_DATE	UINT_TO_DATE	USINT_TO_DATE
TOD_TO_DINT	UDINT_TO_DINT	UINT_TO_DINT	USINT_TO_DINT
TOD_TO_DT	UDINT_TO_DT	UINT_TO_DT	USINT_TO_DT
TOD_TO_DWORD	UDINT_TO_DWORD	UINT_TO_DWORD	USINT_TO_DWORD
TOD_TO_INT	UDINT_TO_INT	UINT_TO_INT	USINT_TO_INT
TOD_TO_REAL	UDINT_TO_REAL	UINT_TO_REAL	USINT_TO_REAL
TOD_TO_SINT	UDINT_TO_SINT	UINT_TO_SINT	USINT_TO_SINT
TOD_TO_STRING	UDINT_TO_STRING	UINT_TO_STRING	USINT_TO_STRING
TOD_TO_TIME	UDINT_TO_TIME	UINT_TO_TIME	USINT_TO_TIME
TOD_TO_UDINT	UDINT_TO_TOD	UINT_TO_TOD	USINT_TO_TOD
TOD_TO_UINT	UDINT_TO_UINT	UINT_TO_UDINT	USINT_TO_UDINT
TOD_TO_USINT	UDINT_TO_USINT	UINT_TO_USINT	USINT_TO_UINT
TOD_TO_WORD	UDINT_TO_WORD	UINT_TO_WORD	USINT_TO_WORD

Table 4-7-1

4.7.1 BOOL_TO_<TYPE>—Bool Type Conversion

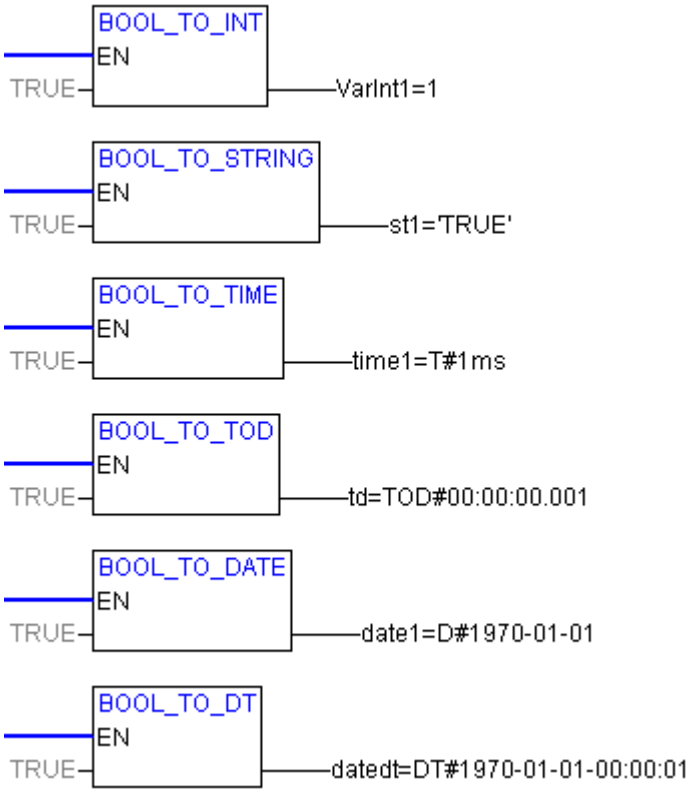
- Function: Convert BOOL into other data types.
- Input/output data type (see table 4-7-1):
- When output is a NUMERIC type, if input is TRUE, then output is 1, and if input is FALSE, then output is 0;
- When output is STRING type, if input is TRUE, then output is 'TRUE', if input is FALSE, then output is 'FALSE'.

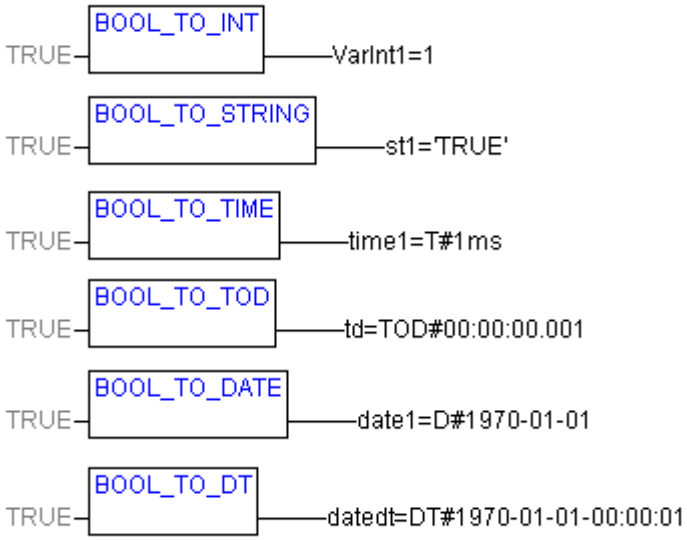
APPLICATION EXAMPLE

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	Name	Address	Type	Initial value	Comment
0001	VarInt1		INT		
0002	st1		STRING		
0003	time1		TIME		
0004	td		TOD		
0005	date1		DATE		
0006	datedt		DT		

Programming Language	Program (partial)
Ladder Diagram (LD)	

Structured Text (ST)	VarInt1:=BOOL_TO_INT(TRUE); (*result is 1*) st1:=BOOL_TO_STRING(TRUE); (*result is 'TRUE'*) time1:=BOOL_TO_TIME(TRUE); (*result is T#1ms*) td:=BOOL_TO_TOD(TRUE); (*result is TOD#00:00:00.001*) date1:=BOOL_TO_DATE(TRUE); (*result is D#1970-01-01*) datedt :=BOOL_TO_DT(TRUE); (*result is DT#1970-01-01-00:00:01*)
Instruction List (IL)	LD TRUE BOOL_TO_INT ST VarInt1 (*result is 1*) LD TRUE BOOL_TO_STRING ST st1 (*result is 'TRUE'*) LD TRUE BOOL_TO_TIME ST time1 (*result is T#1ms*) LD TRUE BOOL_TO_TOD ST td (*result is TOD#00:00:00.001*) LD TRUE BOOL_TO_DATE ST date1 (*result is D#1970-01-01*) LD TRUE BOOL_TO_DT ST datedt (*result is DT#1970-01-01-00:00:01*)
Function Block Diagram (FBD)	

4.7.2 BYTE_TO_<TYPE>—Byte Data Conversion

- Function: Convert BYTE into other data types.
- Input/output data type (see table 4-7-1):
- When BYTE_TO_BOOL, if input is not 0, then output is TRUE, and if input is 0， then output is FALSE;
- When BYTE_TO_TIME, BYTE_TO_TOD, input will convert in millisecond value.
- When BYTE_TO_DATE, BYTE_TO_DT, input will convert in second value.

APPLICATION EXAMPLE

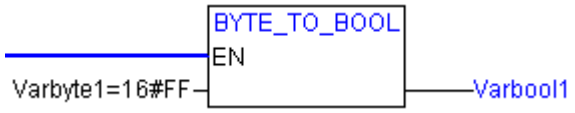
Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

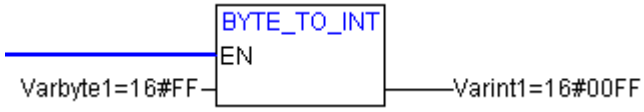
	Name	Address	Type	Initial value	Comment
0001	Varbool1		BOOL		
0002	Varbyte1		BYTE		
0003	Varint1		INT		
0004	Vartime1		TIME		
0005	Vardt1		DT		
0006	Varreal1		REAL		
0007	Varstring1		STRING		

Programming Language: Ladder Diagram (LD)

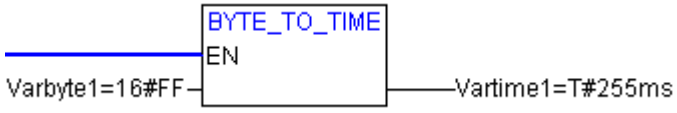
Program (partial)



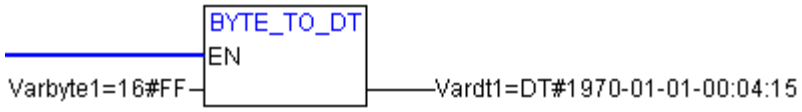
(*result is TRUE *)



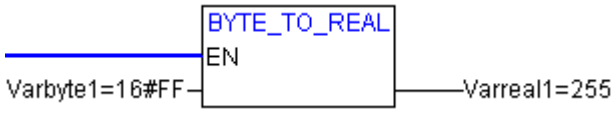
(*result is 16# 00FF *)



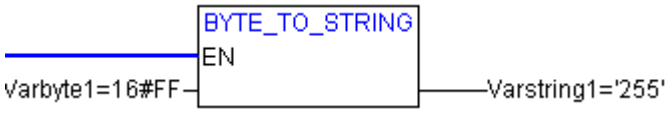
(*result is T#255ms *)



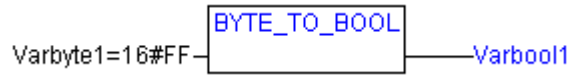
(*result is DT#1977-01-01-00:04:15 *)

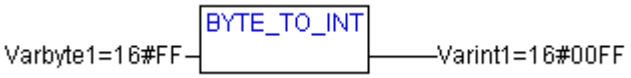


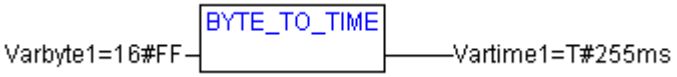
(*result is 255 *)

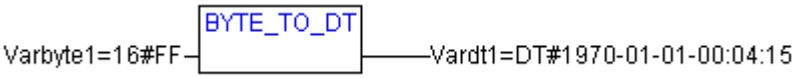


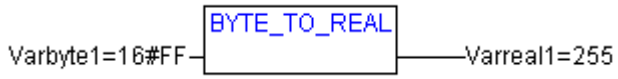
(*result is 字符串'255'*)

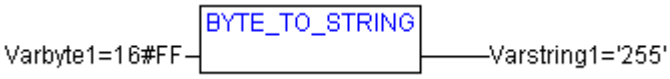
<table>
<tr><td>Structured Text (ST)</td><td>Varbyte1:=16#FF (*Varbyte1 *)
Varbool1:=BYTE_TO_BOOL(Varbyte1); (*result is TRUE *)
Varint1:=BYTE_TO_INT(Varbyte1); (*result is 16# FF *)
Vartime1:=BYTE_TO_TIME(Varbyte1); (*result is T#255ms *)
Vardt1:=BYTE_TO_DT(Varbyte1);
(*result is DT#1977-01-01-00:04:05 *)
Varreal1:=BYTE_TO_REAL(Varbyte1); (*result is 255 *)
Varstring1:=BYTE_TO_STRING(Varbyte1); (*result is string '255'*)</td></tr>
<tr><td>Instruction List (IL)</td><td>LD 16#FF
ST Varbyte1 (*Varbyte1 *)
LD Varbyte1
BYTE_TO_BOOL
ST Varbool1 (*result is TRUE *)
LD Varbyte1
BYTE_TO_INT
ST Varint1 (*result is 16# FF *)
LD Varbyte1
BYTE_TO_TIME
ST Vartime1 (*result is T#255ms *)
LD Varbyte1
BYTE_TO_DT
ST Vardt1 (*result is DT#1970-01-01-00:04:05 *)
LD Varbyte1
BYTE_TO_REAL
ST Varreal1 (*result is 255 *)
LD Varbyte1
BYTE_TO_STRING
ST Varstring1 (*result is string '255'*)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td> 

(*result is TRUE *)
 

(*result is 16# 00FF *)
 

(*result is T#255ms *)
 

(*result is DT#1977-01-01-00:04:05 *)
 

(*result is 255 *)
 

(*result is string '255'*)</td></tr>
</table>

4.7.3 WORD_TO_<TYPE>—Word Data Conversion

- Function: Convert WORD into other data types.
- Input/output data type (see table 4-7-1):
- When WORD_TO_BOOL, if input is not 0, then output is TRUE, and if input is 0, then output is FALSE;
- When WORD_TO_TIME, WORD_TO_TOD, input will convert in millisecond value.
- When WORD_TO_DATE, WORD_TO_DT, input will convert in second value.

APPLICATION EXAMPLE

Variable Declaration

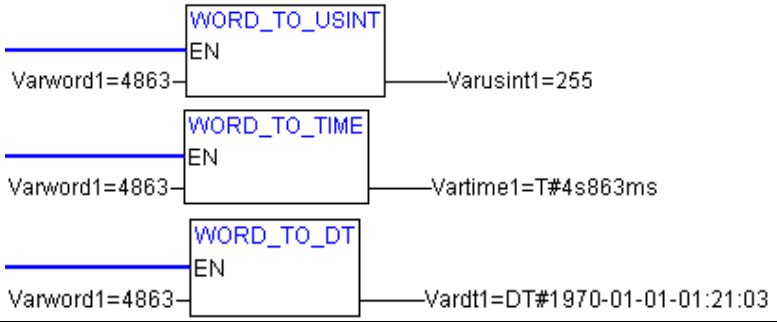
VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	Name	Address	Type	Initial value	Comment
0001	Varusint1		USINT		
0002	Varword1		WORD		
0003	Vartime1		TIME		
0004	Vardt1		DT		

Programming Language

Program (partial)

Ladder Diagram (LD)



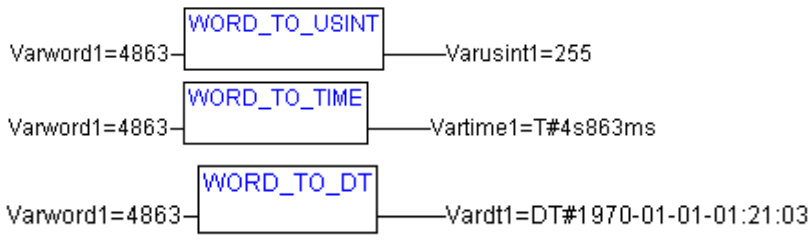
Structured Text (ST)

```
Varword1:=4863;                    (*Varword1 *)
Varusint1:=WORD_TO_USINT(Varword1);    (*result 255 *)
Comments: If INT 4863(16#12FF) is stored as USINT variable, the high bit data is lost, only display low bit 255(16#FF).
Vartime1:=WORD_TO_TIME(Varword1);    (*result T#4s863ms*)
Vardt1:=WORD_TO_DT(Varword1);
(*result DT#1970-01-01-01:21:03 *)
```

Instruction List (IL)

```
LD   4863
ST   Varword1              (*Varword1*)
LD   Varword1
WORD_TO_USINT
ST   Varusint1             (*result 255 *)
LD   Varword1
WORD_TO_TIME
ST   Vartime1              (*result T#4s863ms*)
LD   Varword1
WORD_TO_DT
ST   Vardt1        (*result DT#1970-01-01-01:21:03 *)
```

Function Block Diagram (FBD)



4.7.4 DWORD_TO_<TYPE>—DWORD Type Conversion

- Function: Convert DWORD into other data types.
- Input/output data type (see table 4-7-1):
- When DWORD_TO_BOOL, if input is not 0, then output is TRUE, and if input is 0, then output is FALSE;
- When DWORD_TO_TIME, DWORD_TO_TOD, input will convert in ms.
- When DWORD_TO_DATE, DWORD_TO_DT, input will convert in s.

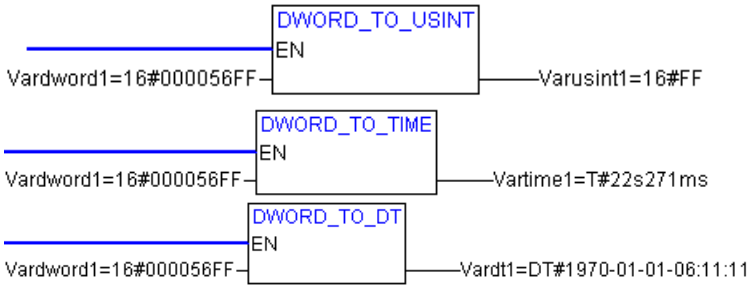
APPLICATION EXAMPLE

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	Name	Address	Type	Initial value	Comment
0001	Varusint1		USINT		
0002	Vardword1		DWORD		
0003	Vartime1		TIME		
0004	Vardt1		DT		

Programming Language: Ladder Diagram (LD)



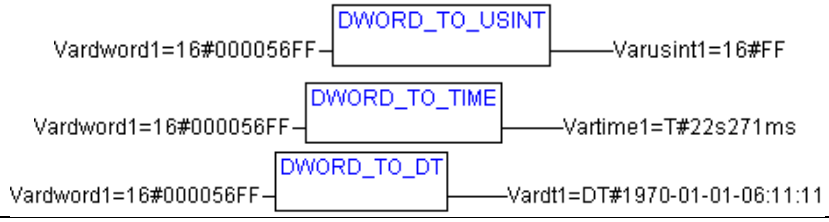
Structured Text (ST)

```
Varword1:=16#56FF;                    (*Varword1*)
Varusint1:=DWORD_TO_USINT(Vardword1);      (*result 255 *)
Comments: If INT 4863(16#12FF) is stored as USINT variable, the high bit data is lost, only display low bit 255(16#FF).
Vartime1:=DWORD_TO_TIME(Vardword1);   (*result T#22s271ms*)
Vardt1:=DWORD_TO_DT(Vardword1);
(*result DT#1970-01-01-06:11:11 *)
```

Instruction List (IL)

```
LD   16#56FF
ST   Vardword1              (*Vardword1*)
LD   Vardword1
DWORD_TO_USINT
ST   Varusint1                (*result 255 *)
LD   Vardword1
DWORD_TO_TIME
ST   Vartime1               (*result T#22s271ms*)
LD   Vardword1
DWORD_TO_DT
ST   Vardt1          (*result DT#1970-01-01-06:11:11 *)
```

Function Block Diagram (FBD)



4.7.5 SINT_TO_<TYPE>—Short Integer Data Conversion

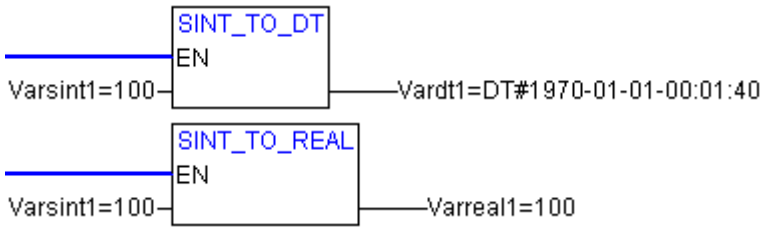
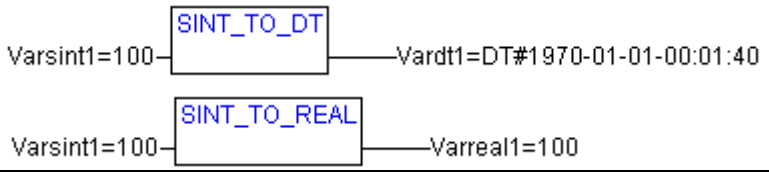
- Function: Convert SINT into other data types.
- Input/output data type (see table 4-7-1):
- When SINT_TO_BOOL, if input is not 0, then output is TRUE, and if input is 0, then output is FALSE;
- When SINT_TO_TIME, SINT_TO_TOD, input will convert in millisecond value.
- When SINT_TO_DATE, SINT_TO_DT, input will convert in second value.

APPLICATION EXAMPLE

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	Name	Address	Type	Initial value	Comment
0001	Varsint1		SINT		
0002	Vardt1		DT		
0003	Varreal1		REAL		

Programming Language	Program (partial)
Ladder Diagram (LD)	
Structured Text (ST)	Varsint1:=100; (*Varsint1*) Vardt1:=SINT_TO_DT(Varsint1); (*result DT#1970-01-01-00:01:40 *) Varreal1:=SINT_TO_REAL(Varsint1); (*result is 100 .0*)
Instruction List (IL)	LD 100 ST Varsint1 (*Varsint1*) LD Varsint1 SINT_TO_DT ST Vardt1 (*result DT#1970-01-01-00:01:40 *) LD Varsint1 SINT_TO_REAL ST Varreal1 (*result Varrea1 is 100.0*)
Function Block Diagram (FBD)	

4.7.6 USINT_TO_<TYPE>—USINT Conversion

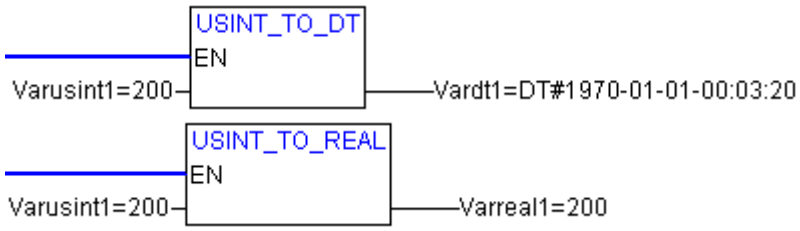
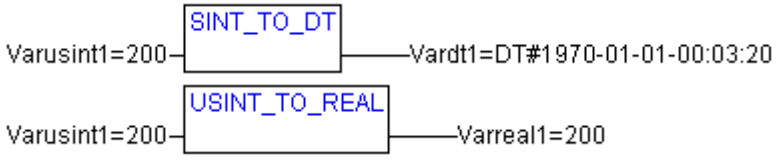
- Function: Convert USINT into other data types.
- Input/output data type (see table 4-7-1):
- When USINT_TO_BOOL, if input is not 0, then output is TRUE, and if input is 0, then output is FALSE;
- When USINT_TO_TIME, USINT_TO_TOD, input will convert in millisecond value.
- When USINT_TO_DATE, USINT_TO_DT, input will convert in second value.

APPLICATION EXAMPLE

Variable Declaration

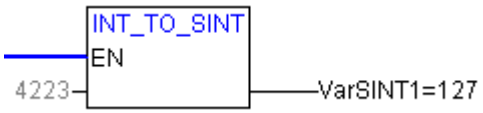
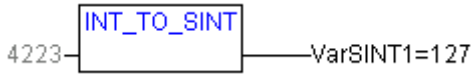
VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	Name	Address	Type	Initial value	Comment
0001	Varusint1		USINT		
0002	Vardt1		DT		
0003	Varreal1		REAL		

Programming Language	Program (partial)
Ladder Diagram (LD)	
Structured Text (ST)	Varusint1:=200; (*Varusint1*) Vardt1:=USINT_TO_DT(Varusint1);(*result DT#1970-01-01-00:03:20 *) Varreal1:=USINT_TO_REAL(Varusint1); (*result is 200 .0*)
Instruction List (IL)	LD 200 ST Varusint1 (*Varusint1*) LD Varusint1 USINT_TO_DT ST Vardt1 (*result DT#1970-01-01-00:03:20 *) LD Varusint1 USINT_TO_REAL ST Varreal1 (*result Varreal1 is 200.0*)
Function Block Diagram (FBD)	

4.7.7 INT_TO_<TYPE>—INT Conversion

- Function: Convert INT into other data types.
- Input/output data type (see table 4-7-1):
- When INT_TO_BOOL, if input is not 0, then output is TRUE, and if input is 0, then output is FALSE;
- When INT_TO_TIME, INT_TO_TOD, input will convert in millisecond value.
- When INT_TO_DATE, INT_TO_DT, input will convert in second value.

<table>
<tr><td colspan="2">APPLICATION EXAMPLE</td></tr>
<tr><td colspan="2">Variable Declaration</td></tr>
<tr><td colspan="2">VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONS
<table>
<tr><th></th><th>Name</th><th>Address</th><th>Type</th><th>Initial value</th><th>Comment</th></tr>
<tr><td>0001</td><td>VarSINT1</td><td></td><td>SINT</td><td></td><td></td></tr>
<tr><td>0002</td><td>VarREAL1</td><td></td><td>REAL</td><td></td><td></td></tr>
</table></td></tr>
<tr><td>Programming Language</td><td>Program (partial)</td></tr>
<tr><td>Ladder Diagram (LD)</td><td> 
</td></tr>
<tr><td>Structured Text (ST)</td><td>VarSINT1 := INT_TO_SINT(4223); (*result VarSINT1 is 127 *)
Comments: If INT 4223 (16#107F) is stored as SINT variable, the high bit data is lost, only displaylow bit 127 (16#7F).</td></tr>
<tr><td>Instruction List (IL)</td><td>LD 2
INT_TO_REAL
ST VarREAL1 (*result VarREAL1 is 2.0*)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td> 
</td></tr>
</table>

4.7.8 UINT_TO_<TYPE>—UINT Conversion

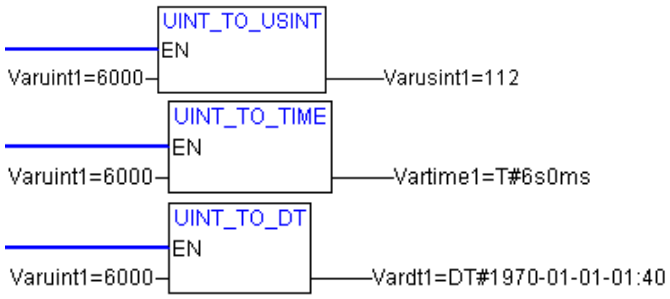
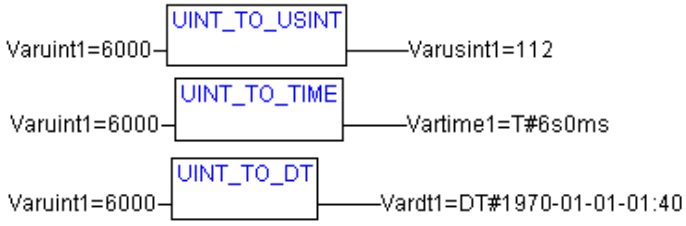
- Function: Convert UINT to other data types.
- Input/output data type (see table 4-7-1);
- When UINT_TO_BOOL, if input is not 0, then output is TRUE, and if input is 0, then output is FALSE;
- When UINT_TO_TIME, UINT_TO_TOD, input will convert in millisecond value.
- When UINT_TO_DATE, UINT_TO_DT, input will convert in second value.

APPLICATION EXAMPLE

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	Name	Address	Type	Initial value	Comment
0001	Varuint1		UINT		
0002	Varusint1		USINT		
0003	Vartime1		TIME		
0004	Vardt1		DT		

Programming Language	Program (partial)
Ladder Diagram (LD)	
Structured Text (ST)	Varuint1:=6000; Varusint1:=UINT_TO_USINT(Varuint1); Comments: If INT 6000 (16#1770) is stored as SINT variable, the high bit data is lost, only displaylow bit 112 (16#70). Vartime1:=UINT_TO_TIME(Varuint1); Vardt1:=UINT_TO_DT(Varuint1);
Instruction List (IL)	LD 6000 ST Varuint1 (*Varuint1 *) LD Varuint1 UINT_TO_USINT ST Varusint1 (*result 112*) LD Varuint1 UINT_TO_TIME ST Vartime1 (*result T#6s0ms*) LD Varuint1 UINT_TO_DT ST Vardt1 (*result DT#1970-01-01-01:40:00 *)
Function Block Diagram (FBD)	

4.7.9 DINT_TO_<TYPE>—DINT Conversion

- Function: Convert DINT into other data types.
- Input/output data type (see table 4-7-1):
- When DINT_TO_BOOL, if input is not 0, then output is TRUE, and if input is 0, then output is FALSE;
- When DINT_TO_TIME, DINT_TO_TOD, input will convert in millisecond value.
- When DINT_TO_DATE, DINT_TO_DT, input will convert in second value.

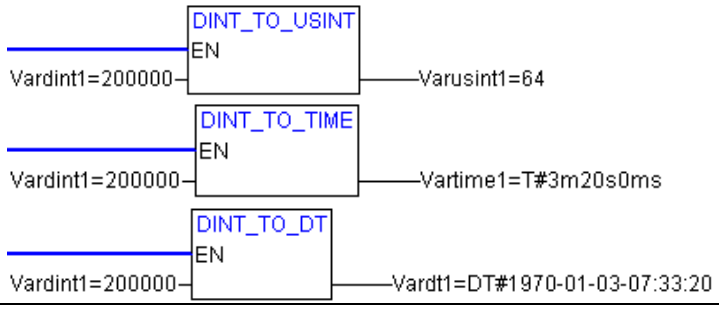
APPLICATION EXAMPLE

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	Name	Address	Type	Initial value	Comment
0001	Vardint1		DINT		
0002	Varusint1		USINT		
0003	Vartime1		TIME		
0004	Vardt1		DT		

Programming Language: Ladder Diagram (LD)



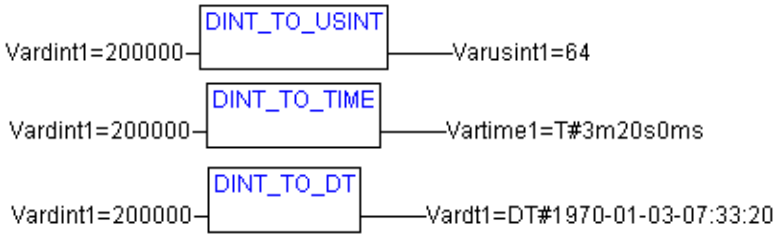
Programming Language: Structured Text (ST)

```
Vardint1:=200000;
Varusint1:=DINT_TO_USINT(Vardint1);       (*result 64*)
Comments: If INT 200000 (16#30D40) is stored as USINT variable, the high bit data is lost, only displaylow bit 64 (16#40).
Vartime1:=DINT_TO_TIME(Vardint1);  (*result T#3m20s0ms*)
Vardt1:=DINT_TO_DT(Vardint1);     (*result DT#1970-01-03-07:33:20 *)
```

Programming Language: Instruction List (IL)

```
        LD   200000
ST   Vardint1              (*Vardint1*)
LD                 Vardint1
        DINT_TO_USINT
        ST                 Varusint1          (*result 64*)
        LD                 Vardint1
        DINT_TO_TIME
        ST                 Vartime1          (*result T#3m20s0ms*)
        LD                 Vardint1
        DINT_TO_DT
        ST                 Vardt1       (*result DT#1970-01-03-07:33:20 *)
```

Programming Language: Function Block Diagram (FBD)



4.7.10 UDINT_TO_<TYPE>—UDINT Conversion

- Function: Convert UDINT to other data types.
- Input/output data type (see table 4-7-1):
- When UDINT_TO_BOOL, if input is not 0, then output is TRUE, and if input is 0, then output is FALSE;
- When UDINT_TO_TIME, UDINT_TO_TOD, input will convert in millisecond value.
- When UDINT_TO_DATE, UDINT_TO_DT, input will convert in second value.

APPLICATION EXAMPLE

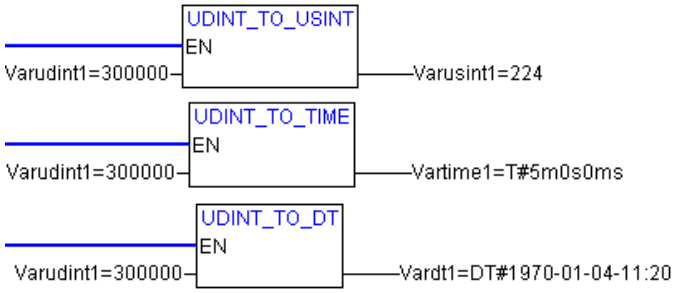
Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	Name	Address	Type	Initial value	Comment
0001	Vardint1		DINT		
0002	Varusint1		USINT		
0003	Vartime1		TIME		
0004	Vardt1		DT		

Programming Language | **Program (partial)**

Ladder Diagram (LD)



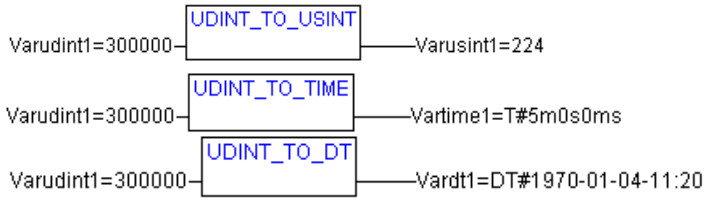
Structured Text (ST)

```
Varudint1:=300000;
Varusint1:= UDINT_TO_USINT(Varudint1);     (*result 224*)
Comments: If INT 300000 (16#493E0) is stored as USINT variable, the high bit data is
lost, only displaylow bit 224 (16#E0).
Vartime1:=UDINT_TO_TIME(Varudint1);       (*result T#5m0s0ms*)
Vardt1:=UDINT_TO_DT(Varudint1);
(*result DT#1970-01-04-11:20:00 *)
```

Instruction List (IL)

```
LD          300000
ST          Varudint1          (*Varudint1*)
LD          Varudint1
UDINT_TO_USINT
ST          Varusint1          (*result 224*)
LD          Varudint1
UDINT_TO_TIME
ST          Vartime1           (*result T#5m0s0ms*)
LD          Varudint1
UDINT_TO_DT
ST          Vardt1      (*result DT#1970-01-04-11:20:00 *)
```

Function Block Diagram (FBD)



4.7.11 REAL_TO_<TYPE>—REAL Conversion

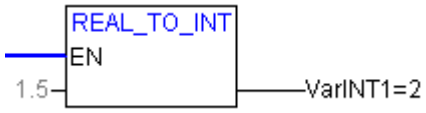
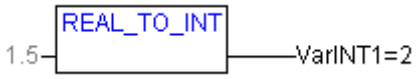
- Function: Convert REAL into other data types. When converting REAL to other data types, it will first convert the value to a nearest integer and then convert to the new type.
- Input/output data type (see table 4-7-1):
- When REAL_TO_BOOL, if input is not 0, then output is TRUE, and if input is 0, then output is FALSE;
- When REAL_TO_TIME, REAL_TO_TOD, input will convert in millisecond value.
- When REAL_TO_DATE, REAL_TO_DT, input will convert in second value.

APPLICATION EXAMPLE

Variable Declaration

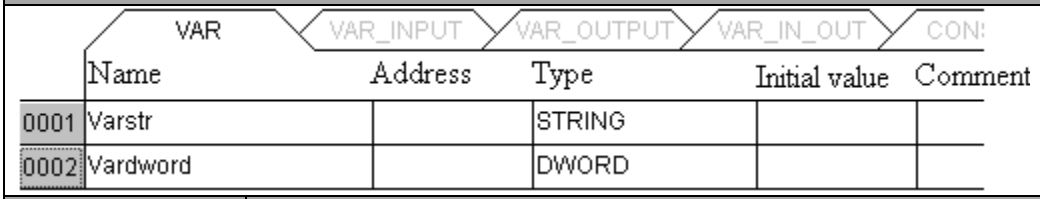
VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

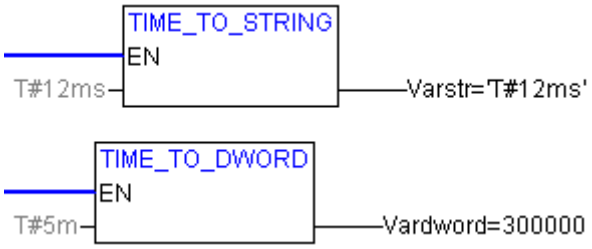
	Name	Address	Type	Initial value	Comment
0001	Varint1		INT		
0002	Varint2		INT		
0003	Varint3		INT		
0004	Varint4		INT		

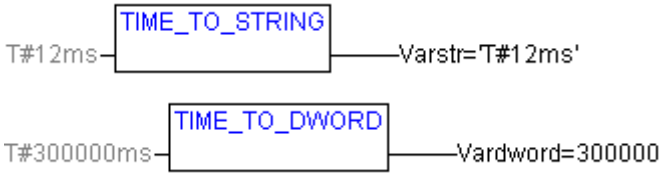
Programming Language	Program (partial)
Ladder Diagram (LD)	
Structured Text (ST)	VarINT1:= REAL_TO_INT(1.5); (*result VarINT1 is 2*) VarINT2:= REAL_TO_INT(1.4); (*result VarINT2 is 1*) VarINT3:= REAL_TO_INT(-1.5); (*result VarINT3 is -2*) VarINT4:= REAL_TO_INT(-1.4); (*result VarINT4 is -1*)
Instruction List (IL)	LD 2.7 REAL_TO_INT ST VarINT1 (*result VarINT1 is 3*)
Function Block Diagram (FBD)	

4.7.12 TIME_TO_<TYPE>—Time Type Conversion

- Function: Convert TIME into other data types. Inside the PowerPro, time is stored into DWORD in millisecond (for TIME_OF_DAY, start from 00:: 00)
- Input/output data type (see table 4-7-1):
- When TIME_TO_BOOL, if input is not 0, then output is TRUE, and if input is 0, then output is FALSE.

<table>
<tr><td colspan="2">APPLICATION EXAMPLE</td></tr>
<tr><td colspan="2">Variable Declaration</td></tr>
<tr><td colspan="2">


</td></tr>
<tr><td>Programming Language</td><td>Program (partial)</td></tr>
<tr><td>Ladder Diagram (LD)</td><td>


</td></tr>
<tr><td>Structured Text (ST)</td><td>Varstr:=TIME_TO_STRING(T#12ms); (*result is‘T#12ms’*)
Vardword:=TIME_TO_DWORD(T#5m); (*result is 300000*)</td></tr>
<tr><td>Instruction List (IL)</td><td>LD T#12ms
TIME_TO_STRING
ST Varstr (*result is‘T#12ms’*)
LD T#300000ms
TIME_TO_DWORD
ST Vardword (*result is 300000*)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td>


</td></tr>
</table>

4.7.13 DATE_TO_<TYPE>—DATE Type Conversion

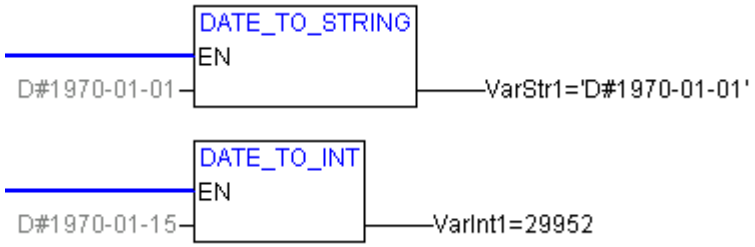
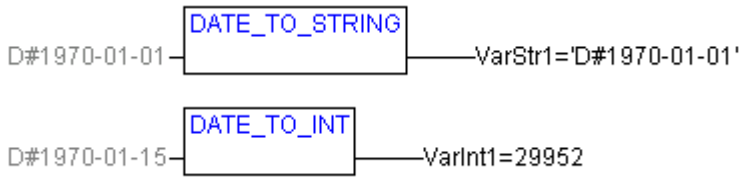
- Function: Convert DATE into other data types, date is stored in second inside PowerPro form January 1, 1970.
- Input/output data type (see table 4-7-1):
- When DATE_TO_BOOL, if input is not 0, then output is TRUE, and if input is 0, then output is FALSE.

APPLICATION EXAMPLE

Variable Declaration

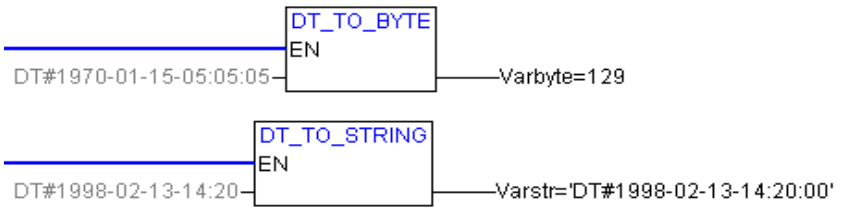
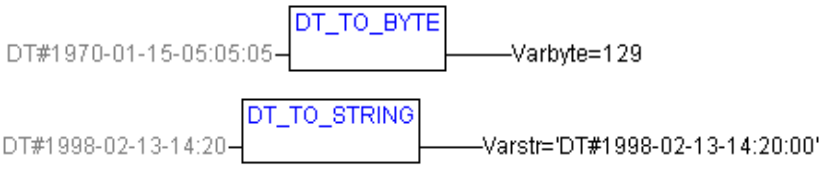
VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONS

	Name	Address	Type	Initial value	Comment
0001	Varint1		INT		
0002	VarStr1		STRING		

Programming Language	Program (partial)
Ladder Diagram (LD)	
Structured Text (ST)	VarStr1:=DATE_TO_STRING(D#1970-01-01); (*result is 'D#1970-01-01' *) VarInt1:=DATE_TO_INT(D#1970-01-15); (*result is 29952*) Comments: If D#1970-01-15 (16#493E0) is stored as INT variable, the high bit data is lost, only display low bit (16#7500, #29952).
Instruction List (IL)	LD D#1970-01-01 DATE_TO_STRING ST VarStr1 (*result is'TRUE'*) LD D#1970-01-15 DATE_TO_INT ST VarInt1 (*result is 29952*)
Function Block Diagram (FBD)	

4.7.14 DT_TO_<TYPE>—DT Type Conversion

- Function: Convert DT into other data types, date is stored in second inside PowerPro form January 1, 1970.
- Input/output data type (see table 4-7-1):
- When DT_TO_BOOL, if input is not 0, then output is TRUE, and if input is 0, then output is FALSE.

<table>
<tr><td colspan="2">APPLICATION EXAMPLE</td></tr>
<tr><td colspan="2">Variable Declaration</td></tr>
<tr><td colspan="2">
VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CON
<table>
<tr><th></th><th>Name</th><th>Address</th><th>Type</th><th>Initial value</th><th>Comment</th></tr>
<tr><td>0001</td><td>Varbyte</td><td></td><td>BYTE</td><td></td><td></td></tr>
<tr><td>0002</td><td>VarStr</td><td></td><td>STRING</td><td></td><td></td></tr>
</table>
</td></tr>
<tr><td>Programming Language</td><td>Program (partial)</td></tr>
<tr><td>Ladder Diagram (LD)</td><td> 
</td></tr>
<tr><td>Structured Text (ST)</td><td>Varbyte:=DT_TO_BYTE(DT#1970-01-15-05:05:05);
(*result Varbyte is 129*)
Comments: convert DT#1970-01-15-05:05:05 to sec, the value is (((14*24+5)*60+5)*60+5)=1227905=16#12BC81, stored as a BYTE variable, then the high 24 bits is lost, only display low 8 bits,6#81= 129.
Varstr:=DT_TO_STRING(DT#1998-02-13-14:20);
(*result Varstr is ‘DT#1998-02-13-14:20’)</td></tr>
<tr><td>Instruction List (IL)</td><td>LD DT#1970-01-15-05:05:05
DT_TO_BYTE
ST Varbyte (*result Varbyte is 129*)
LD DT#1998-02-13-14:20
DT_TO_STRING
ST Varstr (*result Varstr is‘DT#1998-02-13-14:20’)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td> 
</td></tr>
</table>

4.7.15 TOD_TO_<TYPE>—TOD Type Conversion

- Function: Convert TOD into other data types, date is converted inside in second value.
- Input/output data type (see table 4-7-1):
- When TOD_TO_BOOL, if input is not 0, then output is TRUE, and if input is 0, then output is FALSE.

APPLICATION EXAMPLE

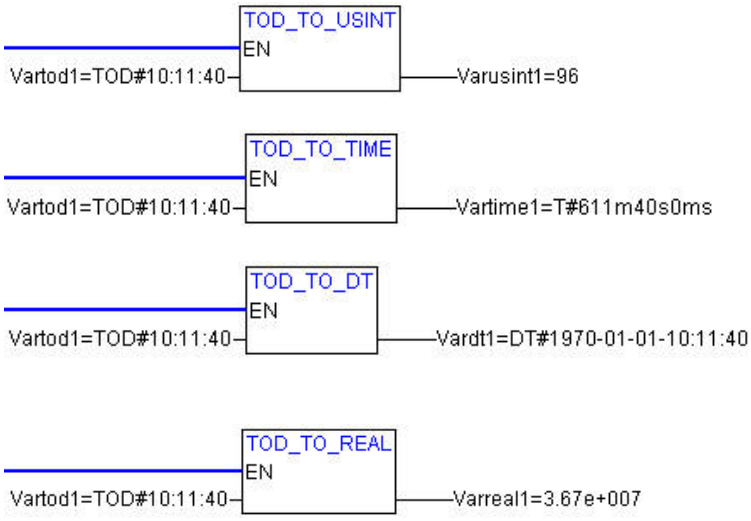
Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	Name	Address	Type	Initial value	Comment
0001	Vartod1		TOD		
0002	varbool1		BOOL		
0003	Varusint1		USINT		
0004	Vartime1		TIME		
0005	Vardt1		DT		
0006	Varreal1		REAL		

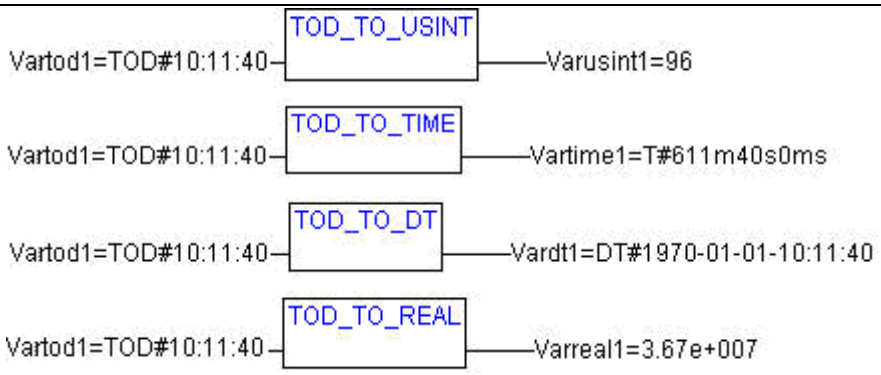
Programming Language: Program (partial)

Ladder Diagram (LD)



Structured Text (ST)

```
Vartod1:=TOD#10:11:40;              (*Vartod1*)
Varusint1:=TOD_TO_USINT(Vartod1);       (*result is 96*)
Vartime1:=TOD_TO_TIME(Vartod1);  (*result is T#611m40s0ms*)
Vardt1:=TOD_TO_DT(Vartod1);
(*result  is DT#1970-01-01-10:11:40*)
Varreal1:=TOD_TO_REAL(Vartod1);     (*result is 3.67e+007*)
```

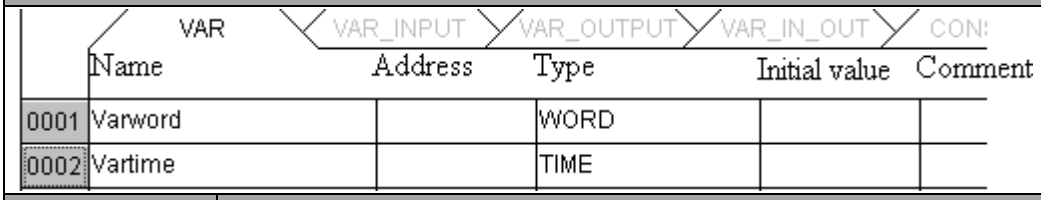
Instruction List (IL)	LD TOD#10:11:40 ST Vartod1 (*Vartod1*) LD Vartod1 TOD_TO_USINT ST Varusint1 (*result is 96*) LD Vartod1 TOD_TO_TIME ST Vartime1 (*result is T#611m40s0ms*) LD Vartod1 TOD_TO_DT ST Vardt1 (*result is DT#1970-01-01-10:11:40*) LD Vartod1 TOD_TO_REAL ST Varreal1 (*result is 3.67e+007*)
Function Block Diagram (FBD)	

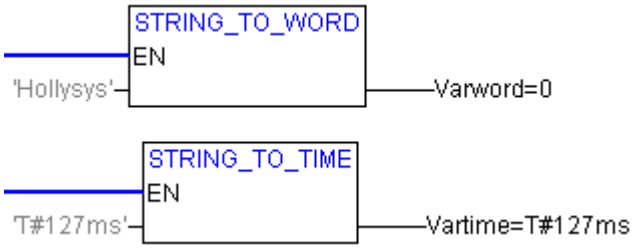
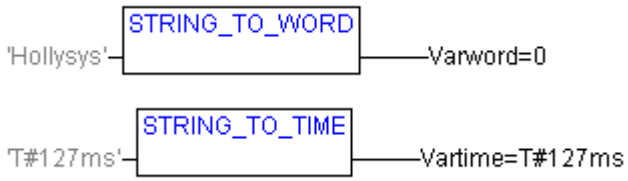
4.7.16 STRING_TO_<TYPE>—STRING Type Conversion

- Function: Convert STRING into other data types. STRING variables must contain a valid target variable value, otherwise the result is 0.
- Input/output data type (see table 4-7-1)

APPLICATION EXAMPLE

Variable Declaration



Programming Language	Program (partial)
Ladder Diagram (LD)	
Structured Text (ST)	Varword:=STRING_TO_WORD('Hollysys'); (*result is 0*) Vartime:=STRING_TO_TIME('T#127ms'); (*result is T#127ms*)
Instruction List (IL)	LD 'Hollysys' STRING_TO_WORD ST Varword (*result Varword is 0*) LD 'T#127ms' STRING_TO_TIME ST Vartime (*result Vartime is T#127ms*)
Function Block Diagram (FBD)	

4.7.17 TRUNC—Truncation

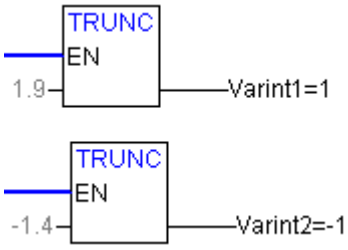
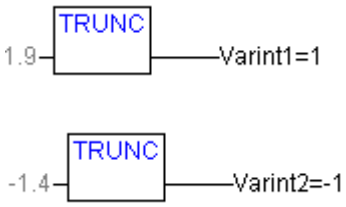
- Function: Truncate the decimal part of value to keep the integral part.
- Input/output data type: Input is REAL; output is INT, WORD, and DWORD.

APPLICATION EXAMPLE

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONS

	Name	Address	Type	Initial value	Comment
0001	Varint1		INT		
0002	Varint2		INT		

Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	Varint1:=TRUNC(1.9); (*result Varint1 is 1*) Varint2:=TRUNC(-1.4); (*result Varint2 is -1*)
Instruction List (IL)	LD 1.9 TRUNC ST Varint1 (*result Varint1 is 1*) LD -1.4 TRUNC ST Varint2 (*result Varint2 is -1*)
Function Block Diagram (FBD)	

Note

- Information may be lost when converting from a larger data type to a smaller data type.
- This instruction is used just to get the integral part of the variable. In case of converting round to a nearest integer, REAL_TO_INT instruction shall be used.

4.8 ELEMENTARY MATHEMATICAL INSTRUCTIONS

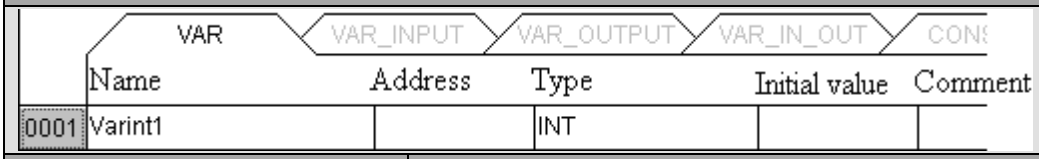
4.8.1 ABS—Absolute Value

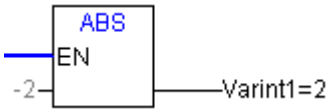
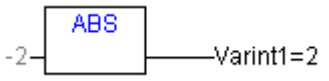
- Function: Assign the absolute value of input to output.
- Input/output data type: See the table below.

Input Data Type	Output Data Type
INT	INT, WORD, DWORD, DINT, UINT, REAL
REAL	REAL
BYTE	INT, BYTE, WORD, DWORD, DINT, UINT, REAL
WORD	WORD, DWORD, DINT, REAL
DWORD	DWORD, DINT, REAL
SINT	INT, BYTE, WORD, DWORD, DINT, UINT, REAL
USINT	INT, BYTE, WORD, DWORD, DINT, UINT, REAL
UINT	WORD, DWORD, DINT, UINT, REAL
DINT	DWORD, DINT, REAL
UDINT	DWORD, DINT, UDINT, REAL

APPLICATION EXAMPLE

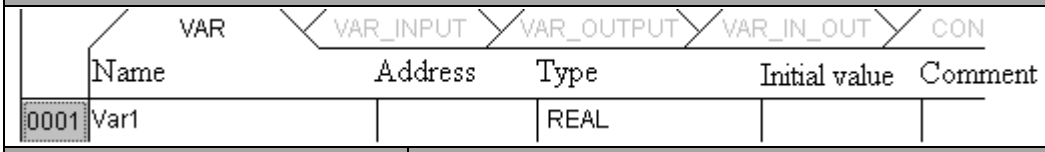
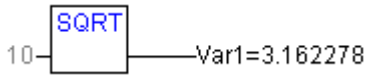
Variable Declaration



Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	i:=ABS(-2); (*result Varint1 is 2*)
Instruction List (IL)	LD -2 ABS ST Varint1 (*result Varint1 is 2*)
Function Block Diagram (FBD)	

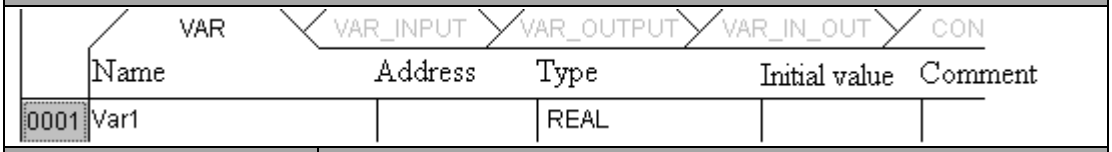
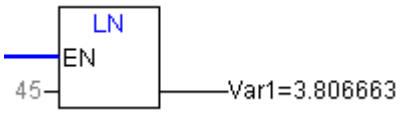
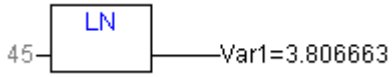
4.8.2 SQRT—Square Root

- Function: Return the square root of the input. The input must be 0 or positive.
- Input/output data type:
- Input data type can be BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT, output data type must be REAL.

<table>
<tr><td colspan="2">APPLICATION EXAMPLE</td></tr>
<tr><td colspan="2">Variable Declaration</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>Programming Language</td><td>Program</td></tr>
<tr><td>Ladder Diagram (LD)</td><td> 
</td></tr>
<tr><td>Structured Text (ST)</td><td>Var1:=SQRT(10); (*result Var1 is 3.162278*)</td></tr>
<tr><td>Instruction List (IL)</td><td>LD 10
SQRT
ST Var1 (*result Var1 is 3.162278*)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td> 
</td></tr>
</table>

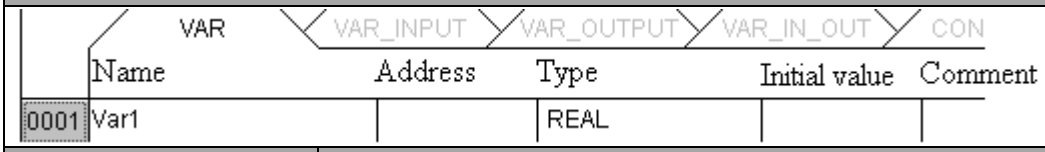
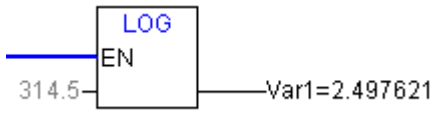
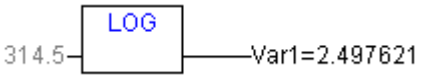
4.8.3 LN—Natural Logarithm

- Function: Return the natural logarithm of input. The input must be positive.
- Input/output data type:
- Input data type can be BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT
- Output data type must be REAL.

<table>
<tr><td colspan="2">APPLICATION EXAMPLE</td></tr>
<tr><td colspan="2">Variable Declaration</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>Programming Language</td><td>Program</td></tr>
<tr><td>Ladder Diagram (LD)</td><td> 
</td></tr>
<tr><td>Structured Text (ST)</td><td>Var1:=LN(45)； (*result Var1 is 3.806663*)</td></tr>
<tr><td>Instruction List (IL)</td><td>LD 45
LN
ST Var1 (*result Var1 is 3.806663*)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td> 
</td></tr>
</table>

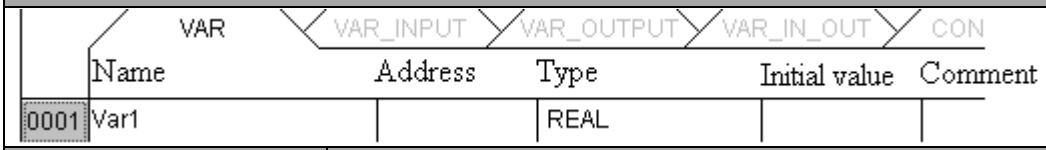
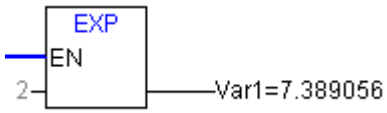
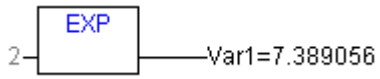
4.8.4 LOG—Logarithm of a Number in Base 10.

- Function: Returns the logarithm of a number in base 10. The input must be positive.
- Input/output data type:
- Input data type can be BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT
- Output data type must be REAL.

<table>
<tr><td colspan="2">APPLICATION EXAMPLE</td></tr>
<tr><td colspan="2">Variable Declaration</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>Programming Language</td><td>Program</td></tr>
<tr><td>Ladder Diagram (LD)</td><td> 
</td></tr>
<tr><td>Structured Text (ST)</td><td>Var1:=LOG(314.5); (*result Var1 is 2.497621 *)</td></tr>
<tr><td>Instruction List (IL)</td><td>LD 314.5
LOG
ST Var1 (*result Var1 is 2.497621 *)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td> 
</td></tr>
</table>

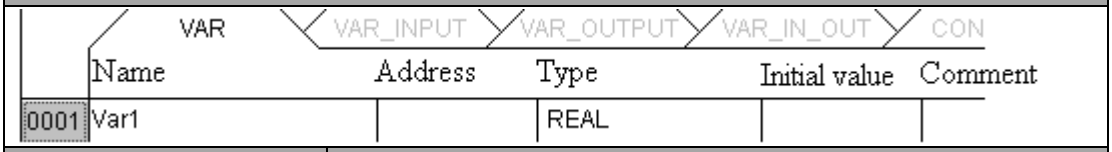
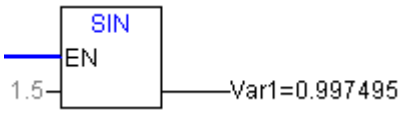
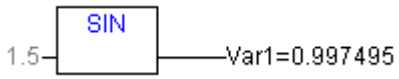
4.8.5 EXP—Exponentiation

- Function: Returns the exponentiation function, i.e. $y = e^x$, where x is the input, y is the output.
- Input/output data type:
- Input data type can be BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT
- Output data type must be REAL.

<table>
<tr><td colspan="2">APPLICATION EXAMPLE</td></tr>
<tr><td colspan="2">Variable Declaration</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>Programming Language</td><td>Program</td></tr>
<tr><td>Ladder Diagram (LD)</td><td> 
</td></tr>
<tr><td>Structured Text (ST)</td><td>Var1:=EXP(2); (*result Var1 is 7.389056*)</td></tr>
<tr><td>Instruction List (IL)</td><td>LD 2
EXP
ST Var1 (*result Var1 is 7.389056*)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td> 
</td></tr>
</table>

4.8.6 SIN—Sine

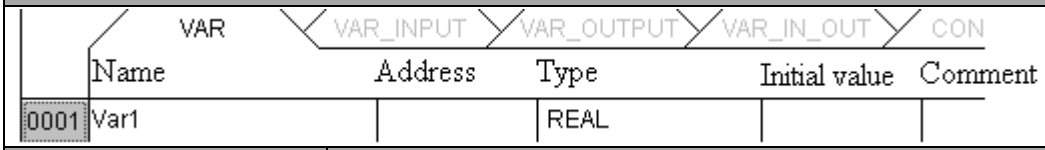
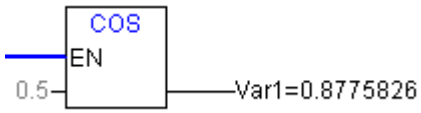
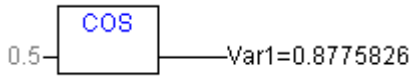
- Function: Returns the sine of the input that is in radian. Radian (rad)=angle* $\frac{\pi}{180}$.
- Input/output data type:
- Input data type can be BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT
- Output data type must be REAL.

<table>
<tr><td colspan="2">APPLICATION EXAMPLE</td></tr>
<tr><td colspan="2">Variable Declaration</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>Programming Language</td><td>Program</td></tr>
<tr><td>Ladder Diagram (LD)</td><td> 
</td></tr>
<tr><td>Structured Text (ST)</td><td>Var1:=SIN(1.5); (*result Var1 is 0.997495 *)</td></tr>
<tr><td>Instruction List (IL)</td><td>LD 1.5
SIN
ST Var1 (*result Var1 is 0.997495 *)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td> 
</td></tr>
</table>

4.8.7 COS—Cosine

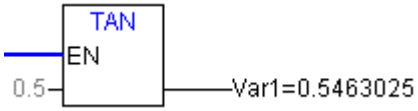
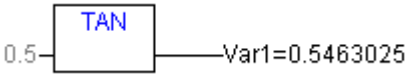
- Function: Return the cosine value of the input that is in radian.

  Radian (rad)=angle* $\frac{\pi}{180}$.
- Input/output data type:
- Input data type can be BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT
- Output data type must be REAL.

<table>
<tr><td colspan="2">APPLICATION EXAMPLE</td></tr>
<tr><td colspan="2">Variable Declaration</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>Programming Language</td><td>Program</td></tr>
<tr><td>Ladder Diagram (LD)</td><td> 
</td></tr>
<tr><td>Structured Text (ST)</td><td>Var1:=COS(0.5); (*result Var1 is 0.8775826 *)</td></tr>
<tr><td>Instruction List (IL)</td><td>LD 0.5
COS
ST Var1 (*result Var1 is 0.8775826 *)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td> 
</td></tr>
</table>

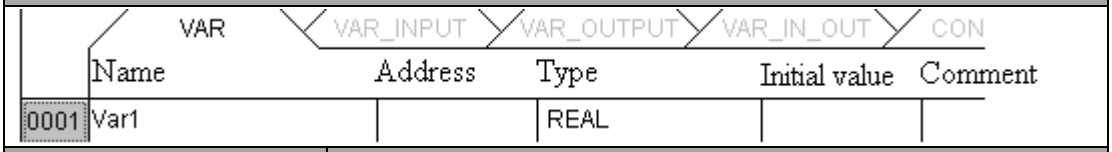
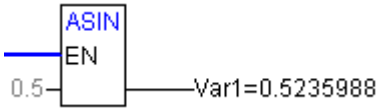
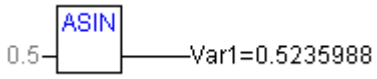
4.8.8 TAN—Tangent

- Function: Return the tangent of the input that is in radian. Radian (rad) =angle* $\frac{\pi}{180}$.
- Input/output data type:
- Input data type can be BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT
- Output data type must be REAL.

APPLICATION EXAMPLE	
Variable Declaration	
	
Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	Var1:=TAN(0.5); (*result Var1 is 0.5463025 *)
Instruction List (IL)	LD 0.5 TAN ST Var1 (*result Var1 is 0.5463025 *)
Function Block Diagram (FBD)	

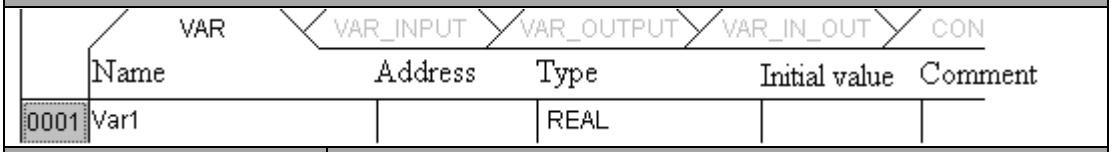
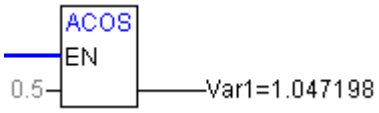
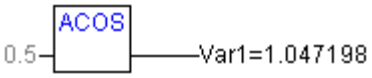
4.8.9 ASIN—ARC sine

- Function: Return the arc sine (inverse sine) of the input data.
- Input/output data type:
- Input data type can be BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT
- Output data type must be REAL, and the output data is in radian.

<table>
<tr><td colspan="2">APPLICATION EXAMPLE</td></tr>
<tr><td colspan="2">Variable Declaration</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>Programming Language</td><td>Program</td></tr>
<tr><td>Ladder Diagram (LD)</td><td> 
</td></tr>
<tr><td>Structured Text (ST)</td><td>Var1:=ASIN(0.5); (*result Var1 is 0.5235988 *)</td></tr>
<tr><td>Instruction List (IL)</td><td>LD 0.5
ASIN
ST Var1 (*result Var1 is 0.5235988 *)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td> 
</td></tr>
</table>

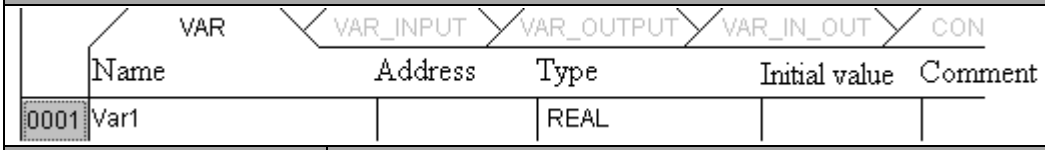
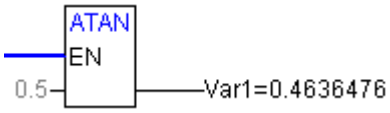
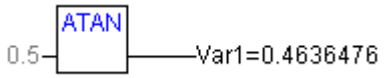
4.8.10 ACOS—ARC cosine

- Function: Return the arc cosine (inverse cosine) of the input data.
- Input/output data type:
- Input data type can be BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT
- Output data type must be REAL, and the output data is in radian.

<table>
<tr><td colspan="2">APPLICATION EXAMPLE</td></tr>
<tr><td colspan="2">Variable Declaration</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>Programming Language</td><td>Program</td></tr>
<tr><td>Ladder Diagram (LD)</td><td> 
</td></tr>
<tr><td>Structured Text (ST)</td><td>Var1:=ACOS(0.5); (*result Var1 is 1.047198 *)</td></tr>
<tr><td>Instruction List (IL)</td><td>LD 0.5
ACOS
ST Var1 (*result Var1 is 1.047198 *)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td> 
</td></tr>
</table>

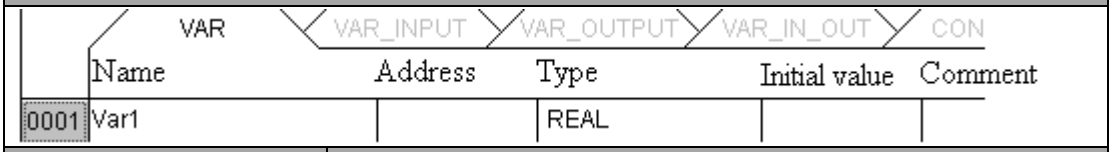
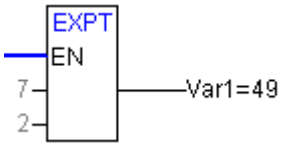
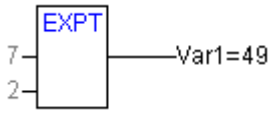
4.8.11 ATAN—ARC tangent

- Function: Return the arc tangent (inverse tangent) of the input.
- Input/output data type:
- Input data type can be BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT
- Output data type must be REAL, and the output data is in radian.

<table>
<tr><td colspan="2">APPLICATION EXAMPLE</td></tr>
<tr><td colspan="2">Variable Declaration</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>Programming Language</td><td>Program</td></tr>
<tr><td>Ladder Diagram (LD)</td><td> 
</td></tr>
<tr><td>Structured Text (ST)</td><td>Var1:=ATAN(0.5); (*result Var1 is 0.4636476 *)</td></tr>
<tr><td>Instruction List (IL)</td><td>LD 0.5
ATAN
ST Var1 (*result Var1 is 0.4636476 *)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td> 
</td></tr>
</table>

4.8.12 EXPT—Exponentiation

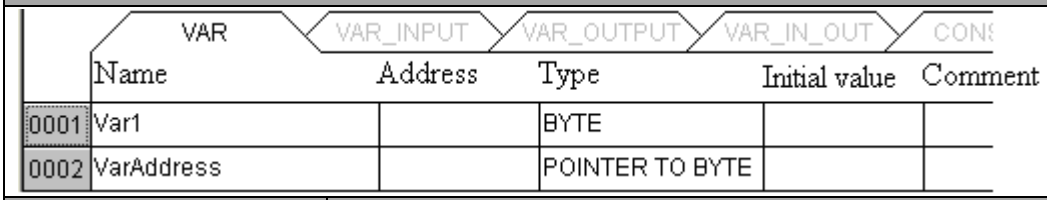
- Function: Exponentiation of a variable with another variable
- Input/output data type:
- Input data type can be BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT
- Output data type must be REAL.

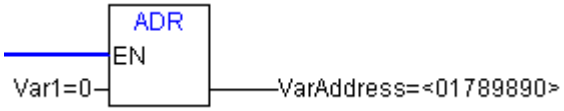
<table>
<tr><td colspan="2">APPLICATION EXAMPLE</td></tr>
<tr><td colspan="2">Variable Declaration</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>Programming Language</td><td>Program</td></tr>
<tr><td>Ladder Diagram (LD)</td><td> 
</td></tr>
<tr><td>Structured Text (ST)</td><td>Var1:=EXPT(7,2); (*result Var1 is 49 *)</td></tr>
<tr><td>Instruction List (IL)</td><td>LD 7
EXPT 2
ST Var1 (*result Var1 is 49 *)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td> 
</td></tr>
</table>

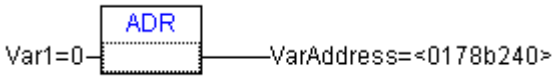
4.9 ADDRESS OPERATORS

4.9.1 ADR—Return the Address of the Variable

- Function: Acquire and output the memory address of the input. The address can be used as a pointer in the program, and also be passed to instructions as a pointer.

<table>
<tr><td colspan="2">APPLICATION EXAMPLE</td></tr>
<tr><td colspan="2">Variable Declaration</td></tr>
<tr><td colspan="2">


</td></tr>
<tr><td>Programming Language</td><td>Program</td></tr>
<tr><td>Ladder Diagram (LD)</td><td>


</td></tr>
<tr><td>Structured Text (ST)</td><td>VarAddress:= ADR(Var1);</td></tr>
<tr><td>Instruction List (IL)</td><td>LD Var1
ADR
ST VarAddress</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td>


</td></tr>
</table>

4.9.2 ^—Return the Content of Address

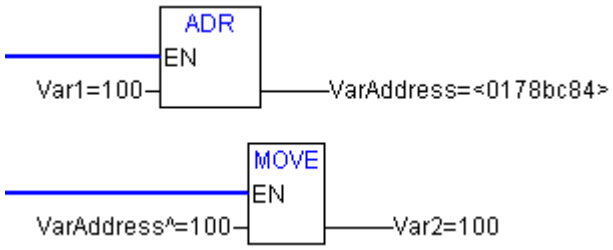
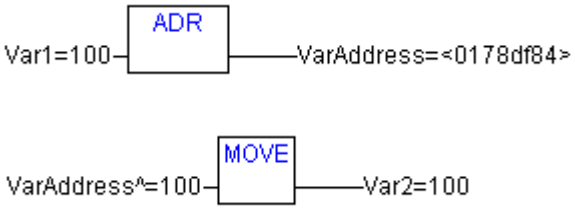
- Function: Return the content of the address by adding the content operator "^"after the pointer variable.

APPLICATION EXAMPLE

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CON

	Name	Address	Type	Initial value	Comment
0001	Var1		BYTE		
0002	Var2		BYTE		
0003	VarAddress		POINTER TO BYTE		

Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	VarAddress:= ADR(Var1); Var2:= VarAddress^; (*result Var2 is 100 *)
Instruction List (IL)	LD Var1 ADR ST VarAddress LD VarAddress^ ST Var2
Function Block Diagram (FBD)	

4.9.3 BITADR—Bit Address

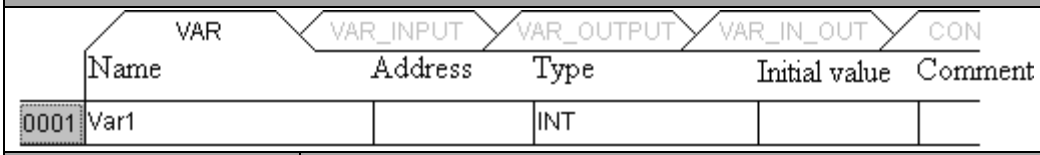
- Function: Return the bit address of BOOL variables. In the following example, the address of MX300.7 is 300*8+7=2407.

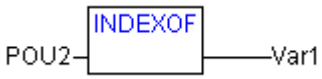
APPLICATION EXAMPLE	
Variable Declaration	
VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONS	
Name | Address | Type | Initial value | Comment	
0001 Varbool1 | %MX300.7 | BOOL	
0002 Bitadr1 | | DWORD	
Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	Bitadr1:=BITADR(Varbool1);
Instruction List (IL)	LD Varbool1 BITADR ST Bitadr1
Function Block Diagram (FBD)	

4.9.4 INDEXOF—Index Of

- Function: Find the internal index of a POU. The input is the name of POU, and the output is an INT variable.

Application Example

<table>
<tr><td colspan="2">APPLICATION EXAMPLE</td></tr>
<tr><td colspan="2">Variable Declaration</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>Programming Language</td><td>Program</td></tr>
<tr><td>Ladder Diagram (LD)</td><td> 

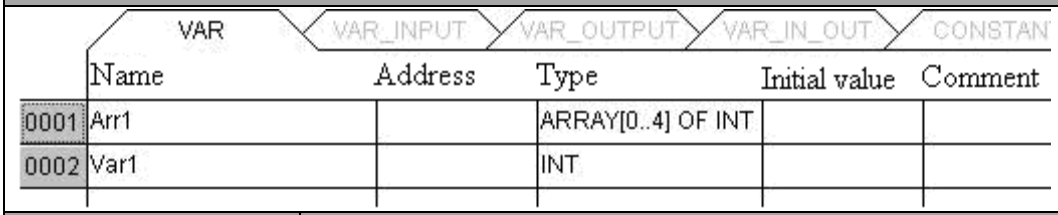
(POU2 can be program,function block, function)</td></tr>
<tr><td>Structured Text (ST)</td><td>Var1:=INDEXOF(POU2); (*result Var1 is 38*)</td></tr>
<tr><td>Instruction List (IL)</td><td>LD POU2
INDEXOF
ST Var1 (*result Var1 is 38*)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td> 

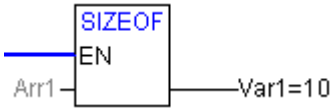
(*result Var1 is 38*)</td></tr>
</table>

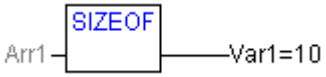
4.9.5 SIZEOF—Number of Bytes

- Function: Return the number of bytes required by the given variable data type.

Application Example

<table>
<tr><td colspan="2">APPLICATION EXAMPLE</td></tr>
<tr><td colspan="2">Variable Declaration</td></tr>
<tr><td colspan="2">


</td></tr>
<tr><td>Programming Language</td><td>Program</td></tr>
<tr><td>Ladder Diagram (LD)</td><td>


</td></tr>
<tr><td>Structured Text (ST)</td><td>Var1:=SIZEOF(arr1); (*result Var1 is 10*)</td></tr>
<tr><td>Instruction List (IL)</td><td>LD arr1
SIZEOF
ST Var1 (*result Var1 is 10*)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td>


</td></tr>
</table>

4.10 CALLING OPERATORS

4.10.1 CAL–calling

- Function: Call function blocks or programs. In IL languages, CAL is used to call function blocks or programs. The input variables of the function block or program being called lie in a pair of braces on the right of the function block or program name.

Application Example

- The following example shows how to call in a program a function block named Inst with the input Par1 = 0, the input Par2 = TRUE, in IL:

```
CAL Inst(Par1:=0, Par2:=TRUE)
```

4.11 INITIALIZATION INSTRUCTION

4.11.1 INI–Initialization Instruction

- Function: Initialize the inside RETAIN variables of a FB program that is used in a POU.

Application Example

```
Syntax: <bool-Variable> := INI(<FB-instance, TRUE|FALSE)
```

- The following is an example of how to call in a program a function block named FB-instance with the input variable Par1=FB-instance, Par2= TRUE|FALSE, and output the initialization completion flag.

APPLICATION EXAMPLE	
Main Program PLC_PRGVariable Declaration：	

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

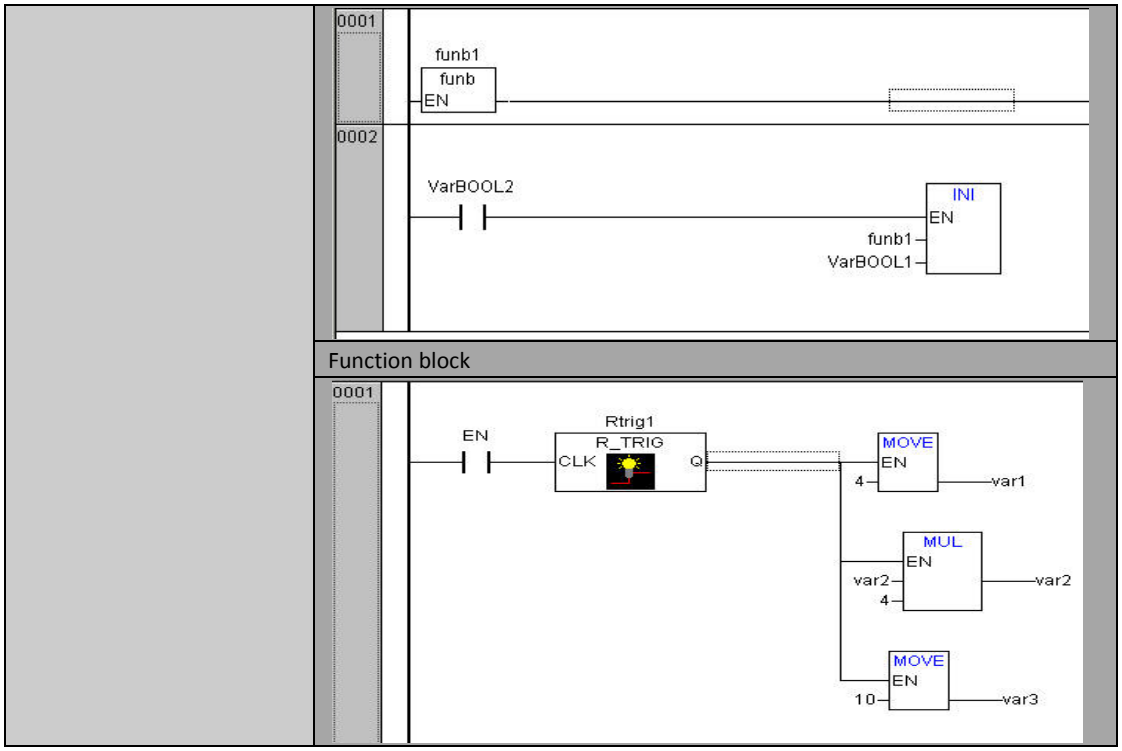
	Name	Address	Type	Initial value	Comment
0001	VarBOOL2		BOOL		
0002	VarBOOL1		BOOL		
0003	funb1		funb		

Variable Declaration：

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT | RETAIN

	Name	Address	Type	Initial value	Comment
0001	Rtrig		R_TRIG		
0002	var1		INT	1	
0003	var2		INT	2	
0004	var3		INT	3	

Programming Language	Program
Ladder Diagram (LD)	PLC_PRG

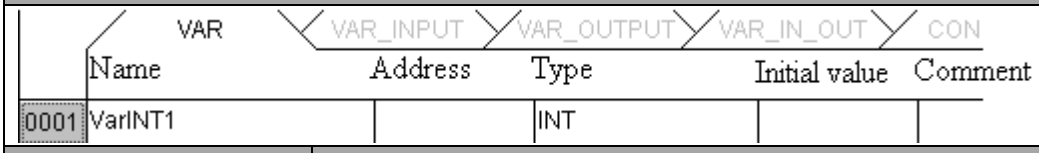
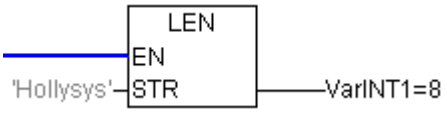


Remarks on the Program:
In the first scanning period of the program operation, the values of var1, var2, var3 are changed into 4, 8 ,10, the forced variable VarBOOL2 is connected, then INI is executed and the three RETAIN variables are restored to the initial values 1, 2, 3.

4.12 STRING INSTRUCTIONS (STANDARD.LIB)

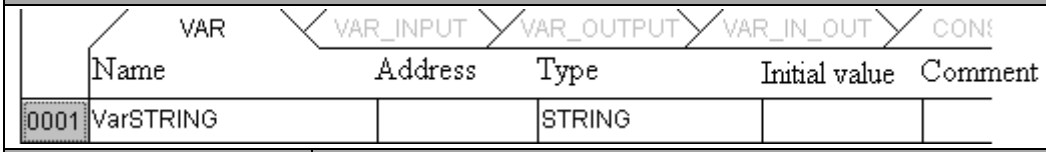
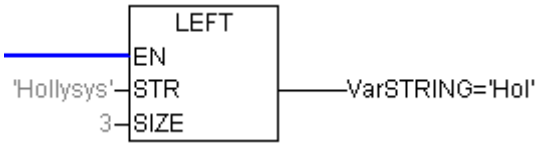
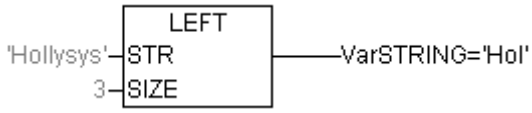
4.12.1 LEN—String Length

- Function: Returns the length of a string.
- Input/output data type: The input data type is STRING, and the output is INT.

<table>
<tr><td colspan="2">APPLICATION EXAMPLE</td></tr>
<tr><td colspan="2">Variable Declaration</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>Programming Language</td><td>Program</td></tr>
<tr><td>Ladder Diagram (LD)</td><td> 
</td></tr>
<tr><td>Structured Text (ST)</td><td>Var1INT1 := LEN (‘Hollysys’); (*result Var1INT1 is 8*)</td></tr>
<tr><td>Instruction List (IL)</td><td>LD ‘Hollysys’
LEN
ST VarINT1 (*result Var1INT1 is 8*)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td> 
</td></tr>
</table>

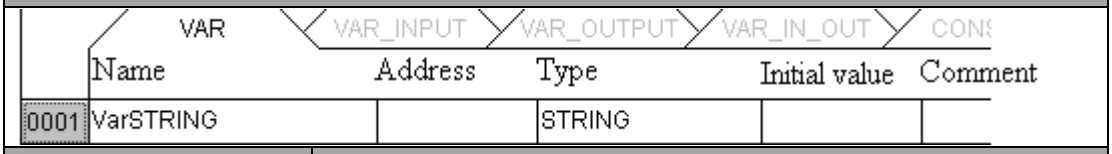
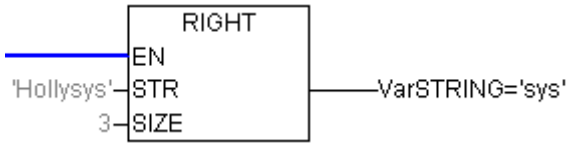
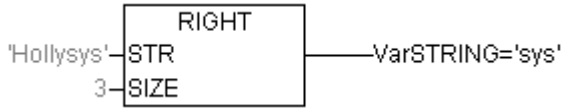
4.12.2 LEFT—Leftmost Characters of STR

- Function: Leftmost SIZE characters of a string.
- Instruction format: LEFT (STR,SIZE), where STR is the STRING type input string, SIZE is the INT type number of characters to be got from the left of the string, and the output is data of STRING type.

<table>
<tr><td colspan="2">APPLICATION EXAMPLE</td></tr>
<tr><td colspan="2">Variable Declaration</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>Programming Language</td><td>Program</td></tr>
<tr><td>Ladder Diagram (LD)</td><td> 
</td></tr>
<tr><td>Structured Text (ST)</td><td>VarSTRING := LEFT (‘Hollysys’,3); (*result is ' Hol '*)</td></tr>
<tr><td>Instruction List (IL)</td><td>LD ' Hollysys '
LEFT 3
ST VarSTRING (*result is ' Hol '*)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td> 
</td></tr>
</table>

4.12.3 RIGHT—Rightmost Characters of STR

- Function: Rightmost SIZE characters of a string.
- Instruction format: RIGHT (STR,SIZE), where STR is the STRING type input string, SIZE is the INT type number of characters to be got from the right of the string, and the output is data of STRING type.

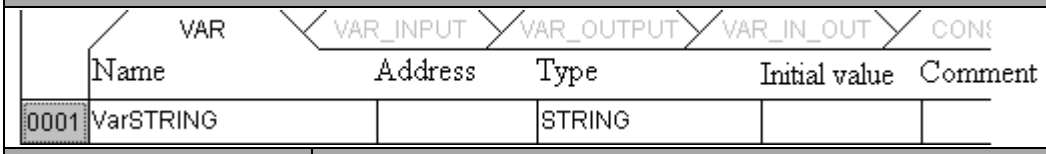
<table>
<tr><td colspan="2">APPLICATION EXAMPLE</td></tr>
<tr><td colspan="2">Variable Declaration</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>Programming Language</td><td>Program</td></tr>
<tr><td>Ladder Diagram (LD)</td><td> 
</td></tr>
<tr><td>Structured Text (ST)</td><td>VarSTRING := RIGHT(‘Hollysys’,3); (*result is ' sys '*)</td></tr>
<tr><td>Instruction List (IL)</td><td>LD ' Hollysys '
RIGHT 3
ST VarSTRING (*result is ' sys '*)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td> 
</td></tr>
</table>

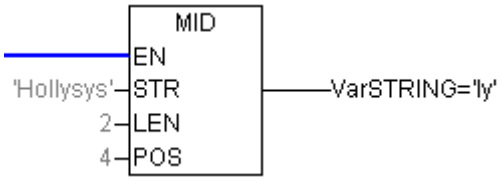
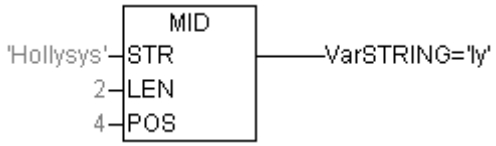
4.12.4 MID—Middle Characters of STR

- Function: return LEN characters of a string beginning at the POS-th character position.
- Instruction format: MID(STR,LEN,POS), where STR is the STRING type input string. LEN and POS are INT variables, MID instruction returns LEN characters of STR, beginning at the POS-th character position from left to right. The output data type is STRING.

APPLICATION EXAMPLE

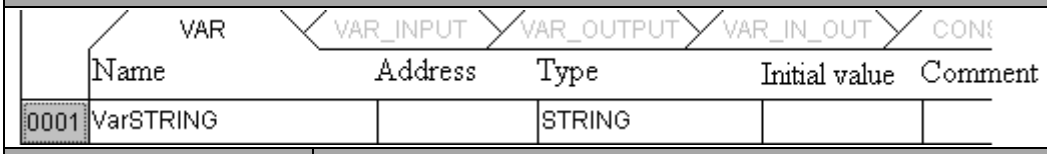
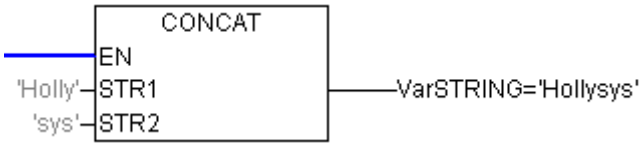
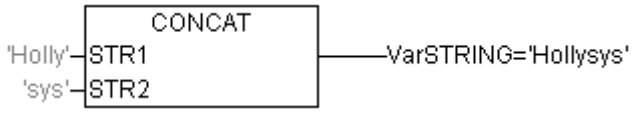
Variable Declaration



Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	VarSTRING:= MID('Hollysys',2,4); (*result is 'ly' *)
Instruction List (IL)	LD 'Hollysys' RIGHT 2,4 ST VarSTRING (*result is 'ly' *)
Function Block Diagram (FBD)	

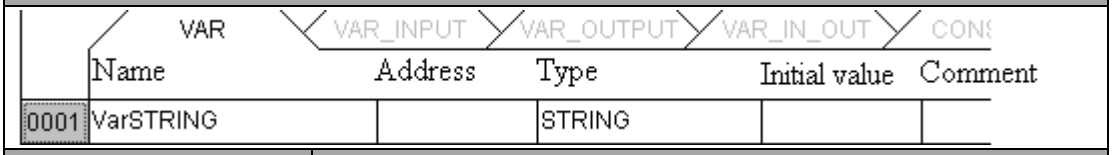
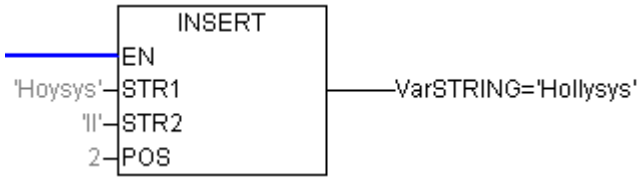
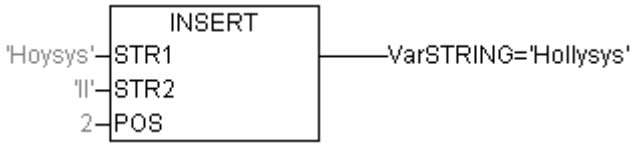
4.12.5 CONCAT—Concatenation of Two STR

- Function: Concatenation of two strings, and the input and output are both STRING type.

APPLICATION EXAMPLE	
Variable Declaration	
	
Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	VarSTRING := CONCAT ('Holly', 'sys'); (*result is 'Hollysys'*)
Instruction List (IL)	LD 'Holly' CONCAT ‘sys’ ST VarSTRING (*result is 'Hollysys'*)
Function Block Diagram (FBD)	

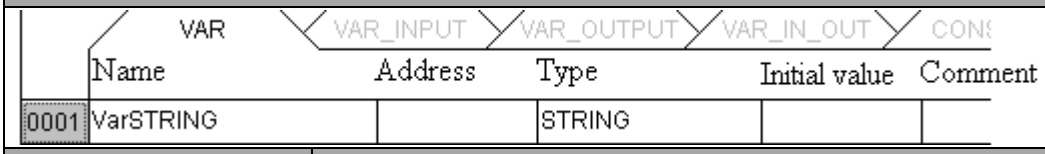
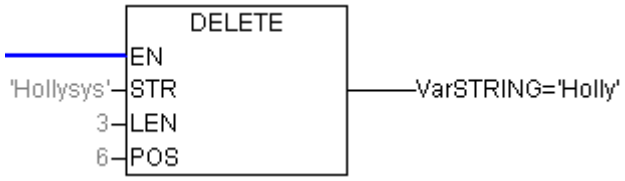
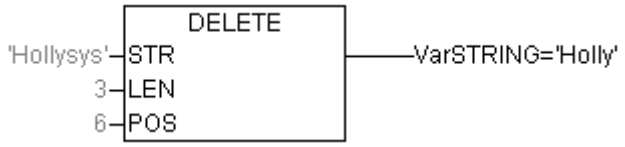
4.12.6 INSERT—Insert STR

- Function: Insert a string STR2 into the string STR1 after the POS-th character position.
- Instruction format: INSERT(STR1,STR2,POS), where STR1 and STR2 are STRING, POS is INT, and output data type is STRING.

<table>
<tr><td colspan="2">APPLICATION EXAMPLE</td></tr>
<tr><td colspan="2">Variable Declaration</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>Programming Language</td><td>Program</td></tr>
<tr><td>Ladder Diagram (LD)</td><td> 
</td></tr>
<tr><td>Structured Text (ST)</td><td>VarSTRING := INSERT ('Hoysys', ‘ll’,2); (*result is 'Hollysys'*)</td></tr>
<tr><td>Instruction List (IL)</td><td>LD 'Hoysys'
INSERT ‘ll’,2
ST VarSTRING (*result is 'Hollysys'*)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td> 
</td></tr>
</table>

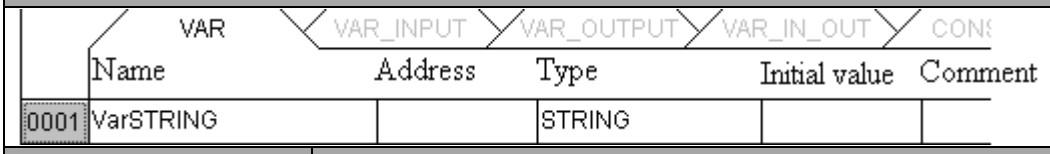
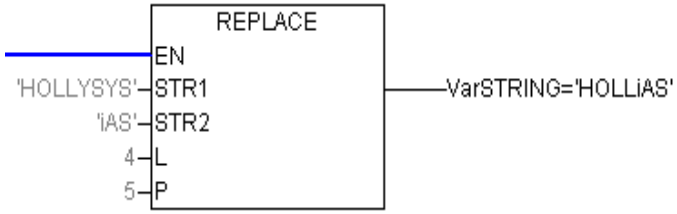
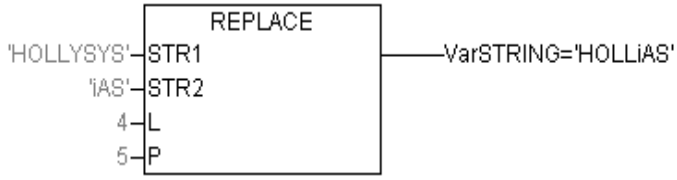
4.12.7 DELETE—Delete Characters of STR

- Function: Delete characters from a string.
- Instruction format: DELETE(STR,LEN,POS), where STR is a STRING type input string, LEN and POS are INT, and the output data type is STRING; the instruction deletes LEN characters of STR, beginning at the POS-th character position from left to right.

<table>
<tr><td colspan="2">APPLICATION EXAMPLE</td></tr>
<tr><td colspan="2">Variable Declaration</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>Programming Language</td><td>Program</td></tr>
<tr><td>Ladder Diagram (LD)</td><td> 
</td></tr>
<tr><td>Structured Text (ST)</td><td>VarSTRING := DELETE ('Hollysys',3,6); (*result is 'Holly'*)</td></tr>
<tr><td>Instruction List (IL)</td><td>LD 'Hollysys'
DELETE 3,6
ST VarSTRING (*result is 'Holly'*)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td> 
</td></tr>
</table>

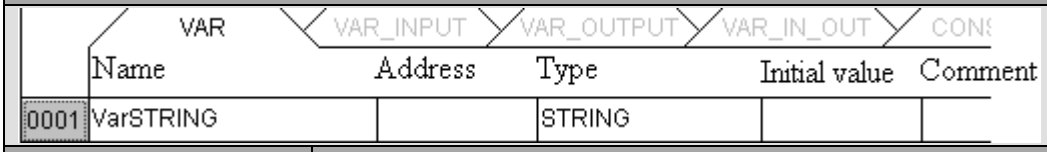
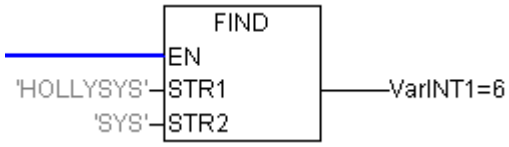
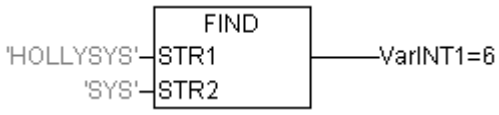
4.12.8 REPLACE—Replace Characters of STR

- Function: Replaces L characters of string STR1 by string STR2, starting at the P-th character position.
- Instruction format: REPLACE (STR1,STR2,L,P), where STR1 and STR2 are input string of STRING, L and P are INT, and the output data type are STRING.

<table>
<tr><td colspan="2">APPLICATION EXAMPLE</td></tr>
<tr><td colspan="2">Variable Declaration</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>Programming Language</td><td>Program</td></tr>
<tr><td>Ladder Diagram (LD)</td><td> 
</td></tr>
<tr><td>Structured Text (ST)</td><td>VarSTRING:= REPLACE ('HOLLYSYS', 'iAS',4,5);
(*result is 'HOLLiAS'*)</td></tr>
<tr><td>Instruction List (IL)</td><td>LD 'HOLLYSYS'
REPLACE 'iAS', 4,5
ST VarSTRING (*result is 'HOLLiAS'*)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td> 
</td></tr>
</table>

4.12.9 FIND—Find Character of STR

- Function: Find the position of the first character where string STR2 appears in string STR1 for the first time.
- Instruction format: FIND(STR1,STR2), where STR1 and STR2 are input string of STRING, and the returned data type is INT. The instruction finds the position of the first character where STR2 appears in STR1 for the first time. If STR2 is not found in STR1, then OUT:=0.

<table>
<tr><td colspan="2">APPLICATION EXAMPLE</td></tr>
<tr><td colspan="2">Variable Declaration</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>Programming Language</td><td>Program</td></tr>
<tr><td>Ladder Diagram (LD)</td><td> 
</td></tr>
<tr><td>Structured Text (ST)</td><td>VarINT1 := FIND ('HOLLYSYS', 'SYS'); (*result is 6*)</td></tr>
<tr><td>Instruction List (IL)</td><td>LD 'HOLLYSYS'
FIND 'SYS'
ST VarINT1 (*result is 6*)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td> 
</td></tr>
</table>

4.13 LIBRARY VERSION CHECK INSTRUCTION (UTIL.LIB)

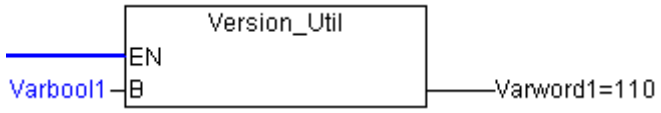
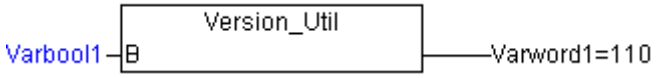
- Version_Util——Library version check
- Function: Read the library version information code of the current software.
- Input/output data type: input: BOOL; output: WORD.

APPLICATION EXAMPLE

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	Name	Address	Type	Initial value	Comment
0001	Varbool1		BOOL		
0002	Varword1		WORD		

Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	Varword1:=Version_Util(Varbool1);
Instruction List (IL)	LD Varbool1 Version_Util ST Varword1
Function Block Diagram (FBD)	

4.14 PLC VERSION CHECK INSTRUCTION (SYSLIBC16X.LIB)

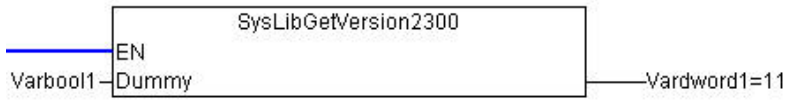
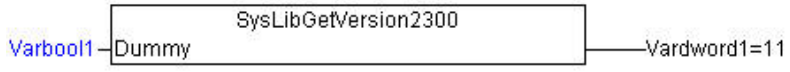
- SyslibGetVersion2300——PLC version check
- Function: Read the version information code of the current PLC.
- Input/output data type: input: BOOL; output: WORD

APPLICATION EXAMPLE

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	Name	Address	Type	Initial value	Comment
0001	Varbool1		BOOL		
0002	Vardword1		DWORD		

Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	Vardword1:= SysLibGetVersion2300 (Varbool1);
Instruction List (IL)	LD Varbool1 SysLibGetVersion2300 ST Vardword1
Function Block Diagram (FBD)	

4.15 CHECK INSTRUCTIONS (CHECK.LIB)

4.15.1 CheckBounds—Check Bounds of Array

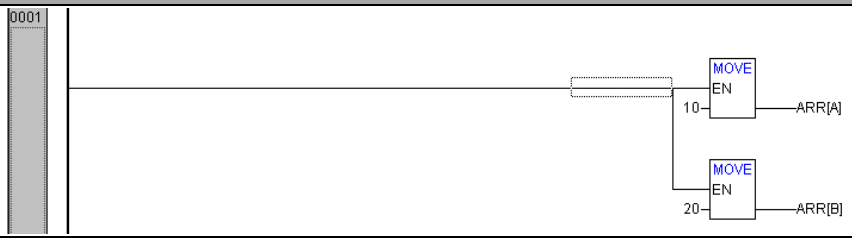
- Function: Use CheckBounds to check bounds of an array, if the array exceeds the bounds, the program will calculate according to bound values.
- Input/output data type: Input: Index, lower, upper: DINT; Output: CheckBounds: DINT.

APPLICATION EXAMPLE

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	Name	Address	Type	Initial value	Comment
0001	Arr		ARRAY[1..7] OF BYT		
0002	A		INT	-2	
0003	B		INT	10	

Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	ARR[A]:=10; ARR[B]:=20;
Instruction List (IL)	LD 10 ST Arr[A] LD 20 ST Arr[B]
Function Block Diagram (FBD)	

Remarks on the Program:

- If the Check.lib library is added, the array bounds can be check without calling the CheckBounds instruction. For example, the initial values of A and B in the program are respectively -2 and 10, exceeding the range 1...7, so Arr[A] and Arr[B] will automatically correspond to Arr[1] and Arr[7], i.e.10 is assigned to Arr[1] while 20 to Arr[7].

4.15.2 CheckDivByte—Check Whether the Divisor is 0 in BYTE

- Function: Check whether the divisor is 0 in BYTE divisions, in case the divisor is 0, it will be calculated as 1.

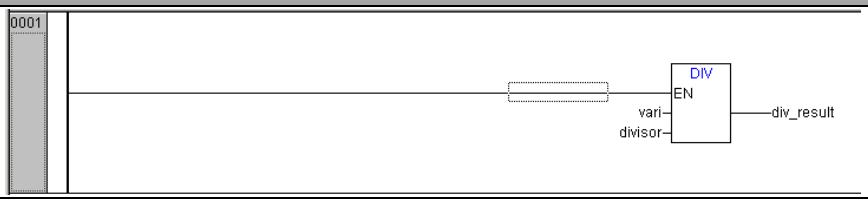
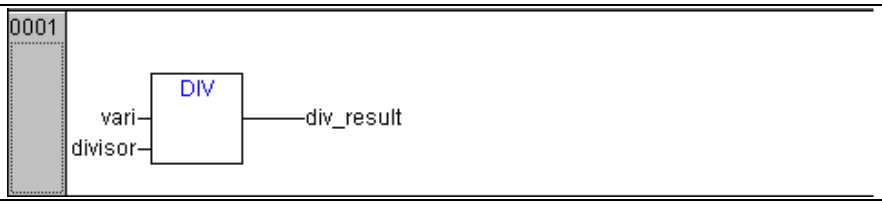
- Input/output data type: Input type: divisor: BYTE; output type: CheckDivByte: BYTE.

APPLICATION EXAMPLE

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	Name	Address	Type	Initial value	Comment
0001	divisor		BYTE	0	
0002	vari		BYTE	100	
0003	div_result		BYTE	10	

Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	Div_result:=vari/divsor;
Instruction List (IL)	LD vari DIV divisor ST div_result
Function Block Diagram (FBD)	

Remarks on the Program:
If the Check.lib library is added, 0 as divisor can be checked without calling the CheckDivByte instruction. If the divisor is 0 in the program, it will be calculated as 1.

4.15.3 CheckDivWord—Check Whether the Divisor is 0 in WORD

- Function: Check whether the divisor is 0 in WORD divisions, in case the divisor is 0, it will be calculated as 1.
- Input/output data type:
- Input type: divisor: WORD; output type: CheckDivByte: WORD.

Application Example

See”4.15.2 CheckDivByte” Application Example, replace BYTE by WORD.

4.15.4 CheckDivDWord—Check Whether the Divisor is 0 in DWORD

- Function: Check whether the divisor is 0 in DWORD divisions, in case the divisor is 0, it will be calculated as 1.
- Input/output data type:
- Input type: divisor: DWORD; output type: CheckDivByte: DWORD.

Application Example

- **See”4.15.2 CheckDivByte” Application Example, replace BYTE by DWORD.**

4.15.5 CheckDivReal—Check Whether the Divisor is 0 in REAL

- Function: Check whether the divisor is 0 in REAL divisions, in case the divisor is 0, it will be calculated as 1.
- Input/output data type:
- Input type: divisor: REAL; output type: CheckDivByte: REAL.

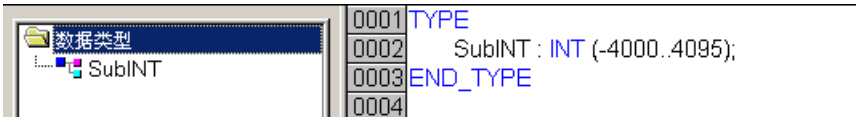
Application Example

- **See 4.15.2 CheckDivByte Application Example, replace BYTE by REAL.**

4.15.6 CheckRangeSigned—Check of Signed Range

- Function: Check whether the subrange variables of INT are over the signed range. If the assigned value is less than the LOWER limit of the subrange variable, then the variable will be defined as the minimum. If the assigned value is greater than the UPPER limit, then the variable will be defined as the maximum.
- Subrange type: It is a subset of a certain basic data type, it can be declared in Data Type (see the figure below), and it also can be declared directly in the Variable Declaration of the program (see the following Application example).

Declare Subrange in Data Type

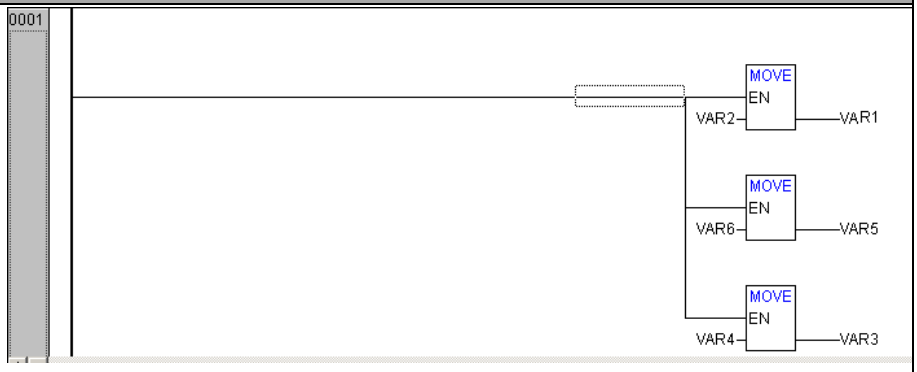
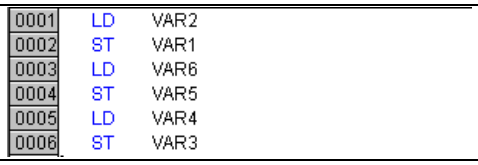


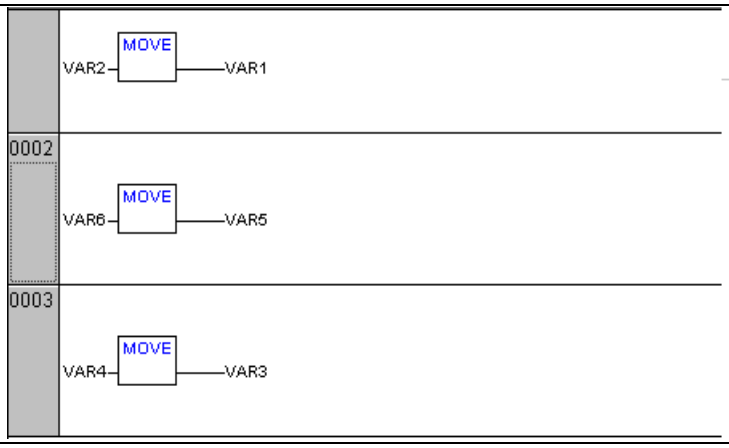
APPLICATION EXAMPLE

Variable Declaration

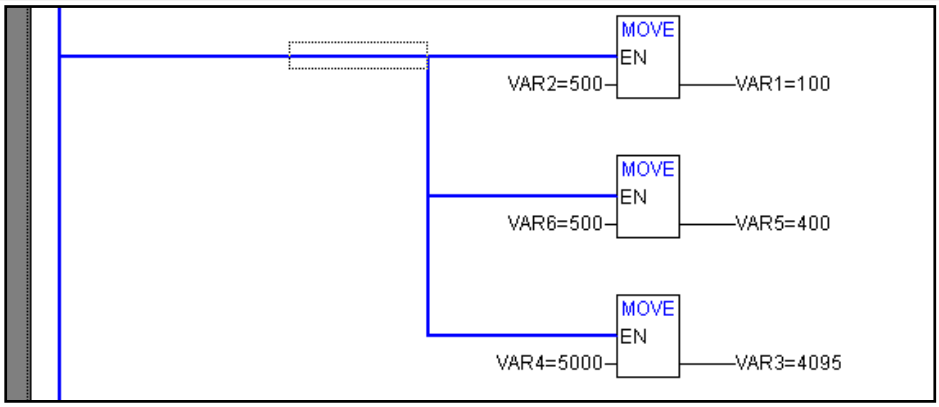
VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	Name	Address	Type	Initial value	Comment
0001	VAR1		INT(-100..100)		
0002	VAR2		INT	500	
0003	VAR3		subint		
0004	VAR4		INT	5000	
0005	VAR5		WORD(100..400)		
0006	VAR6		WORD	500	

Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	VAR1:=VAR2; VAR5:=VAR6; VAR3:=VAR4
Instruction List (IL)	

Function Block Diagram (FBD)	

Remarks on the Program:
If the Check.lib library is added, the calling of CheckRangeSigned instruction will be unnecessary. Variable VAR3 is declared as SubINT with a range of -4000~4095 in the program, assign VAR4=5000, i to VAR3, because 5000 is greater than the UPPER limit 4095, so VAR3 returns the UPPER limit value 4095 after the running of the program. It is the same for VAR1 and VAR5, the results are:



4.15.7 CheckRangeUnsigned—Check of Unsigned Range

- Function: Check whether subrange variables are over the unsigned range. If the assigned value is less than the LOWER limit, then the variable will be defined as the minimum. If the assigned value is greater than the UPPER, then the variable will be defined as the maximum.

Application Example

- **See Application example of 4.15.6CheckRangesigned, change data type to Unsigned.**

4.16 BCD CONVERSIONS (UTIL.LIB)

A byte of BCD (Binary-Coded Decimal) contains integers within the range of 0-99. Each decimal number corresponds to 4 bits, the tens is restored in 4-7 bits and the digit is restored in 0-3 bits. BCD and hexadecimal are similar, the difference lies in BCD value is 0-99 while the hexadecimal value is 0-FF.

In BCD code, 4 bits of binary number stand for a decimal digit, the whole decimal number is indicated by a string of BCD codes. For example, the BCD code of decimal number 59 is 0101 1001 but its binary code is 2#111011. Binary code of decimal number 5 is 0101, and binary code of decimal number 1 is 0001, so BCD code of 51 is 0101 0001.

4.16.1 BCD_TO_INT—BCD to INT

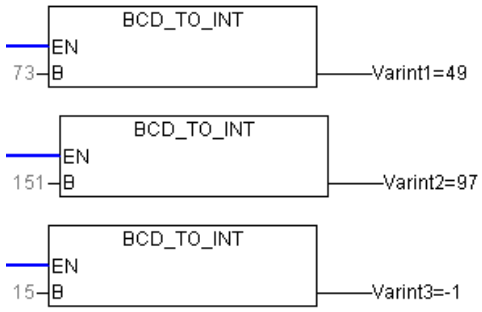
- Function: Convert BCD code to INT value.
- Input/output data type:
- Input B: BYTE, input BCD-Byte in binary format (or decimal and hexadecimal format that correspond to this binary code). For example, the binary format of BCD-Type 49 is 2#100 1001(10#73 or 16#49), so the input shall be 2#100 1001 (10#73 or 16#49).
- Output: INT, return the INT value of BCD-Type. If the input is not a BCD value, -1 will be returned.

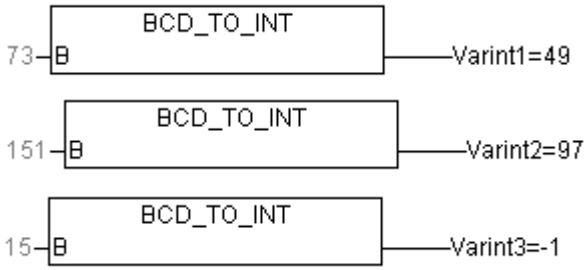
APPLICATION EXAMPLE

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	Name	Address	Type	Initial value	Comment
0001	Varint1		INT		
0002	Varint2		INT		
0003	Varint3		INT		

Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	Varint1:=BCD_TO_INT(73); (*result is 49 *) Varint2:=BCD_TO_INT(151); (*result is 97*) Varint3:=BCD_TO_INT(15); (*output -1, because it is not BCD-Byte *)

Instruction List (IL)	LD 73 BCD_TO_INT ST Varint1 (*result is 49 *)
	LD 151 BCD_TO_INT ST Varint2 (*result is 97*)
	LD 15 BCD_TO_INT ST Varint3 (*output -1, because it is not BCD-Byte *)
Function Block Diagram (FBD)	

4.16.2 INT_TO_BCD—INT to BCD

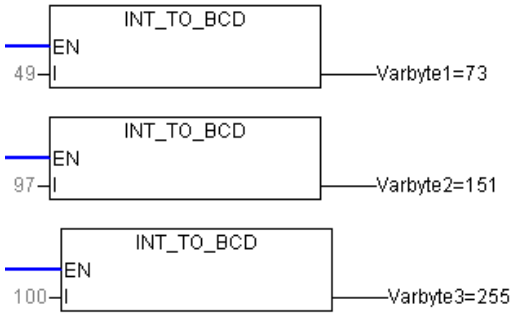
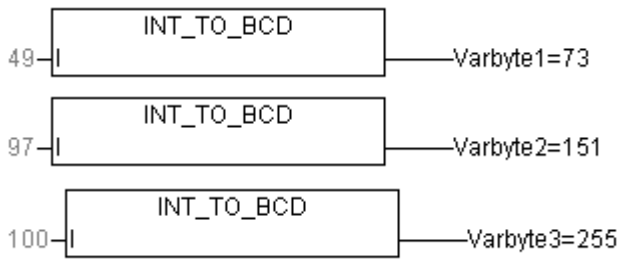
- Function: Convert INT to BCD value. In case of a wrong INT value (<0 or >99) input, 255 will be returned.
- Input/output data type:
- Input: INT, for example, if the input value is 49, then the input INT is 49.
- Output: BYTE, return the BCD value of INT. For example, the BCD-code of 49 is 2#100 1001, then the output will be 2#100 1001 (10#73 or 16#49).

APPLICATION EXAMPLE

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	Name	Address	Type	Initial value	Comment
0001	Varbyte1		BYTE		
0002	Varbyte2		BYTE		
0003	Varbyte3		BYTE		

Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	Varbyte1:=INT_TO_BCD(49); (*result is 73*) Varbyte2:= INT_TO_BCD(97); (*result is 151*) Varbyte3:= INT_TO_BCD(100); (*error! Output:255*)
Instruction List (IL)	LD 49 INT_TO_BCD ST Varbyte1 (*result is 73*) LD 97 INT_TO_BCD ST Varbyte2 (*result is 151*) LD 100 INT_TO_BCD ST Varbyte3 (*error! Output:255*)
Function Block Diagram (FBD)	

4.17 BIT/BYTE INSTRUCTIONS (UTIL.LIB)

4.17.1 EXTRACT—Bit Extract

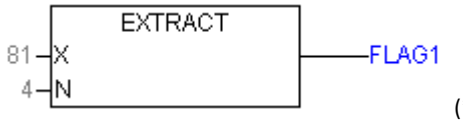
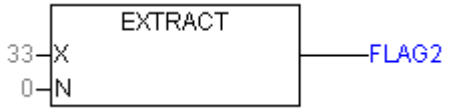
- Function: Return the value of Bit (N=0, 1..) N of the input variable X.
- Input/output data type: Input X: DWORD, N: BYTE, output: BOOL.

APPLICATION EXAMPLE

Variable Declaration

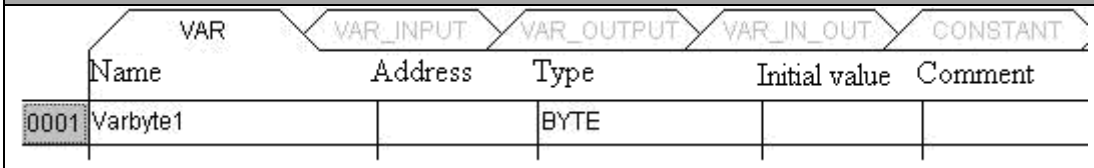
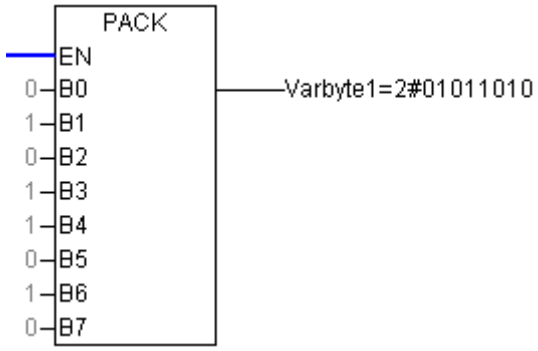
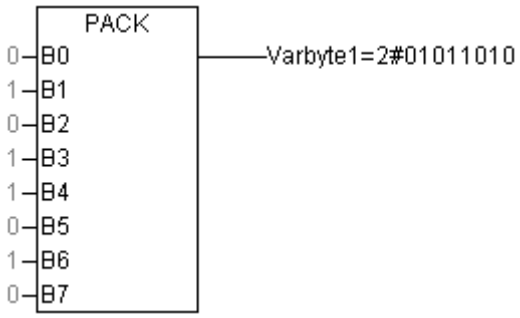
VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	Name	Address	Type	Initial value	Comment
0001	FLAG1		BOOL		
0002	FLAG2		BOOL		

Programming Language	Program
Ladder Diagram (LD)	  (*result FLAG1=TRUE，FLAG2=TRUE *)
Structured Text (ST)	FLAG1:=EXTRACT(X:=81, N:=4); (*result: TRUE, because binary of 81 is 1010001, the 4th is 1*) FLAG2:=EXTRACT(X:=33, N:=0); (*result: TRUE, because binary of 33 is 100001, the 0th is 1*)
Instruction List (IL)	LD 81 EXTRACT 4 ST FLAG1 (*result: TRUE, because binary of 81 is 1010001, the 4th is 1*) LD 33 EXTRACT 0 ST FLAG2 (*result: TRUE, because binary of 81 is 1010001, the 4th is 1*)
Ladder Diagram (LD)	 (*result FLAG1=TRUE *)  (*result FLAG2=TRUE *)

4.17.2 PACK—Bit Pack

- Function: Pack eight input bits (B0, B1…B7) into 1 byte. Reverse to UNPACK.
- Input/output data type: Input B0, B1…B7: BOOL; Output: BYTE.

<table>
<tr><td colspan="2">APPLICATION EXAMPLE</td></tr>
<tr><td colspan="2">Variable Declaration</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>Programming Language</td><td>Program</td></tr>
<tr><td>Ladder Diagram (LD)</td><td> 
</td></tr>
<tr><td>Structured Text (ST)</td><td>Varbyte1:= PACK(0,1,0,1,1,0,1,0); (*result is 2#01011010*)</td></tr>
<tr><td>Instruction List (IL)</td><td>LD FALSE
PACK TRUE,FALSE,TRUE,TRUE,FALSE,TRUE,FALSE
ST Varbyte1 (*result is 2#01011010*)</td></tr>
<tr><td>Function Block Diagram (FBD)</td><td> 
</td></tr>
</table>

4.17.3 PUTBIT—Set Bit Value

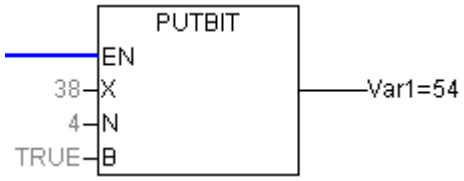
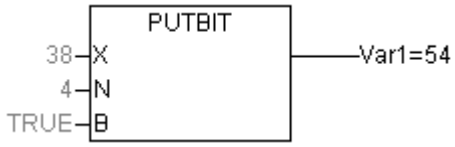
- Function: Set bit N (N=0,1...) of the input variable X as B and returns new value.
- Input/output data type:
- Input X: DWORD, N: BYTE, B: BOOL
- Output: DWORD.

APPLICATION EXAMPLE

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	Name	Address	Type	Initial value	Comment
0001	Var1		DWORD		
0002	Var2		DWORD		

Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	Var2:=38; (*binary 100110*) Var1:=PUTBIT(Var2,4,TRUE); (*result: 54 = 2#110110*)
Instruction List (IL)	LD 38 (*binary 100110*) PUTBIT 4,TRUE ST Var1 (*result: 54 = 2#110110*)
Function Block Diagram (FBD)	

4.17.4 UNPACK—Bit Unpack

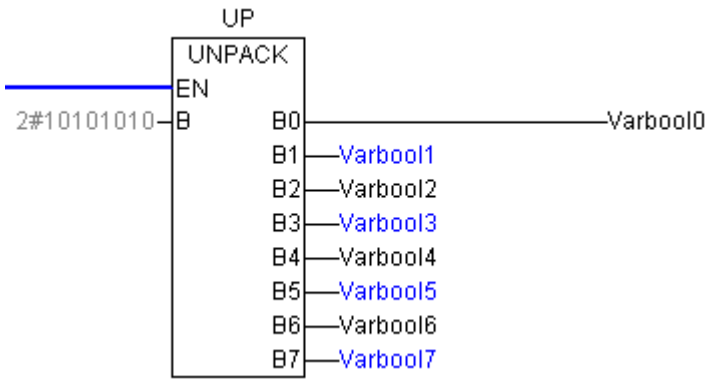
- Function: Unpack 1 byte into 8 bits (B0…B7). Reverse to PACK.
- Inout/putout data type:
- Input: BYTE
- Outout: B0, B1,……,B7: BOOL.

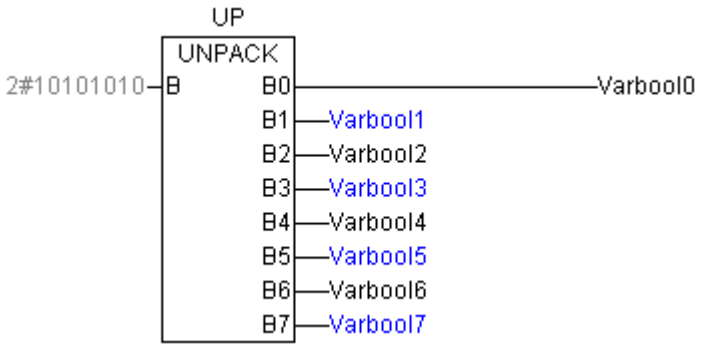
APPLICATION EXAMPLE

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	Name	Address	Type	Initial value	Comment
0001	UP		UNPACK		
0002	Varbool0		BOOL		
0003	Varbool1		BOOL		
0004	Varbool2		BOOL		
0005	Varbool3		BOOL		
0006	Varbool4		BOOL		
0007	Varbool5		BOOL		
0008	Varbool6		BOOL		
0009	Varbool7		BOOL		

Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	UP(B:=2#10101010); Varbool0:=UP.B0; Varbool1:=UP.B1; Varbool2:=UP.B2; Varbool3:=UP.B3; Varbool4:=UP.B4; Varbool5:=UP.B5; Varbool6:=UP.B6; Varbool7:=UP.B7;

Instruction List (IL)	CAL UP(B := 2#10101010) LD UP.B1 ST Varbool1 LD UP.B2 ST Varbool2 LD UP.B3 ST Varbool3 LD UP.B4 ST Varbool4 LD UP.B5 ST Varbool5 LD UP.B6 ST Varbool6 LD UP.B7 ST Varbool7 LD UP.B0 ST Varbool0
Function Block Diagram (FBD)	

4.18 ADVANCED MATHEMATICAL INSTRUCTIONS (UTIL.LIB)

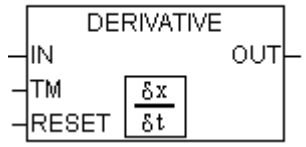
4.18.1 DERIVATIVE—Derivative operation

- Function: Return the derivative of continuous input variables. To acquire the best result, DERIVATIVE instruction always returns the derivative of the four latest input values in order to avoid the errors caused by inaccurate inputs.

Differentiation iterative formula:

$$OUT = \frac{3*[IN(k) - IN(k-3)] + IN(k-1) - IN(k-2)}{3*TM(k-2) + 4*TM(k-1) + 3*TM(k)}$$

Where k-3, k-2, k-1, k are the flags of four continuous inputs.

Example of Ladder Diagram Language


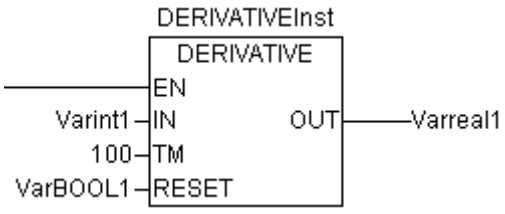
PARAMETER SPECIFICATIONS			
Input	Data Type	Function Description	Value
IN	REAL	continuous input variables	
TM	DWORD	derivative time	ms
RESET	BOOL	reset signal	restart when it is TRUE
Output	Data Type	Function Description	Value
OUT	REAL	derivative result	

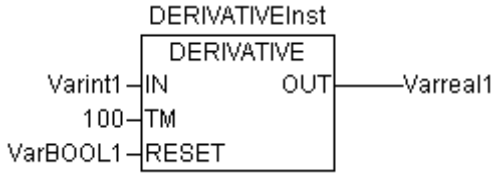
APPLICATION EXAMPLE

Variable Declaration

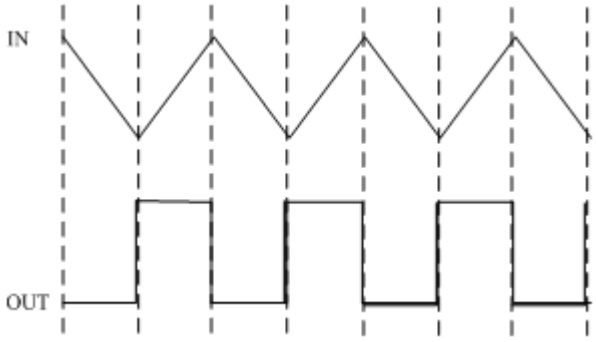
VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	Name	Address	Type	Initial value	Comment
0001	DERIVATIVEInst		DERIVATIVE		
0002	Varreal1		REAL		
0003	Varint1		INT		
0004	VarBOOL1		BOOL		

Programming Language	Program
Ladder Diagram (LD)	

Structured Text (ST)	DERIVATIVEInst (IN:=Varint1,TM:=100,RESET:=VarBOOL1); Varreal1:=DERIVATIVEInst.OUT;
Instruction List (IL)	CAL DERIVATIVEInst(IN := Varint1, TM := 100, RESET := VarBOOL1) LD DERIVATIVEInst.OUT ST Varreal1
Function Block Diagram (FBD)	

The relationship between input and output is shown in the following figure.



4.18.2 INTEGRAL—Integral

- Function: Return the integral of the continuous input variables.

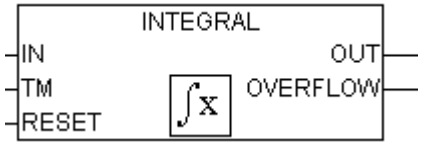
Integral iterative formula:

$$A(k) = A(k-1) + TM * IN(k-1)$$

$$B(k) = B(k-1) + TM * IN(k)$$

$$OUT(k) = \frac{A(k) + B(k)}{2}$$

k-1 and k are flags of two continuous inputs.

Example of Ladder Diagram Language


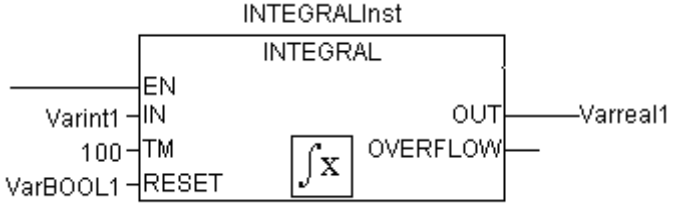
PARAMETER SPECIFICATIONS			
Input	Data Type	Function Description	Value
IN	REAL	continuous input variables	
TM	DWORD	integral time	
RESET	BOOL	reset signal	restart when it is TRUE
Output	Data Type	Function Description	Value
OUT	REAL	integral output	
OVERFLOW	BOOL	flag of overflow	overflow when it is TRUE

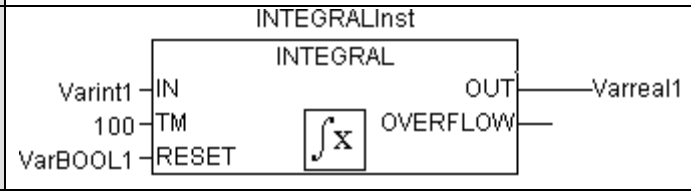
APPLICATION EXAMPLE

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	Name	Address	Type	Initial value	Comment
0001	INTEGRALInst		INTEGRAL		
0002	Varreal1		REAL		
0003	Varint1		REAL		
0004	VarBOOL1		BOOL		

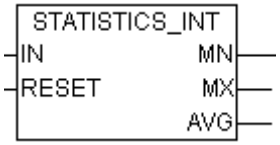
Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	INTEGRALInst(IN := Varint1, TM := 100, RESET := VarBOOL1); Varreal1:=INTEGRALInst.OUT;
Instruction List (IL)	CAL INTEGRALInst(IN := Varint1, TM := 100, RESET := VarBOOL1) LD INTEGRALInst.OUT

	ST Varreal1
Function Block Diagram (FBD)	

The relationship between input and output is shown in the following figure.

4.18.3 STATISTICS_INT—Integer Statistics

- Function: Calculate the minimum, maximum and average of INT data inputs.

Example of Ladder Diagram Language


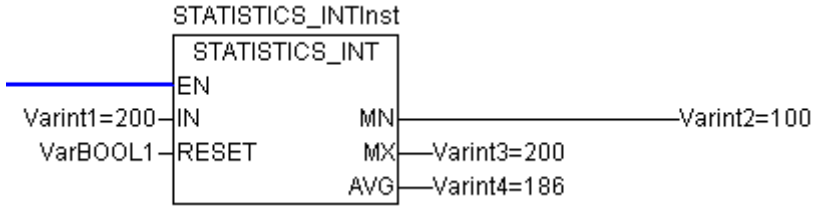
PARAMETER SPECIFICATIONS			
Input	Data Type	Function Description	Value
IN	INT	Input	
RESET	BOOL	Reset	reset when it is TRUE
Output	Data Type	Function Description	Value
MN	INT	minimum	
MX	INT	maximum	
AVG	INT	average	

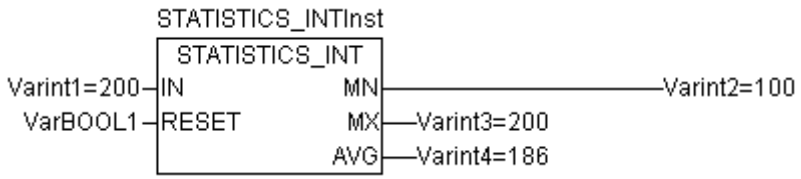
APPLICATION EXAMPLE

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

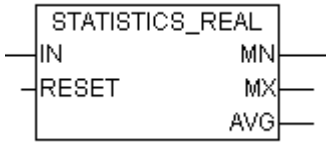
	Name	Address	Type	Initial value	Comment
0001	STATISTICS_INTInst		STATISTICS_INT		
0002	VarBOOL1		BOOL		
0003	Varint1		INT		
0004	Varint2		INT		
0005	Varint3		INT		
0006	Varint4		INT		

Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	STATISTICS_INTInst(IN := Varint1, RESET := VarBOOL1); Varint3:=STATISTICS_INTInst.MX; Varint4:=STATISTICS_INTInst.AVG; Varint2:=STATISTICS_INTInst.MN;

Instruction List (IL)	CAL STATISTICS_INTInst(IN := Varint1, RESET := VarBOOL1) LD STATISTICS_INTInst.MX ST Varint3 LD STATISTICS_INTInst.AVG ST Varint4 LD STATISTICS_INTInst.MN ST Varint2
Function Block Diagram (FBD)	

4.18.4 STATISTICS_REAL—Real Statistics

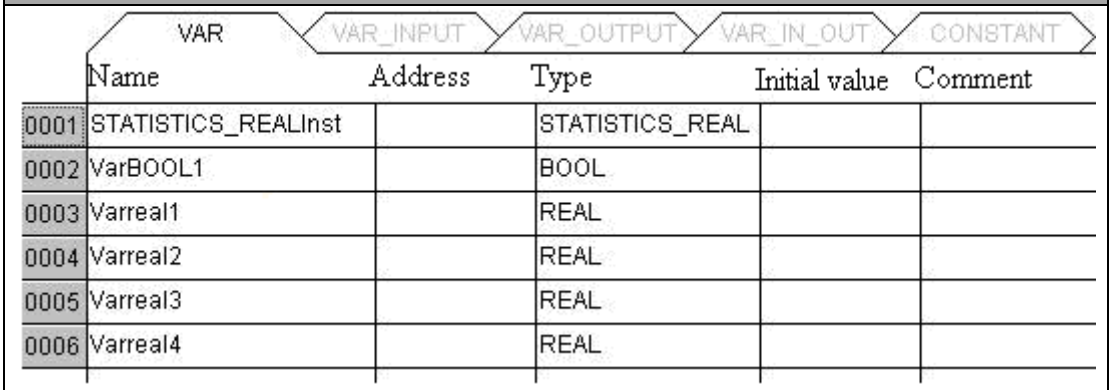
- Function: Calculate the minimum, maximum and average of REAL data inputs.

Example of Ladder Diagram Language


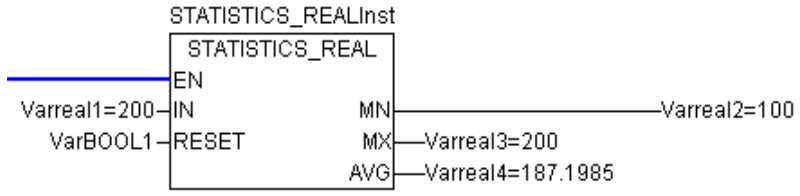
PARAMETER SPECIFICATIONS			
Input	Data Type	Function Description	Value
IN	REAL	input	
RESET	BOOL	reset	Reset when it is TRUE.
Output	Data Type	Function Description	Value
MN	REAL	minimum	
MX	REAL	maximum	
AVG	REAL	average	

APPLICATION EXAMPLE

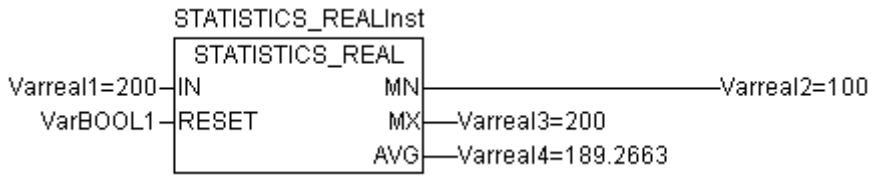
Variable Declaration



	Name	Address	Type	Initial value	Comment
0001	STATISTICS_REALInst		STATISTICS_REAL		
0002	VarBOOL1		BOOL		
0003	Varreal1		REAL		
0004	Varreal2		REAL		
0005	Varreal3		REAL		
0006	Varreal4		REAL		

Programming Language	Program
Ladder Diagram (LD)	

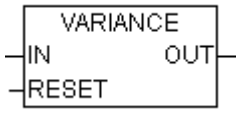
Structured Text (ST)	STATISTICS_REALInst(IN :=Varreal1,RESET:=VarBOOL1); Varreal3:=STATISTICS_REALInst.MX; Varreal4:=STATISTICS_REALInst.AVG; Varreal2:=STATISTICS_REALInst.MN;

Instruction List (IL)	CAL STATISTICS_REALInst(IN := Varreal1, RESET := VarBOOL1) LD STATISTICS_REALInst.MX ST Varreal3 LD STATISTICS_REALInst.AVG ST Varreal4 LD STATISTICS_REALInst.MN ST Varreal2
Function Block Diagram (FBD)	

Note:
The function is the same with STATISTICS_INT, and the only difference is that the input and the output data types for this instruction are REAL.

4.18.5 VARIANCE—Square Variance

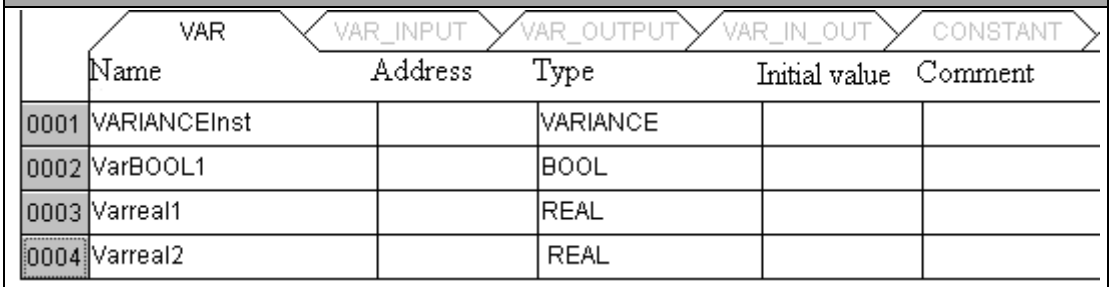
- Function: Calculate the mathematical square variance of a variable. The standard variance is the square root of the square variance.

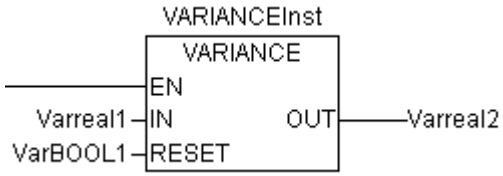
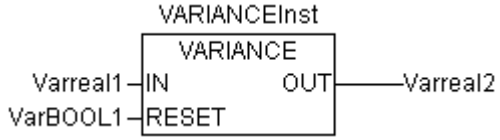
Example of Ladder Diagram Language


PARAMETER SPECIFICATIONS			
Input	Data Type	Function Description	Value
IN	REAL	Input	
RESET	BOOL	Reset	Reset when it is TRUE.
Output	Data Type	Function Description	Value
OUT	REAL	square variance	

APPLICATION EXAMPLE

Variable Declaration



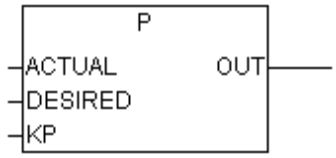
Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	VARIANCEInst(IN := Varreal1, RESET := VarBOOL1); Varreal2:=VARIANCEInst.OUT;
Instruction List (IL)	CAL VARIANCEInst(IN := Varreal1, RESET := VarBOOL1) LD VARIANCEInst.OUT ST Varreal2
Function Block Diagram (FBD)	

4.19 CONTROLLERS (UTIL.LIB)

4.19.1 P—Proportion Controller

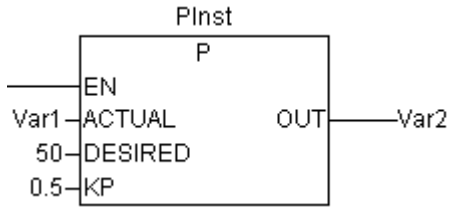
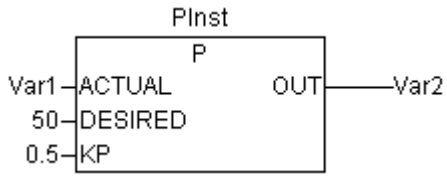
- Function: Proportion controller

- Control equation: OUT=ACTUAL+ (DESIRED-ACTUAL)*KP.
- Input/output data type:
- Inputs: actual value ACTUAL, desired value DESIRED, proportionality const KP: REAL
- Output: OUT: REAL.

Example of Ladder Diagram Language


PARAMETER SPECIFICATIONS			
Input	Data Type	Function Description	Value
ACTUAL	REAL	actual value	
DESIRED	REAL	desired value	
KP	REAL	proportion	
Output	Data Type	Function Description	Value
OUT	REAL	output	

APPLICATION EXAMPLE					
Variable Declaration					
VAR / VAR_INPUT / VAR_OUTPUT / VAR_IN_OUT / CONSTANT					
	Name	Address	Type	Initial value	Comment
0001	PInst		P		
0002	Var1		REAL		
0003	Var2		REAL		

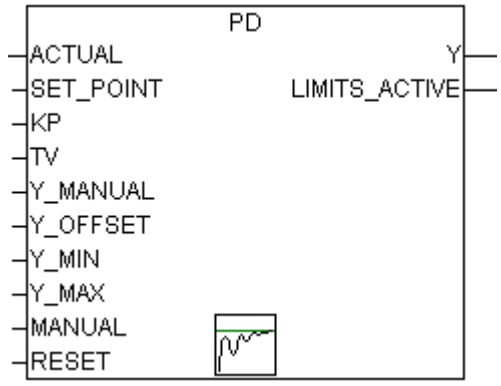
Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	PInst(ACTUAL := Var1, DESIRED := 50, KP := 0.5); Var2:=PInst.OUT;
Instruction List (IL)	CAL PInst(ACTUAL := Var1, DESIRED := 50, KP := 0.5) LD PInst.OUT ST Var2
Function Block Diagram (FBD)	

4.19.2 PD—PD Controller

- Function: Proportion and Derivative controller
- **Control equation:** $\Delta = \mathrm{SET}\_POINT - ACTUAL$

$$Y = KP * (\Delta + TV * \frac{\sigma\Delta}{\sigma t}) + Y\_OFFSET$$

- $\frac{\sigma\Delta}{\sigma t}$ **will be calculated automatically, users do not need to consider it.**

Example of Ladder Diagram Language


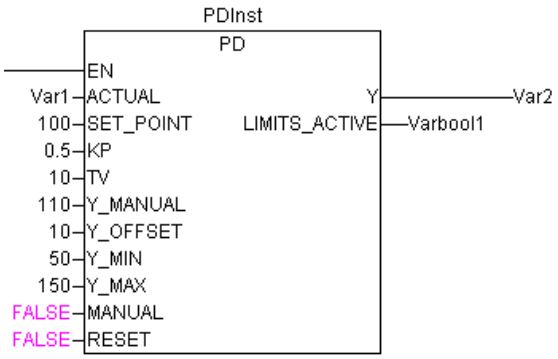
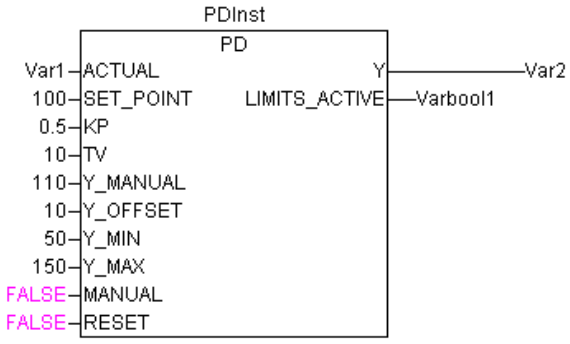
PARAMETER SPECIFICATIONS			
Input	Data Type	Function Description	Value
ACTUAL	REAL	actual value	
SET_POINT	REAL	desired value	
KP	REAL	proportionality const.	
TV	REAL	derivative time	in sec
Y_MANUAL	REAL	manual value	MANUAL=TRUE时, Y= Y_MANUAL
Y_OFFSET	REAL	offset for manipulated variable	
Y_MIN	REAL	minimum value for manipulated variable	
Y_MAX	REAL	maximum value for manipulated variable	
MANUAL	BOOL	manual/auto	TRUE: manual; FALSE: auto.
RESET	BOOL	reset	Reset TRUE: reset, FALSE: normal run
Output	Data Type	Function Description	Value
Y	REAL	output value	
LIMITS_ACTIVE	BOOL	exceed limits	TRUE: manipulated variable exceeds limits Y_MIN, Y_MAX.

APPLICATION EXAMPLE

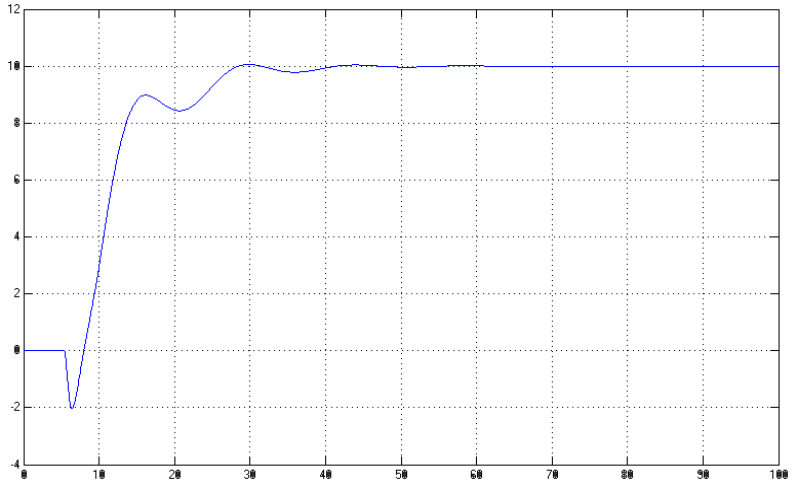
Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	Name	Address	Type	Initial value	Comment
0001	PDInst		PD		
0002	Var1		REAL		
0003	Var2		REAL		
0004	Varbool1		BOOL		

Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	PDInst(ACTUAL := Var1, SET_POINT := 100, KP := 0.5, TV := 10, Y_MANUAL := 110, Y_OFFSET := 10, Y_MIN := 50, Y_MAX := 150, MANUAL := FALSE, RESET := FALSE); Varbool1:=PDInst.LIMITS_ACTIVE; Var2:=PDInst.Y;
Instruction List (IL)	CAL PDInst(ACTUAL := Var1, SET_POINT := 100, KP := 0.5, TV := 10, Y_MANUAL := 110, Y_OFFSET := 10, Y_MIN := 50, Y_MAX := 150, MANUAL := FALSE, RESET := FALSE) LD PDInst.LIMITS_ACTIVE ST Varbool1 LD PDInst.Y ST Var2
Function Block Diagram (FBD)	

The adjusted waveform is shown as below.



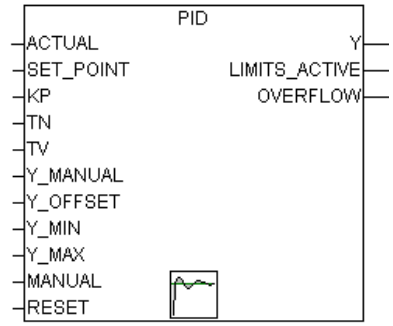
4.19.3 PID—Proportion, Integral and Derivative Controller

- Function: Proportion integral and derivative controller. The PID controller can be used as a PI control when TV=0.
- **Control equation:**

$$Y = KP * (\Delta + \frac{1}{TN} * \int \Delta(t)dt + TV * \frac{\sigma\Delta}{\sigma t}) + Y\_OFFSET$$

$$\Delta = SET\_POINT - ACTUAL$$

$\int \Delta(t)dt$ and $\frac{\sigma\Delta}{\sigma t}$ **will be calculated automatically, no need to consider it for users.**

Example of Ladder Diagram Language


Parameter Specifications

PARAMETER SPECIFICATIONS			
Input	Data Type	Function Description	Value
ACTUAL	REAL	actual value	
SET_POINT	REAL	desired value	
KP	REAL	proportionality const.	
TN	REAL	Integral time	in sec
TV	REAL	derivative time	in sec
Y_MANUAL	REAL	manual value	When MANUAL=TRUE, Y= Y_MANUAL
Y_OFFSET	REAL	offset for manipulated variable	
Y_MIN	REAL	minimum value for manipulated variable	
Y_MAX	REAL	maximum value for manipulated variable	
MANUAL	BOOL	manual/auto	TRUE: manual; FALSE: auto.
RESET	BOOL	reset	TRUE: reset, FALSE: normal run
Output	Data Type	Function Description	Value
Y	REAL	output value	
LIMITS_ACTIVE	BOOL	exceed limits	TRUE: manipulated variable exceeds limits Y_MIN, Y_MAX.
OVERFLOW	BOOL	overflow	When integral overflow, is TRUE

APPLICATION EXAMPLE

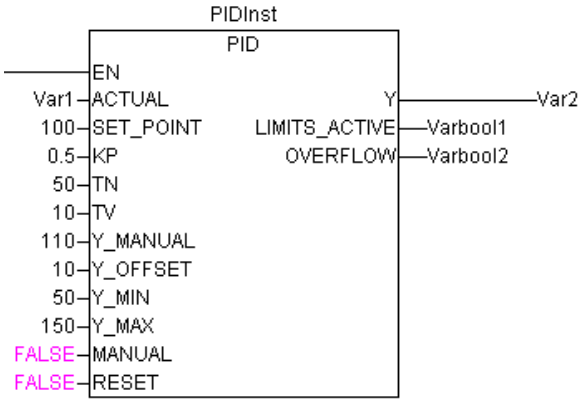
Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	Name	Address	Type	Initial value	Comment
0001	PIDInst		PID		
0002	Var1		REAL		
0003	Var2		REAL		
0004	Varbool1		BOOL		
0005	Varbool2		BOOL		

Programming Language	Program

Ladder Diagram (LD)



Structured Text (ST)

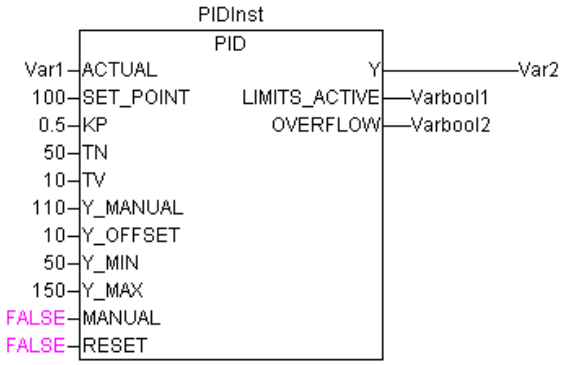
```
PIDInst(ACTUAL := Var1, SET_POINT := 100, KP := 0.5, TN := 50,
TV := 10, Y_MANUAL := 110, Y_OFFSET := 10, Y_MIN := 50,
Y_MAX := 150, MANUAL := FALSE, RESET := FALSE);
Varbool1:=PIDInst.LIMITS_ACTIVE;
Varbool2:=PIDInst.OVERFLOW;
Var2:=PIDInst.Y;
```

Instruction List (IL)

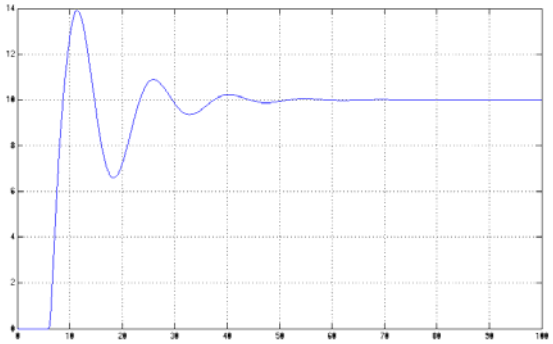
```
CAL     PIDInst(ACTUAL := Var1, SET_POINT := 100, KP := 0.5, TN := 50, TV := 10, Y_MANUAL
:= 110, Y_OFFSET := 10, Y_MIN := 50, Y_MAX := 150, MANUAL := FALSE, RESET := FALSE)

LD          PIDInst.LIMITS_ACTIVE
ST          Varbool1
LD          PIDInst.OVERFLOW
ST          Varbool2
LD          PIDInst.Y
ST          Var2
```

Function Block Diagram (FBD)



The adjusted waveform is shown as below.



4.19.4 PID_FIXCYCLE—PID Controller with Fixed Cycle Time

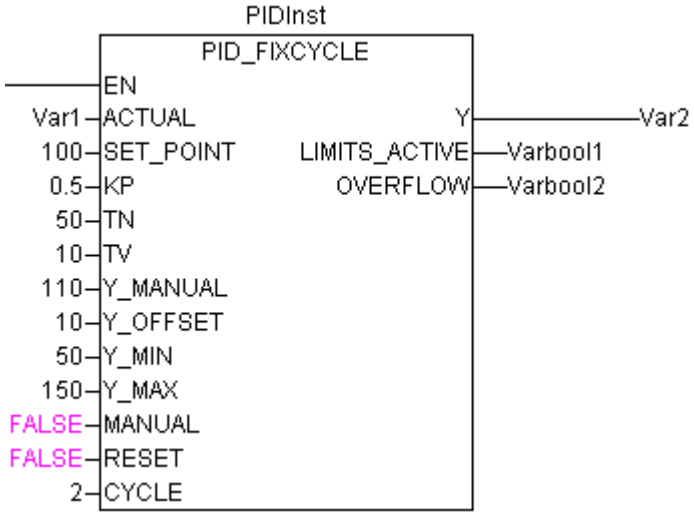
- Function: PID controller. Compared with the abovementioned PID controller, it has an extra parameter of sampling cycle period CYCLE, a REAL data input to set the steps of reset time and rate time in second. See PID for detailed control equation and Values.

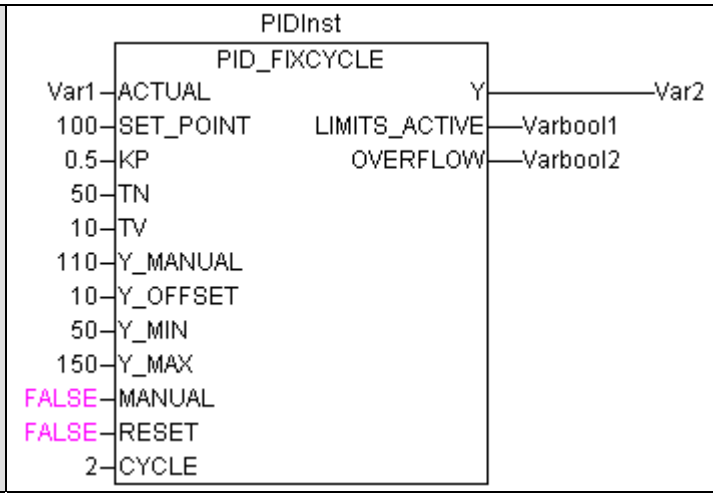
APPLICATION EXAMPLE

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	Name	Address	Type	Initial value	Comment
0001	Var1		REAL		
0002	Var2		REAL		
0003	Varbool1		BOOL		
0004	Varbool2		BOOL		
0005	PIDInst		PID_FIXCYCLE		

Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	PIDInst(ACTUAL := Var1, SET_POINT := 100, KP := 0.5, TN := 50, TV := 10, Y_MANUAL := 110, Y_OFFSET := 10, Y_MIN := 50, Y_MAX := 150, MANUAL := FALSE, RESET := FALSE, CYCLE := 2); Varbool1:=PIDInst.LIMITS_ACTIVE; Varbool2:=PIDInst.OVERFLOW; Var2:= PIDInst.Y;
Instruction List (IL)	CAL PIDInst(ACTUAL := Var1, SET_POINT := 100, KP := 0.5, TN := 50, TV := 10, Y_MANUAL := 110, Y_OFFSET := 10, Y_MIN := 50, Y_MAX := 150, MANUAL := FALSE, RESET := FALSE, CYCLE := 2) LD PIDInst.LIMITS_ACTIVE ST Varbool1 LD PIDInst.OVERFLOW ST Varbool2 LD PIDInst.Y ST Var2

Function Block Diagram (FBD)	

The adjusted waveform is the same as that of PID.

4.20 SIGNAL GENERATORS (UTIL.LIB)

4.20.1 BLINK—Pulse Signal Generator

- Function: Simulate a pulse signal. After the BLINK instruction is executed, the high voltage time TIMEHIGH and the low voltage time TIMELOW will be output cyclically.

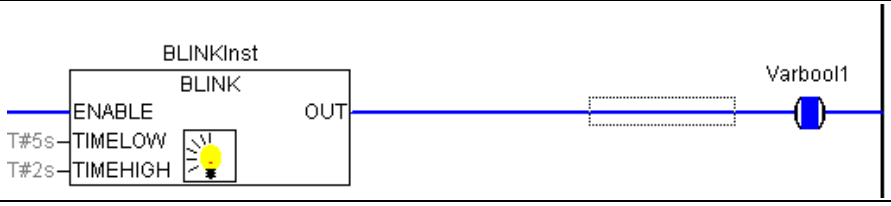
PARAMETER SPECIFICATIONS			
Input	Data Type	Function Description	Value
ENABLE	BOOL	enable	TRUE: run
TIMELOW	TIME	time for OUT=FALSE	
TIMEHIGH	TIME	time for OUT=TRUE	
Output	Data Type	Function Description	Value
OUT	BOOL	output variable	

APPLICATION EXAMPLE

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	Name	Address	Type	Initial value	Comment
0001	Varbool1		BOOL		
0002	PIDInst		PID_FIXCYCLE		
0003	BLINKInst		BLINK		

Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	BLINKInst(TIMELOW := T#5s,TIMEHIGH := T#2s); Varbool1:=BLINKInst.OUT;
Instruction List (IL)	CAL BLINKInst(TIMELOW := T#5s,TIMEHIGH := T#2s) LD BLINKInst.OUT ST Varbool1
Function Block Diagram (FBD)	

The waveform of OUT after the execution of this instruction is shown in figure 4-20-1:

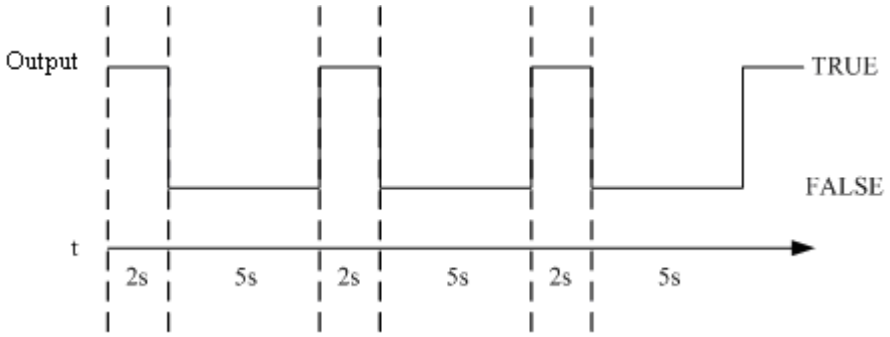


Figure 4-20-1

4.20.2 GEN—Periodic Signal Generator

- Function: Generate typical periodic signals, such as TRIANGLE, TRIANGLE_POS, SAWTOOTH_RISE, SAWTOOTH_FALL, RECTANGLE, SINUS and COSINUS.

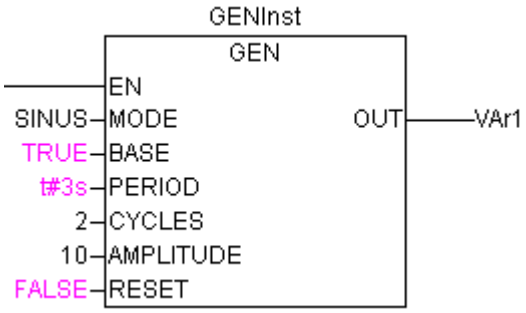
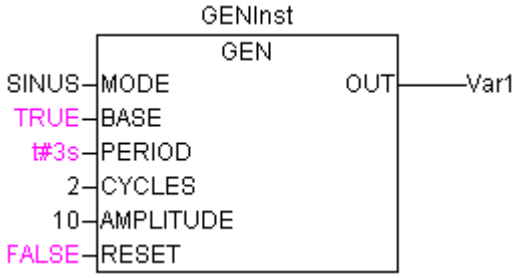
PARAMETER SPECIFICATIONS			
Input	Data Type	Function Description	Value
MODE	GEN_MODE	signal type	signal type on MODE, TRIANGLE, TRIANGLE_POS, SAWTOOTH_RISE, SAWTOOTH_FALL, RECTANGLE, SINUS, COSINUS,
		TRIANGLE	triangularfrom - AMPL. to + AMPL
		TRIANGLE_POS	triangularfrom 0 to AMPL
		SAWTOOTH_RISE	sawtooth increasing from -AMPL. to +AMPL
		SAWTOOTH_FALL	sawtooth decreasing from AMPL to –AMPL
		RECTANGLE	rectangular switching from -AMPL. to +AMPL
		SINUS	sine
		COSINU	cosine
BASE	BOOL	circle mode	FALSE: period referring to call; TRUE: period referring to time
PERIOD	TIME	period	
CYCLES	INT	cycles	
AMPLITUDE	INT	signal amplitude	
RESET	BOOL	reset	reset when RESET=TRUE
Output	Data Type	Function Description	Value
OUT	INT	output	

APPLICATION EXAMPLE

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	Name	Address	Type	Initial value	Comment
0001	GENInst		GEN		
0002	Var1		INT		

Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	GENInst(MODE:=SINUS, BASE:=TRUE, PERIOD:=T#3s, CYCLES:=2, AMPLITUDE:=10, RESET:=FALSE); Var1:=GENInst.OUT;
Instruction List (IL)	CAL GENInst(MODE:=SINUS, BASE:=TRUE, PERIOD:=T#3s, CYCLES:=2, AMPLITUDE:=10, RESET:=FALSE) LD GENInst.OUT ST Var1
Function Block Diagram (FBD)	

According to different inputs in MODE, waveforms generated are as follows in figure 4-20-2:

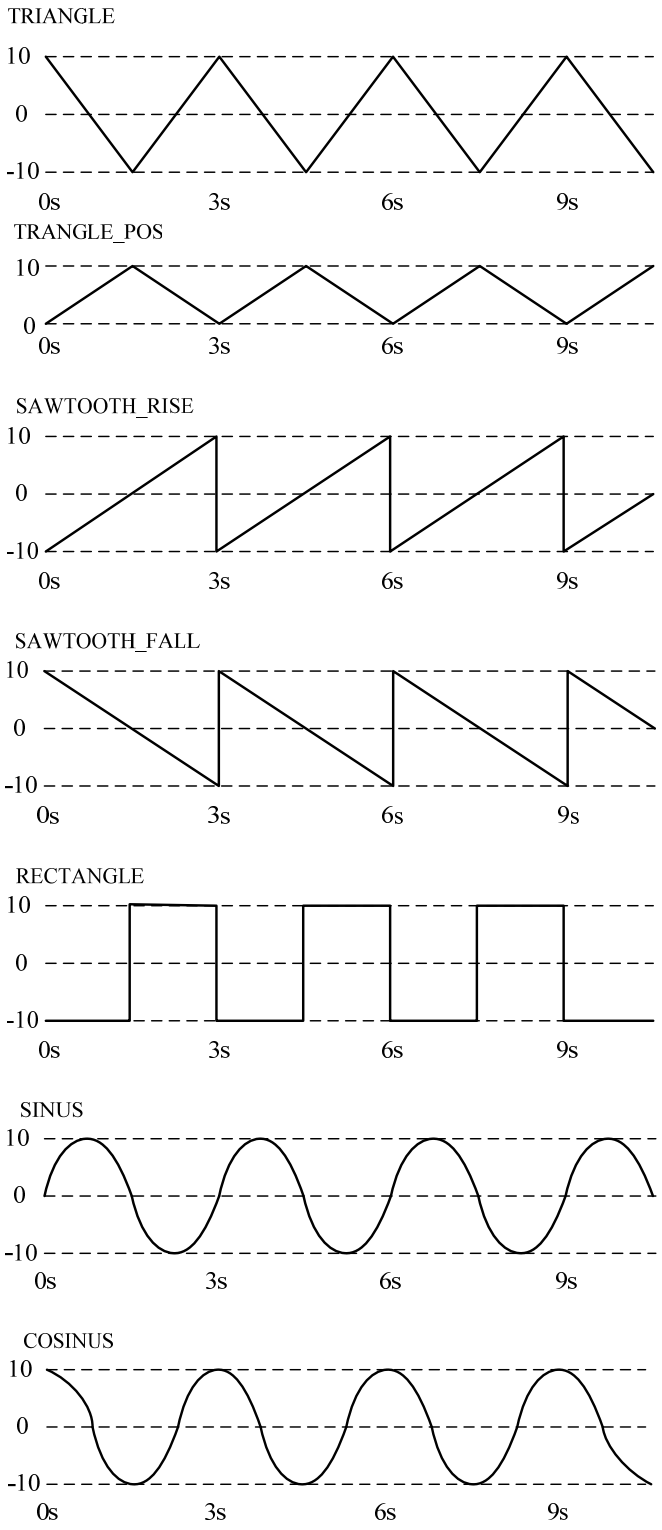


Figure 4-20-2

4.21 FUNCTION MANIPULATORS (UTIL.LIB)

4.21.1 CHARCURVE—Characteristic Curve

Function: The input POINT array P[0..N-1] maps a characteristic curve on a XY plot, where IN stands for the input value on X-axis and OUT is the output value on Y-axis.

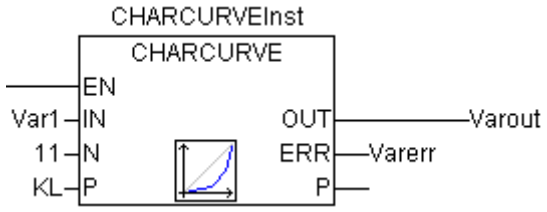
PARAMETER SPECIFICATIONS			
Input	Data Type	Function Description	Value
IN	INT	input signal on X-axis	
N	BYTE	number of points defining the characteristic curve	
P	ARRAY[0..n] OF POINT	(2≤n≤11)	describe the characteristic curve
Output	Data Type	Function Description	Value
OUT	INT	output variable	input signal on Y-axis
ERR	BYTE	display error type	ERR=1: error in ARRAY: wrong sequence ERR=2: IN outside of limits of P ERR=4: number of POINTS (N) invalid (N <2 or N >11).
P	BYTE	ARRAY of N points	

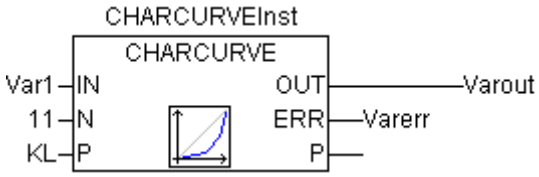
APPLICATION EXAMPLE

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT | RETAIN

	Name	Address	Type	Initial value	Comment
0001	CHARCURVEInst		CHARCURVE		
0002	Var1		INT		
0003	Varout		INT		
0004	Varerr		BYTE		
0005	KL		ARRAY[0..10] OF POINT	(X:=0,Y:=0),(X:=250,Y:=50),(X:=500,Y:=150),(X:=750,Y:=400),7((X:=1000,Y:=1000));	

Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	CHARCURVEInst(IN := Var1, N := 11, P := KL); Varerr:=CHARCURVEInst.ERR; Varout:=CHARCURVEInst.OUT;
Instruction List (IL)	CAL CHARCURVEInst(IN := Var1, N := 11, P := KL) LD CHARCURVEInst.ERR ST Varerr LD CHARCURVEInst.OUT ST Varout

Function Block Diagram (FBD)	

In the above example, when CHARCURVE instruction is executed, the output changes according to the input, as shown in figure 4-21-1. The curve is defined by array KL, IN is the input value on X-axis, and OUT is the output value on Y-axis.

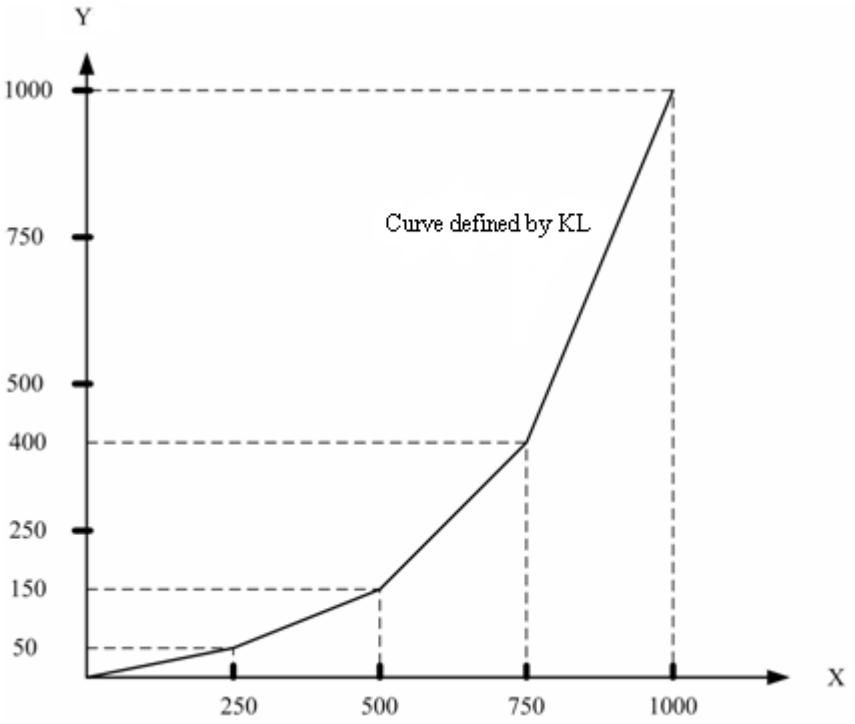


Figure 4-21-1

4.21.2 RAMP_INT—Limit the Slope of a INT Value

- Function: Limit the slope of an INT value input function.

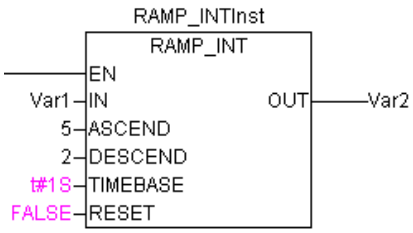
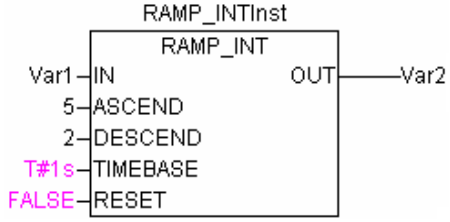
PARAMETER SPECIFICATIONS			
Input	Data Type	Function Description	Value
IN	INT	target value	If IN>previous value, ascend according to specified time and slope, and OUT. If IN<previous value, descend according to specified time and slope, and OUT.
ASCEND	INT	maximum positive slope	ascended In TIMEBASE
DESCEND	INT	maximum negative slope	descended In TIMEBASE
TIMEBASE	TIME	time base	The slope of ascend or descend
RESET	BOOL	reset	TRUE: reset RAMP_INT
Output	Data Type	Function Description	Value
OUT	INT	value of instruction with limited slope	

APPLICATION EXAMPLE

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	Name	Address	Type	Initial value	Comment
0001	RAMP_INTInst		RAMP_INT		
0002	Var1		INT		
0003	Var2		INT		

Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	RAMP_INTInst(IN := Var1, ASCEND := 5, DESCEND := 2, TIMEBASE := T#1000ms, RESET := FALSE); Var2:=RAMP_INTInst.OUT;
Instruction List (IL)	CAL RAMP_INTInst(IN := Var1, ASCEND := 5, DESCEND := 2, TIMEBASE := T#1000ms, RESET := FALSE) LD RAMP_INTInst.OUT ST Var2
Function Block Diagram (FBD)	

In the above example, when RAMP_INT is executed, the output changes according to the input, as shown in figure 4-21-2.

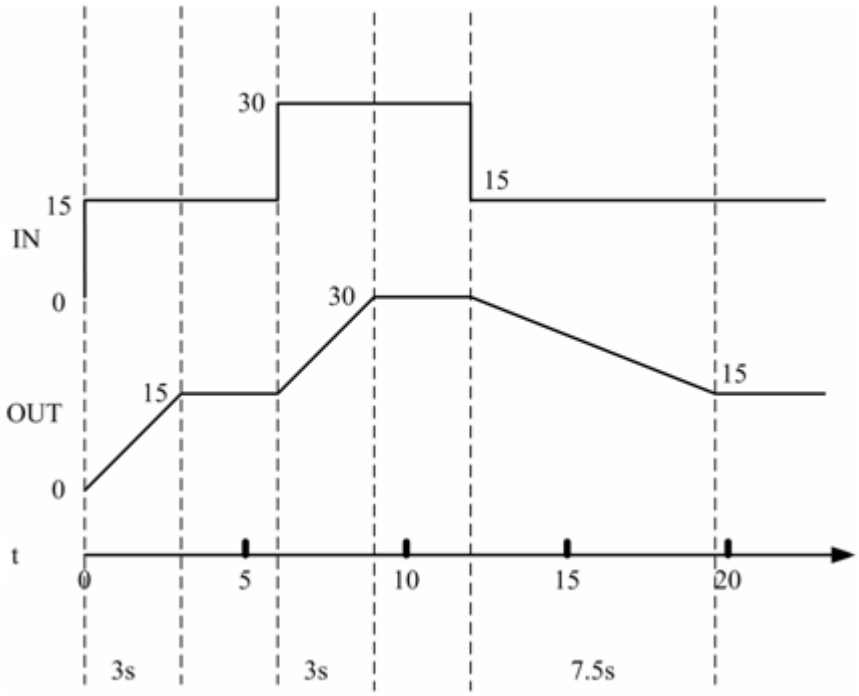


Figure 4-21-2

4.21.3 RAMP_REAL—Limit the Slope of a REAL Value

- Function: Limit the slope of a REAL value input function.

PARAMETER SPECIFICATIONS			
Input	Data Type	Function Description	Value
IN	REAL	target value	If IN>previous value, ascend according to specified time and slope, and OUT. If IN<previous value, descend according to specified time and slope, and OUT.
ASCEND	REAL	maximum positive slope	ascended In TIMEBASE
DESCEND	REAL	maximum negative slope	descended In TIMEBASE
TIMEBASE	TIME	time base	The slope of ascend or descend
RESET	BOOL	reset	TRUE: reset RAMP_INT
Output	Data Type	Function Description	Value
OUT	REAL	value of instruction with limited slope	

See RAMP_INT instruction for the detailed Application example and figures.

4.22 ANALOG MONITORS (UTIL.LIB)

4.22.1 HYSTERESIS—Hysteresis

- Function: the inputs of this instruction are 3 INT values IN, HIGH and LOW. Set OUT as TRUE when input IN is less than LOW, and set OUT as FALSE when input IN is more than HIGH, and circulate the settings.

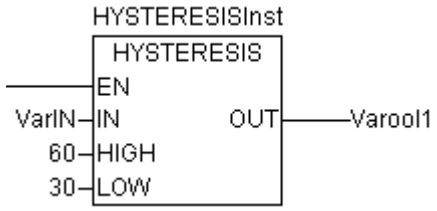
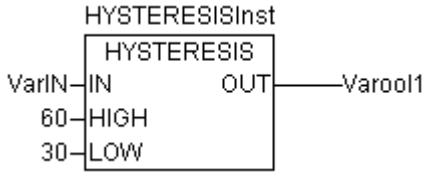
PARAMETER SPECIFICATIONS			
Input	Data Type	Function Description	Value
IN	INT	input value	
HIGH	INT	upper threshold value	
LOW	INT	lower threshold value	
Output	Data Type	Function Description	Value
OUT	BOOL	hysteresis value	Set OUT as TRUE when input IN is less than LOW, and set OUT as FALSE when input IN is bigger than HIGH.

APPLICATION EXAMPLE

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	Name	Address	Type	Initial value	Comment
0001	HYSTERESISInst		HYSTERESIS		
0002	Varool1		BOOL		
0003	VarIN		INT		

Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	HYSTERESISInst(IN := VarIN, HIGH := 60, LOW := 30); Varool1:=HYSTERESISInst.OUT;
Instruction List (IL)	CAL HYSTERESISInst(IN := VarIN, HIGH := 60, LOW := 30) LD HYSTERESISInst.OUT ST Varool1
Function Block Diagram (FBD)	

In the example, when HYSTERESIS instruction is executed, the output changes according to the input, as shown in figure 4-22-1.

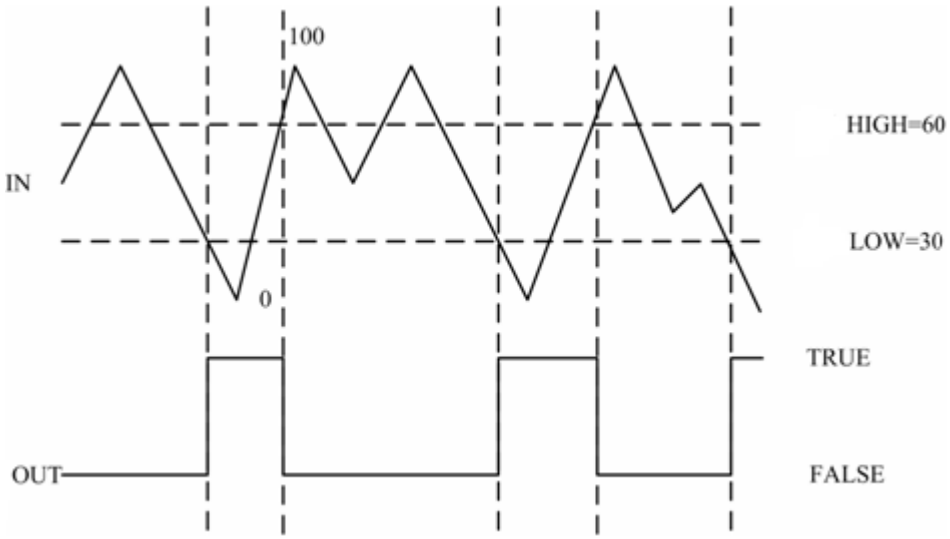


Figure 4-22-1

4.22.2 LIMITALARM—Up and Down Limit Alarm

- Function: If IN>the upper limit HIGH, then O is TRUE, U and IL are FALSE. If IN< the lower limit LOW, then U is TRUE, O and IL are FALSE. If LOW<IN<HIGH, then IL is TRUE, O and U are FALSE.

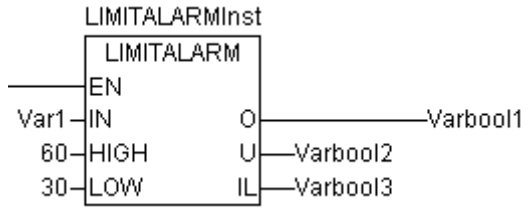
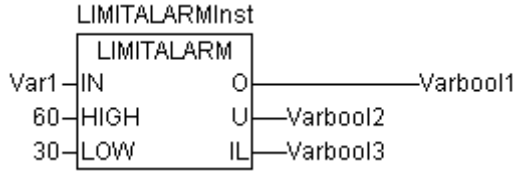
PARAMETER SPECIFICATIONS			
Input	Data Type	Function Description	Value
IN	INT	input value	
HIGH	INT	upper threshold value	
LOW	INT	lower threshold value	
Output	Data Type	Function Description	Value
O	BOOL	output value	
U	BOOL	output value	
IL	BOOL	output value	

APPLICATION EXAMPLE

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	Name	Address	Type	Initial value	Comment
0001	LIMITALARMInst		LIMITALARM		
0002	Var1		INT		
0003	Varbool1		BOOL		
0004	Varbool2		BOOL		
0005	Varbool3		BOOL		

Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	LIMITALARMInst(IN := Var1, HIGH := 60, LOW := 30); Varbool2:=LIMITALARMInst.U; Varbool3:= LIMITALARMInst.IL; Varbool1:= LIMITALARMInst.O;
Instruction List (IL)	CAL LIMITALARMInst(IN := Var1, HIGH := 60, LOW := 30) LD LIMITALARMInst.U ST Varbool2 LD LIMITALARMInst.IL ST Varbool3 LD LIMITALARMInst.O ST Varbool1
Function Block Diagram (FBD)	

In the above example, when LIMITALARM instruction is executed, the output changes according to the input, as shown in figure 4-22-2.

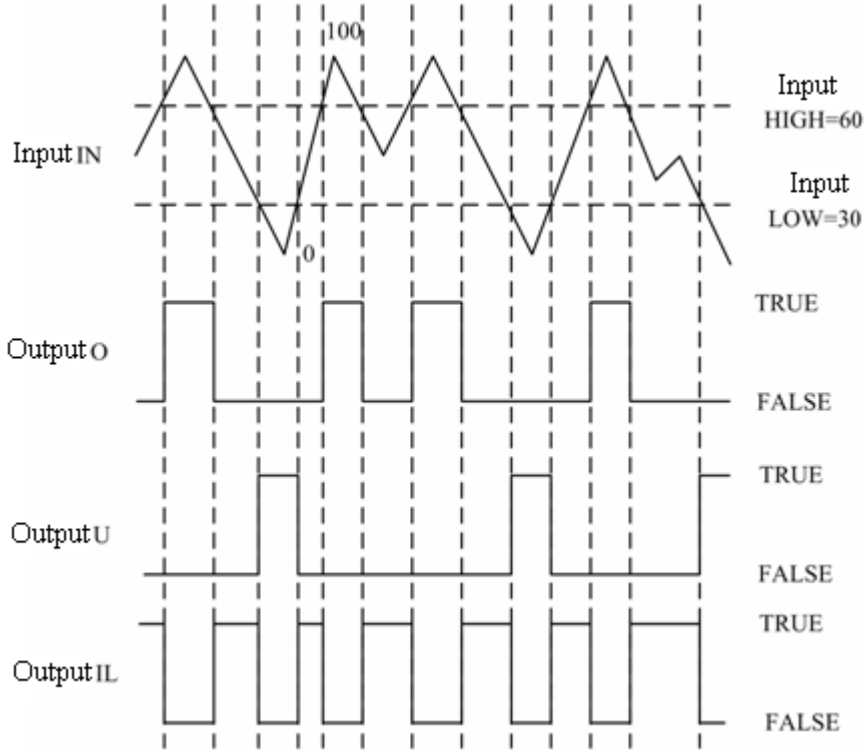


Figure 4-22-2

4.23 BI-STABLE INSTRUCTIONS (STANDARD.LIB)

4.23.1 SR—Set Dominant

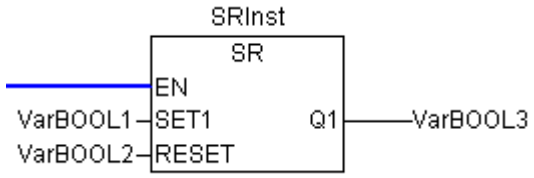
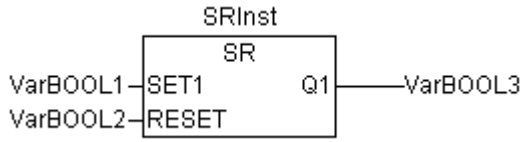
- Function: configure a Bi-stable RS flip-flop as set dominant.
  Logic relation: Q1=(NOT RESET AND Q1) OR SET1
  SET1 is the set signal, and RESET is the reset signal.
- Input/output data type: BOOL.

APPLICATION EXAMPLE

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	Name	Address	Type	Initial value	Comment
0001	SRInst		SR		
0002	VarBool1		BOOL		
0003	VarBool2		BOOL		
0004	VarBool3		BOOL		

Programming Language	Program
Ladder Diagram (LD)	 Comments:VarBOOL3＝ (NOT VarBOOL2 AND VarBOOL3)OR VarBOOL1
Structured Text (ST)	SRInst(SET1:= VarBOOL1 , RESET:=VarBOOL2); VarBOOL3 := SRInst.Q1 ;
Instruction List (IL)	CALSRInst(SET1:= VarBOOL1, RESET:= VarBOOL2) LD SRInst.Q1 ST VarBOOL3
Function Block Diagram (FBD)	

Instruction	SET1	RESET	Output
SR	0	0	Retain
	1	0	1
	0	1	0
	1	1	1

Table of the Instruction True-False Value

4.23.2 RS—Reset Dominant

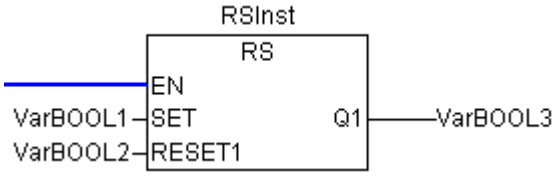
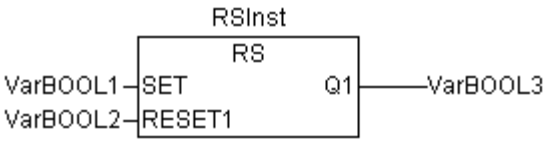
- Function: Configure a Bi-stable RS flip-flop as reset dominant.
  Logic relation: Q1=NOT RESET1 AND (Q1 OR SET)
  SET is the set signal, and RESET1 is the reset signal.
- Input/output data type: BOOL.

APPLICATION EXAMPLE

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	Name	Address	Type	Initial value	Comment
0001	SRInst		SR		
0002	VarBool1		BOOL		
0003	VarBool2		BOOL		
0004	VarBool3		BOOL		

Programming Language	Program
Ladder Diagram (LD)	 Comments: VarBOOL3＝NOT VarBOOL2 AND (VarBOOL3 OR VarBOOL1)
Structured Text (ST)	RSInst(SET:= VarBOOL1 , RESET1:=VarBOOL2); VarBOOL3 := RSInst.Q1 ;
Instruction List (IL)	CAL RSInst(SET := VarBOOL1, RESET1 := VarBOOL2) LD RSInst.Q1 ST VarBOOL3
Function Block Diagram (FBD)	

Instruction	SET1	RESET	Output
RS	0	0	Retain
	1	0	1
	0	1	0
	1	1	1

Table of the Instruction True-False Values

4.24 TRIGGERS (STANDARD.LIB)

Triggers include the rising edge detecting R_TRIG and the falling edge detecting F_TRIG.

4.24.1 R_TRIG—Rising Edge Detection

- Function: Rising edge detection.
- Logic relationship:

```
Q := CLK AND NOT M;
M := CLK;
```

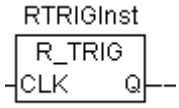
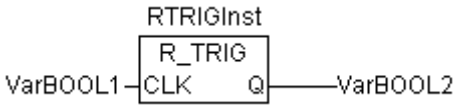
- M is a middle variable with the initial value TRUE, Q and M are FALSE as long as CLK is FALSE. When the R_TRIG instruction is called, Q will return FALSE. When CLK detects the rising edge, then Q will return returns TRUE.
- Input/output data type: CLK, Q: BOOL.

APPLICATION EXAMPLE

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	Name	Address	Type	Initial value	Comment
0001	RTRIGInst		R_TRIG		
0002	VarBOOL1		BOOL		
0003	VarBOOL2		BOOL		

Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	RTRIGInst(CLK:= VarBOOL1); VarBOOL2 := RTRIGInst.Q;
Instruction List (IL)	CAL RTRIGInst(CLK := VarBOOL1) LD RTRIGInst.Q ST VarBOOL2
Function Block Diagram (FBD)	

4.24.2 F_TRIG—Falling Edge Detection

- Function: Falling edge detection.
- Logic relationship:

Q := NOT CLK AND NOT M;
M := NOT CLK;

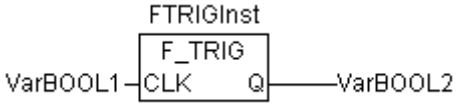
- M is a middle variable with the initial value FALSE, Q and M retain FALSE as long as CLK is TRUE. When CLK is FALSE, Q returns TRUE at the first time, and M is set as TRUE. When F_TRIG is called, Q will return FALSE. When CLK detects the falling edge, Q will return TRUE.
- Input/output data type: CLK, Q: BOOL.

APPLICATION EXAMPLE

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	Name	Address	Type	Initial value	Comment
0001	FTRIGInst		F_TRIG		
0002	VarBOOL1		BOOL		
0003	VarBOOL2		BOOL		

Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	FTRIGInst(CLK:= VarBOOL1); VarBOOL2 := FTRIGInst.Q;
Instruction List (IL)	CAL FTRIGInst(CLK := VarBOOL1) LD FTRIGInst.Q ST VarBOOL2
Function Block Diagram (FBD)	

4.25 COUNTERS (STANDARD.LIB)

Counters include CTU (counter up), CTD (counter down) and CTUD (counter up down).

4.25.1 CTU—Counter Up

- Function: Counter up instruction.

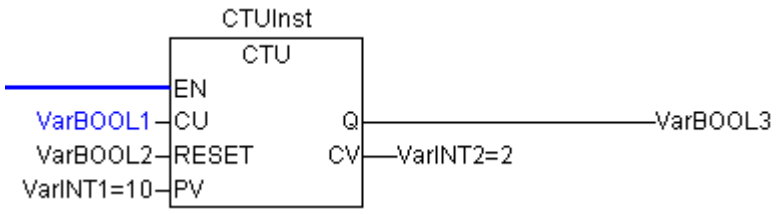
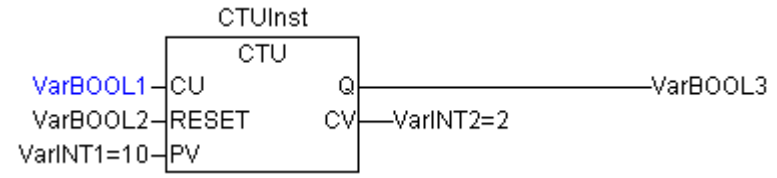
PARAMETER SPECIFICATIONS			
Input	Data Types	Function Description	Value
CU	BOOL	counter up	CV is increased by 1 if CU has a rising edge from FALSE to TRUE.
RESET	BOOL	reset	Reset Counter to 0 when TRUE.
PV	WORD	counter limit	0-65535
Output	Data Types	Function Description	Value
Q	BOOL	counter reached the limit	When CV>=PV, Q is TRUE.
CV	WORD	current counter value	

APPLICATION EXAMPLE

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	Name	Address	Type	Initial value	Comment
0001	CTUInst		CTU		
0002	VarBOOL1		BOOL		
0003	VarBOOL2		BOOL		
0004	VarBOOL3		BOOL		
0005	VarINT1		INT		
0006	VarINT2		INT		

Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	CTUInst(CU:= VarBOOL1, RESET:=VarBOOL2 , PV:= VarINT1); VarBOOL3 := CTUInst.Q; VarINT2 := CTUInst.CV;
Instruction List (IL)	CAL CTUInst(CU := VarBOOL1, RESET := VarBOOL2, PV :=VarINT1) LD CTUInst.Q ST VarBOOL3 LD CTUInst.CV ST VarINT2
Function Block Diagram (FBD)	

4.25.2 CTD—Counter Down

- Function: Counter down instruction.

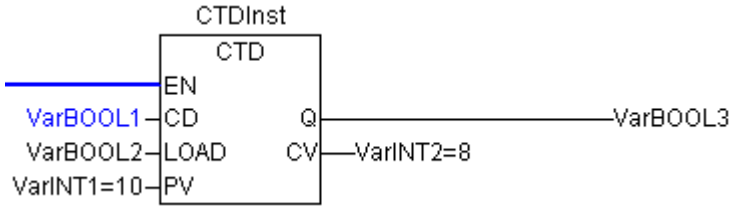
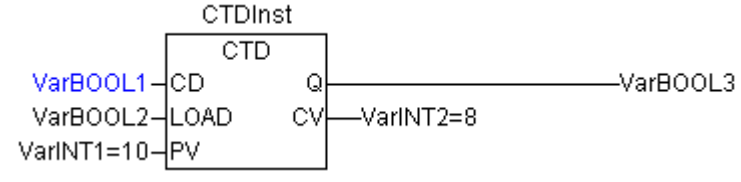
PARAMETER SPECIFICATIONS			
Input	Data Types	Function Description	Value
CD	BOOL	counter down	CV (>0) is decremented by 1 if CD has a rising edge from FALSE to TRUE.
LOAD	BOOL	reset	When LOAD is TRUE, CV reaches PV.
PV	WORD	start value	0-65535
Output	Data Types	Function Description	Value
Q	BOOL	counter reached 0	When CV is 0, Q is TRUE.
CV	WORD	current valeu	

APPLICATION EXAMPLE

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	Name	Address	Type	Initial value	Comment
0001	CTUInst		CTU		
0002	VarBOOL1		BOOL		
0003	VarBOOL2		BOOL		
0004	VarBOOL3		BOOL		
0005	VarINT1		INT		
0006	VarINT2		INT		

Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	CTDInst(CD:= VarBOOL1, LOAD:=VarBOOL2 , PV:= VarINT1); VarBOOL3 := CTDInst.Q; VarINT2 := CTDInst.CV;
Instruction List (IL)	CAL CTDInst(CD := VarBOOL1, LOAD := VarBOOL2, PV :=VarINT1) LD CTDInst.Q ST VarBOOL3 LD CTDInst.CV ST VarINT2
Function Block Diagram (FBD)	

4.25.3 CTUD—Counter Up Down

- Function: Counter up down instruction

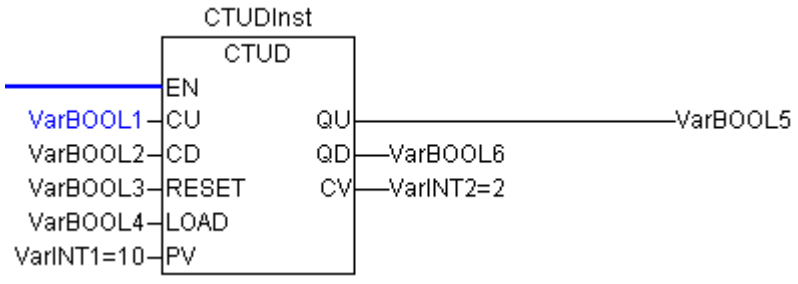
PARAMETER SPECIFICATIONS			
Input	Data Types	Function Description	Value
CU	BOOL	counter up	CV is incremented by 1 if CU has a rising edge from FALSE to TRUE.
CD	BOOL	counter down	CV (>0) is decremented by 1 if CD has a rising edge from FALSE to TRUE.
RESET	BOOL	reset	When RESET is TRUE, reset CV to 0.
LOAD	BOOL	load start value	When LOAD is TRUE, CV reaches PV.
PV	WORD	start value	0-65535
Output	Data Types	Function Description	Value
QU	BOOL	counter reached Limit	When CV is PV, QU is TRUE.
QD	BOOL	counter reached Null	When CV is 0, QD is TRUE.
CV	WORD	current Counter Value	

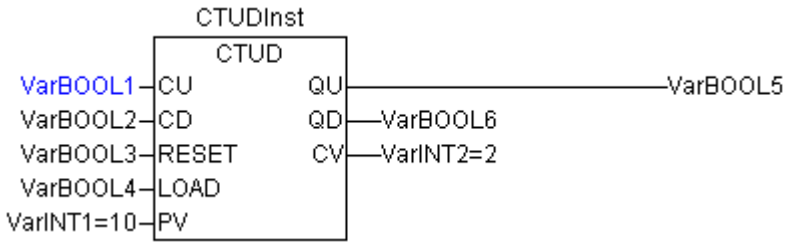
APPLICATION EXAMPLE

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	Name	Address	Type	Initial value	Comment
0001	CTUDInst		CUTD		
0002	VarBOOL1		BOOL		
0003	VarBOOL2		BOOL		
0004	VarBOOL3		BOOL		
0005	VarBOOL3		BOOL		
0006	VarBOOL4		BOOL		
0007	VarBOOL5		BOOL		
0008	VarBOOL6		BOOL		
0009	VarINT1		INT		
0010	VarINT2		INT		

Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	CTUDInst(CU:=VarBOOL1,CD:=VarBOOL2,RESET:=VarBOOL3, LOAD:=VarBOOL4,PV:=VarINT1) VarBOOL5 := CTUDInst.QU

	VarBOOL6 := CTUDInst.QD VarINT2 := CTUDInst.CV
Instruction List (IL)	CAL CTUDInst(CU:=VarBOOL1,CD:=VarBOOL2, RESET:=VarBOOL3,LOAD:=VarBOOL4,PV:=VarINT1) LD CTUDInst.QU ST VarBOOL5 LD CTUDInst.QD ST VarBOOL6 LD CTUDInst.CV ST VarINT2
Function Block Diagram (FBD)	

4.26 TIMER (STANDARD.LIB)

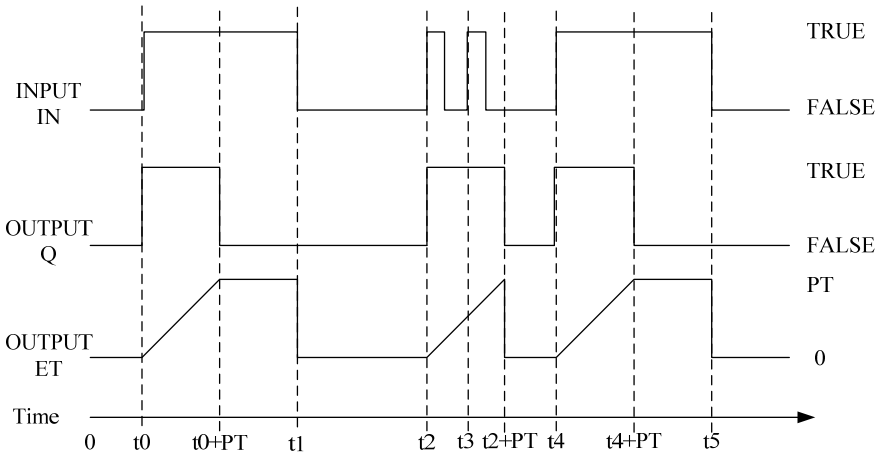
Timers include TP (Pulse Timer), TON (On-delay Timer), TOF (Off-delay Timer), RTC (Real Time Clock).

4.26.1 TP—Timer Pulse

- Function: Pulse timer instruction.

PARAMETER SPECIFICATIONS			
Input	Data Types	Function Description	Value
IN	BOOL	trigger for Start of the Signal	When IN is TRUE, ET times in millisecond until ET reaches PT, and then ET retains.
PT	TIME	the length of the High-Signal in 10ms	
Output	Data Types	Function Description	Value
Q	BOOL	the pulse	If IN is FALSE, then Q is FALSE and ET is 0. When IN is TRUE, TP starts to work, Q is TRUE, and IN is invalid before ET<=PT. Q is FALSE when ET reaches PT.
ET	TIME	the current value	When IN changes to TRUE, ET times in millisecond until ET reaches PT and then ET retains. After timing, IN is FALSE and ET equals 0.

TP Timing Diagram

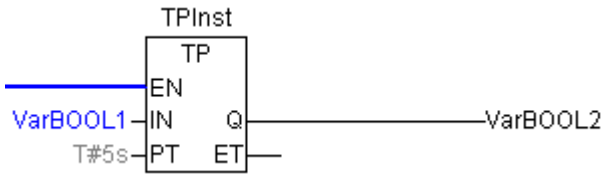
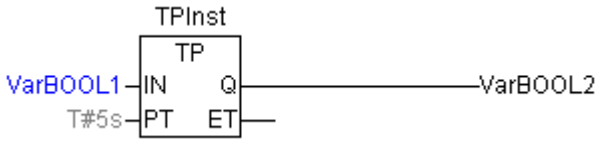


APPLICATION EXAMPLE

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	Name	Address	Type	Initial value	Comment
0001	TPInst		TP		
0002	VarBOOL1		BOOL		
0003	VarBOOL2		BOOL		

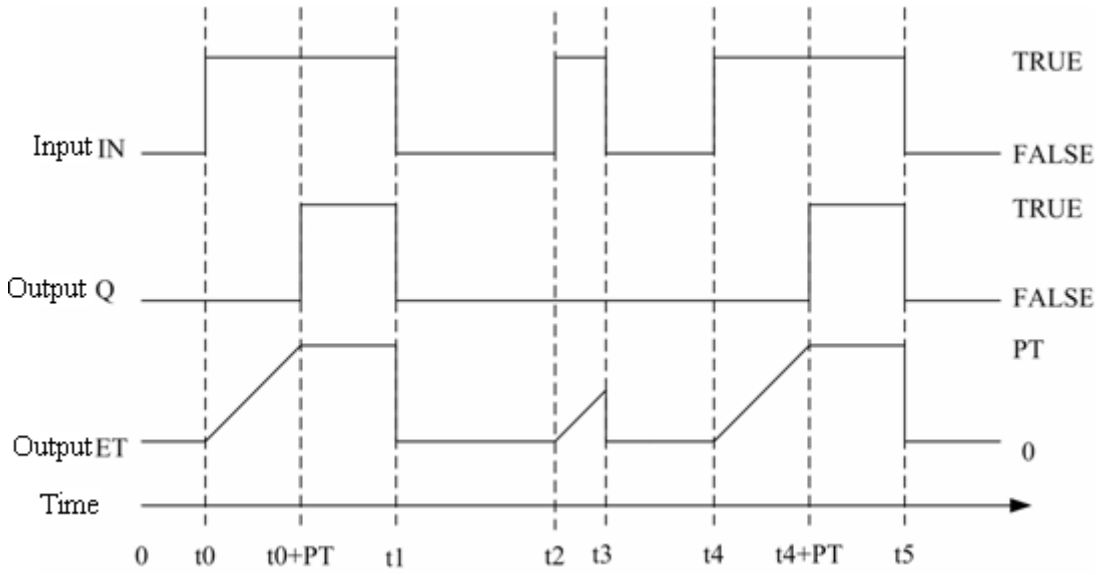
Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	TPInst(IN:= VarBOOL1,PT:= T#5s); VarBOOL2:=TPInst.Q;
Instruction List (IL)	CAL TPInst(IN:= VarBOOL1,PT:=T#5s) LD TPInst.Q ST VarBOOL2
Function Block Diagram (FBD)	

4.26.2 TON—On-delay Timer

- Function: On-delay timer instruction.

PARAMETER SPECIFICATIONS			
Input	Data Type	Function Description	Value
IN	BOOL	starts timer with rising edge	When IN is TRUE, ET times in millisecond until ET reaches PT and then ET retains.
PT	TIME	time to pass, before Q is set	
Output	Data Type	Function Description	Value
Q	BOOL	timer out	When IN is FALSE, Q is FALSE and ET is 0. When IN is TRUE, TON starts to work, when ET reaches PT, Q is TRUE.
ET	TIME	current time value	When IN is TRUE, ET times in millisecond until ET reaches PT and then ET retains. Whenever IN is FALSE, ET equals to 0.

TON Timing Diagram:

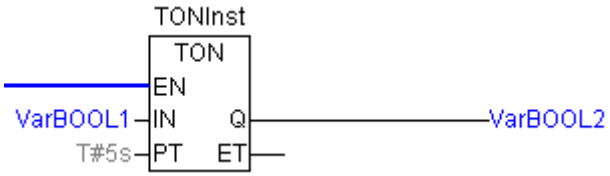
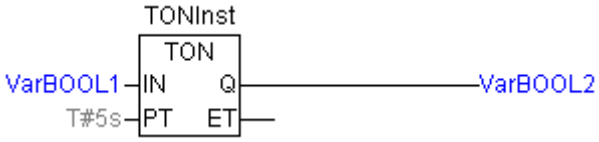


APPLICATION EXAMPLE

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	Name	Address	Type	Initial value	Comment
0001	TONInst		TON		
0002	VarBOOL1		BOOL		
0003	VarBOOL2		BOOL		

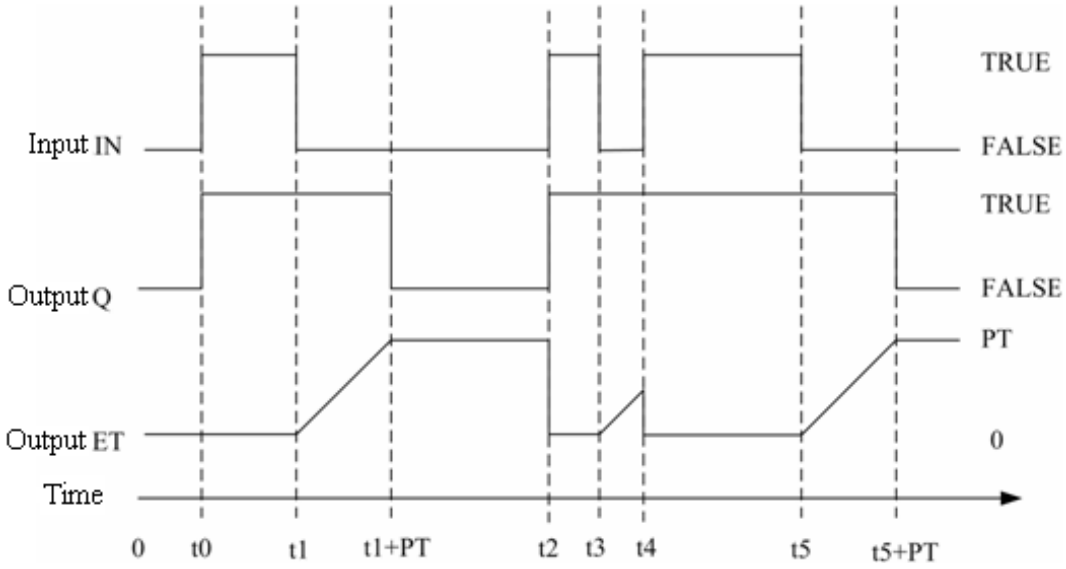
Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	TONInst(IN:=VarBOOL1,PT:= T#5s);
Instruction List (IL)	CAL TONInst(IN:= VarBOOL1,PT:=T#5s) LD TONInst.Q ST VarBOOL2
Function Block Diagram (FBD)	

4.26.3 TOF—Off-delay Timer

- Function: Off-delay timer instruction.

PARAMETER SPECIFICATIONS			
Input	Data Type	Function Description	Value
IN	BOOL	starts timer with falling edge	When IN is from TRUE to FALSE, ET times in millisecond until ET reaches PT and then ET retains.
PT	TIME	time to pass, before Q is set	
Output	Data Type	Function Description	Value
Q	BOOL	timer out	When IN is FALSE and ET equals to PT, Q is FALSE, otherwise Q is TRUE.
ET	TIME	current time value	When IN is FALSE, ET times in millisecond until ET reaches PT and then ET retains. Whenever IN is TRUE, ET equals to 0, Q is TRUE.

TOF Timing Diagram:

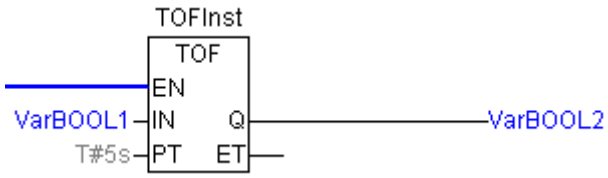
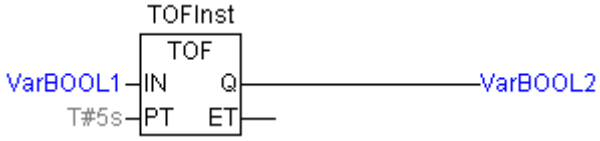


APPLICATION EXAMPLE

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	Name	Address	Type	Initial value	Comment
0001	TOFInst		TOF		
0002	VarBOOL1		BOOL		
0003	VarBOOL2		BOOL		

Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	TOFInst(IN:= VarBOOL1,PT:=T#5s); VarBOOL2 :=TOFInst.Q;
Instruction List (IL)	CAL TOFInst(IN:= VarBOOL1,PT:= T#5s) LD TOFInst.Q ST VarBOOL2
Function Block Diagram (FBD)	

4.26.4 RTC—Real Time Clock

- Function: Start at a given time in PDT, and return the current date and time.

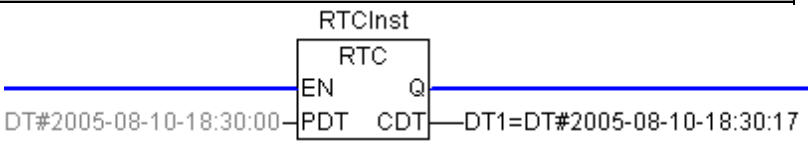
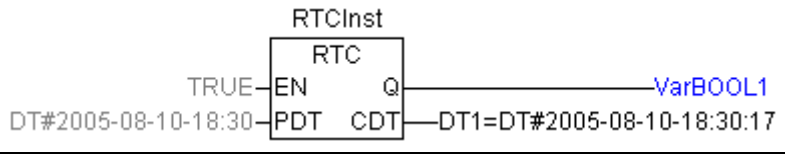
PARAMETER SPECIFICATIONS			
Input	Data Type	Function Description	Value
EN	BOOL	enable signal	start the RTC
PDT	DT	time base	
Output	Data Type	Function Description	Value
Q	BOOL	timer out	When EN is FALSE, Q is FALSE; When EN is TRUE, Q is TRUE.
CDT	DT	current time value	When EN is FALSE, CDT is 1970-01-01-00:00:00. When EN is TRUE, CDT starts at the time in PDT.

APPLICATION EXAMPLE

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	Name	Address	Type	Initial value	Comment
0001	RTCInst		RTC		
0002	DT1		DT		
0003	VarBOOL1		BOOL		

Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	RTCInst(PDT:=DT#2005-08-10-18:30:31); DT1:=RTCInst.CDT; VarBOOL1:=RTCInst.Q;
Instruction List (IL)	CAL RTCInst(PDT := DT#2005-08-10-18:30:31) LD RTCInst.CDT ST DT1 LD RTCInst.Q ST VarBOOL1
Function Block Diagram (FBD)	

Chapter 5

Expansion Instructions

Expansion instructions includes other various sets of instructions includes external interrupt instructions, pulse outputs, bi-phase counters and the instructions related to hardware ports, etc. All the expansion instructions are presented as function blocks, and the names of their libraries are all started with “HollySys”, *see appendix A.1 for details.*

5.1 ANALOG MODULES (HOLLYSYS_PLC_ANALOG.LIB)

5.1.1 Analog_IN—Call Analog_In Modules

Example of Ladder Diagram Language

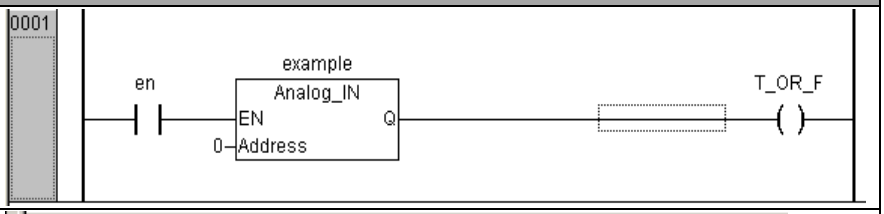

PARAMETER SPECIFICATIONS			
Storage Address Of Input Data	Meaning		
%IWxx	Analog input channels correspond to addresses of I area in PLC configuration.		
Input	Data Type	Function Description	Value
EN	BOOL	enable	0: Disabled, can not scan Analog_IN module. 1: Enabled, can scan Analog_IN module.
Address	BYTE	module node id	0-7(consistent with node id in PLC configuration)
Output	Data Type	Function Description	Value
Q	BOOL	Indicates if valid data has been read	0：data not read 1：valid data read

APPLICATION EXAMPLE (LD and ST)

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT | RETAIN

	Name	Address	Type	Initial value	Comment
0001	T_OR_F		BOOL		
0002	en		BOOL		
0003	example		Analog_IN		

Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	0001 example(EN:= en,Address:=0); 0002 T_OR_F:=example.Q;

Input value of Address in Application example should be consistent with the node id in PLC configuration, as shown in figure 5-1-1.

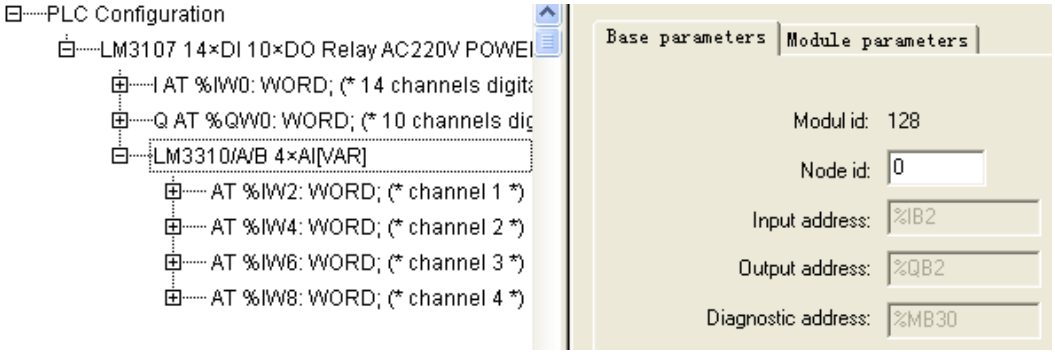


Figure 5-1-1

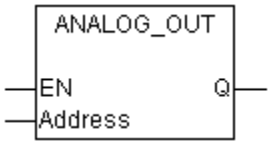
Remarks on the program:

- When EN is set and retained, Analog_IN modules are scanned. As shown in figure 5-1-1, the scanned values of four channels are stored in registers %IW2, %IW4, %IW6, and %IW8 respectively.
- When EN is reset, Analog_IN modules are not scanned.

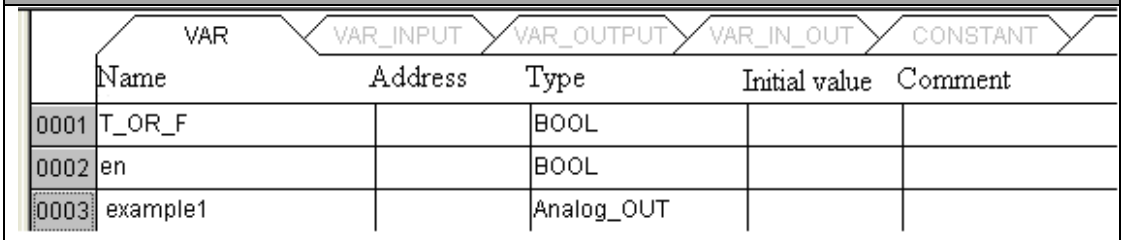
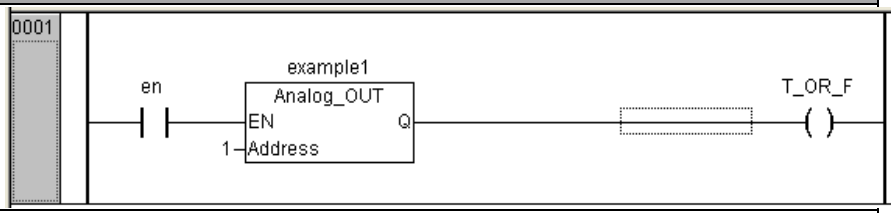
Note:

- If multiple Analog_IN modules (including LM3310, LM3311, LM3312, LM3313 and LM3314) shall be used, it is necessary to configure multiple Analog_IN instructions and the address of each Analog_IN must be consistent with the corresponding module node id.
- Before calling Analog_IN, PLC must be configured (as shown in figure 5-1-1) and Hollysys_PLC_Analog.lib must be added.

5.1.2 Analog_OUT—Call Analog_OUT Modules

Example of Ladder Diagram Language


PARAMETER SPECIFICATIONS			
Storage Address Of Output Data	Meaning		
%QWxx	Analog output channels correspond to addresses of Q area in PLC configuration.		
Input	Data Type	Function Description	Value
EN	BOOL	enable	0: disabled, can not scan Analog_OUT module. 1: enabled, can scan Analog_ OUT module.
Address	BYTE	module node id	0-7(consistent with node id in PLC configuration)
Output	Data Type	Function Description	Value
Q	BOOL	Indicates if valid data has been output	0: no output data 1: correctly output data

APPLICATION EXAMPLE (LD and ST)	
Variable Declaration	
	
Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	0002 example1(en:=en,address:=1); 0003 T_OR_F:=example1.Q;

Input value of Address in Application example should be consistent with the node id in PLC configuration in figure 5-1-2.

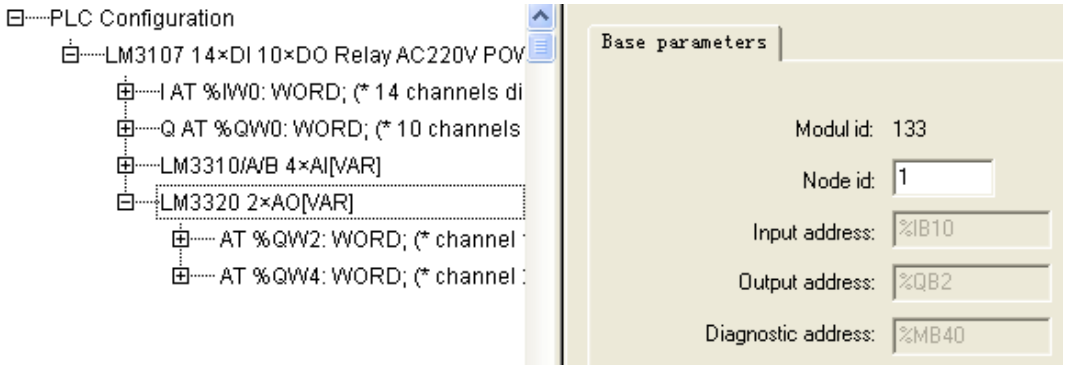


Figure 5-1-2

Remarks on the program:

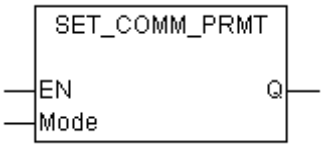
- When EN is set and retains, Analog_OUT modules are scanned. As shown in figure 5-1-2, the scanned values of four channels are stored in registers %QW2, and %QW4 respectively.
- When EN is reset, Analog_OUT modules are not scanned.

Note

- If multiple Analog_OUT modules shall be used, it is necessary to configure multiple Analog_ OUT instructions and the address of each Analog_ OUT must be consistent with corresponding module node id.
- Before calling Analog_ OUT, PLC must be configured (shown in figure 5-1-2) and Hollysys_PLC_Analog.lib must be added.

5.2 RS232 FREE PORT COMMUNICATION SETTING (HOLLYSYS_PLC_COMM.LIB)

5.2.1 Set_COMM_PRMT— Set RS232 Free Port Communication Parameters

Example of Ladder Diagram Language


PARAMETER SPECIFICATIONS			
Input	Data Type	Function Description	Value
EN	BOOL	enable	0: disabled 1: rising edge enabled
Mode	BYTE	communication modes (7 bits no parity is not supported, default is 7 bits even parity)	Bits 7 6 5 4 3 2 1 0 Parity / number of bits / baud rate / not defined See the following Table of Mode Parameter Specifications.
Output	Data Type	Function Description	Value
Q	BOOL	flag of setting completion	0: setting not completed 1: setting completed

TABLE OF MODE PARAMETER								
Parity		No. of Bits	Baud Rate			Undefined		Comments
7	6	5	4	3	2	1	0	
0	0							keep current communication parameters (startup free port)
0	1							even
1	0							no parity (7 bits no parity is not supported; default to 7 bits even parity)
1	1							Odd parity
		0						8 bits
		1						7 bits
			0	0	0			38400bps
			0	0	1			19200bps
			0	1	0			9600bps
			0	1	1			4800bps
			1	0	0			2400bps
			1	0	1			1200bps
			1	1	0			600bps
			1	1	1			300bps

Note:

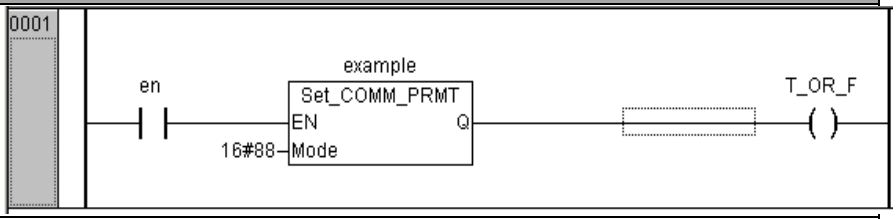
- After using Set_COMM_PRMT to set free port parameters, the RUN/STOP switch needs to be turned to the STOP position before log in and download/debug again.

APPLICATION EXAMPLE (LD and ST)

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	Name	Address	Type	Initial value	Comment
0001	T_OR_F		BOOL		
0002	en		BOOL		
0003	example		Set_COMM_PRMT		

Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	0002 example(EN:=en, Mode:=16#88); 0003 T_OR_F:=example.Q;

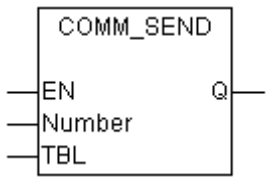
As shown in the table below, when the Mode is 16#88(2#10001000), the free port is set to no parity, 8 bits communication with baud rate of 9600bps.

Mode=16#88							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Parity		Bit number	Baud rate			No definition	
1	0	0	0	1	0	0	0

Remarks on the program:

- When EN is set, RS232 is set as a free port of no parity, 8 bits and 9600bps.
- When EN is reset, the original setting is retained.

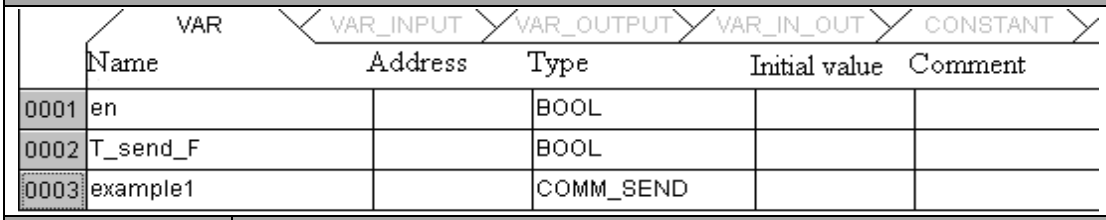
5.2.2 COMM_SEND—Send RS232 Free Port Communication Data

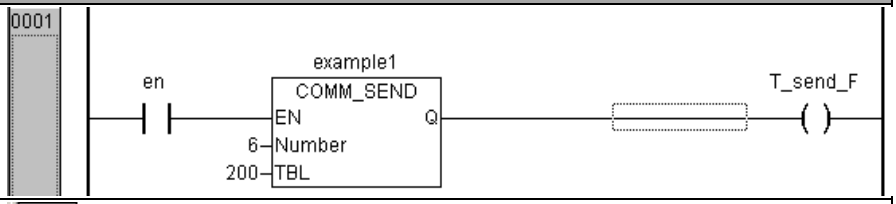
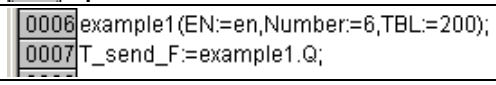
Example of Ladder Diagram Language


PARAMETER SPECIFICATIONS			
Input	Data Type	Function Description	Value
EN	BOOL	enable	0: disabled 1: rising edge enabled
Number	BYTE	bytes sent	
TBL	INT	first address of send buffer in M area	For: TBL=X, First address is %MBX
Output	Data Type	Function Description	Value
Q	BOOL	indicate if sending is completed	0: sending not completed 1: sending completed

APPLICATION EXAMPLE (LD and ST)

Variable Declaration



Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	

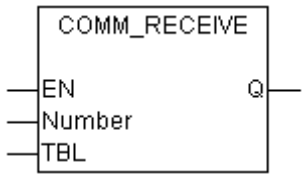

```
0006 example1(EN:=en,Number:=6,TBL:=200);
0007 T_send_F:=example1.Q;
```

Remarks on the program:

- Under the RS232 free port mode， when EN is set and retained， 6 consecutive bytes starting from %MB200 (%MB200-%MB205) are sent, and Q equals TRUE.
- When EN is reset, no data will be sent.

5.2.3 COMM_RECEIVE—Receive RS232 Free Port Communication Data

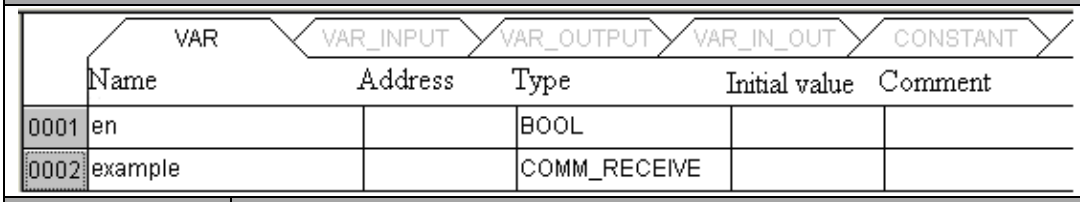
Example of Ladder Diagram Language

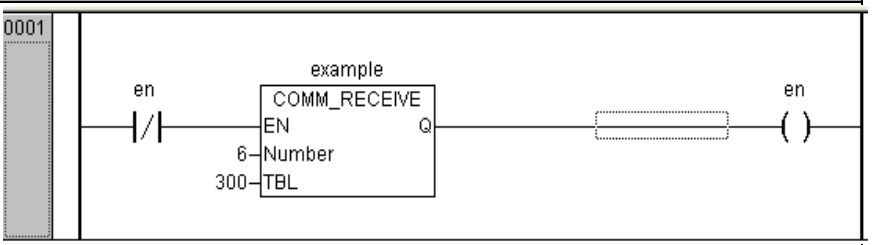
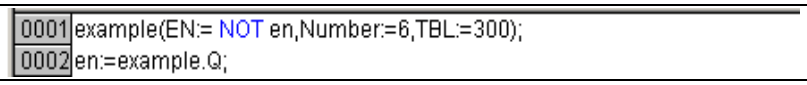


PARAMETER SPECIFICATIONS			
Input	Data Type	Function Description	Value
EN	BOOL	enable	0: disabled 1: rising edge enabled
Number	BYTE	bytes received	
TBL	INT	first address of receive buffer in M area	For: TBL=X, first address is %MBX
Output	Data Type	Function Description	Value
Q	BOOL	indicate if receiving is completed	0: receiving not completed 1: receiving completed

APPLICATION EXAMPLE (LD and ST)

Variable Declaration



Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	

Remarks on the program:

- Under the RS232 free port mode, when EN equals to FALSE at power-on, in the first scanning period 6 bytes of received data are stored in %MB300-%MB305 respectively, and Q is TRUE, which makes EN to be TRUE. In the second scanning period, no data is received, and Q is FALSE, which makes EN to be FALSE. In the third scanning period 6 bytes of received data are stored in %MB300-%MB305 respectively. Data are received in this cyclic manner.

5.2.4 Reset_COMM_PRMT—Reset RS232 Settings

Example of Ladder Diagram Language


PARAMETER SPECIFICATIONS			
Input	Data Type	Function Description	Value
EN	BOOL	enable	0: disabled 1: rising edge enabled
Mode	BYTE	communication mode (7 bits no parity is not supported, default is 7 bits even parity)	Bits 7 6 5 4 3 2 1 0 parity / number of bits / baud rate / not defined See the following Table of Mode Parameter Specifications .
Output	Data Type	Function Description	Value
Q	BOOL	indicate if setting is completed	0: setting not completed 1: setting completed

TABLE OF MODE PARAMETER								
Parity		No. of Bits	Baud Rate			Not Defined		Comments
7	6	5	4	3	2	1	0	
0	0							keep current communication parameters(reset LM special protocol)
0	1							even parity
1	0							no parity (7 bits with no parity is not supported; default to 7 bits with even parity)
1	1							Odd parity
		0						8 bits
		1						7 bits
			0	0	0			38400bps
			0	0	1			19200bps
			0	1	0			9600bps
			0	1	1			4800bps
			1	0	0			2400bps
			1	0	1			1200bps
			1	1	0			600bps
			1	1	1			300bps

Note:

- The default protocol of RS232 is MODBUS and LM proprietary protocol, after a port is set as the free port by using Set_COMM_PRMT, Reset_COMM_PRMT shall be used to reset LM proprietary protocol and change the communication parameters at the same time.
- Communication parameters of LM proprietary protocol are no parity, 8 bits, 38400bps (Mode value is 16#80).

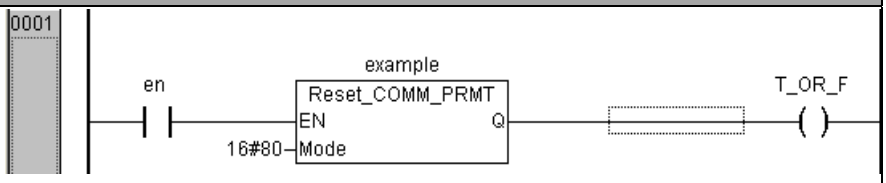
Application Example (LD and ST)

APPLICATION EXAMPLE (LD and ST)

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	Name	Address	Type	Initial value	Comment
0001	en		BOOL		
0002	example		Reset_COMM_PRM		
0003	T_OR_F		BOOL		

Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	0001 example(EN:=en,Mode:=16#80); 0002 T_OR_F:=example.Q;

As shown in the table below, when the Mode is 16#80(2#10000000), LM proprietary protocol is no parity, 8 bits and 38400bps.

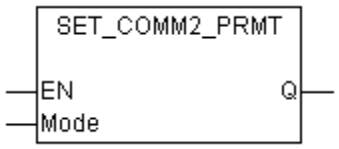
Mode=16#80							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Parity		Number of Bits	Baud Rate			Not Defined	
1	0	0	0	0	0	0	0

Remarks on the program

- When EN is set and retained, RS232 is set as LM proprietary protocol mode with no parity, 8 bits and 38400bps.

5.3 RS485 FREE PORT COMMUNICATION SETTING (HOLLYSYS_PLC_COMM2.LIB)

5.3.1 Set_COMM2_PRMT— Set RS485 Free Port Communication Parameters

Example of Ladder Diagram Language


PARAMETER SPECIFICATIONS			
Input	Data Type	Function Description	Value
EN	BOOL	enable	0: disabled 1: rising edge enabled
Mode	BYTE	communication modes	Bits 7 6 5 4 3 2 1 0 parity | number of bits | baud rate | not defined See the following Table of Mode Parameter Specifications.
Output	Data Type	Function Description	Value
Q	BOOL	indicate if setting is completed	0: setting not completed 1: setting completed

TABLE OF MODE PARAMETER								
Parity		Number of Bits	Baud Rate			Not Defined		Comments
7	6	5	4	3	2	1	0	
0	0							keep current communication parameters(startup free port)
0	1							even
1	0							no parity (7 bits no parity is not supported; default to 7 bits even parity)
1	1							odd parity
		0						8 bits
		1						7 bits
			0	0	0			38400bps
			0	0	1			19200bps
			0	1	0			9600bps
			0	1	1			4800bps
			1	0	0			2400bps
			1	0	1			1200bps
			1	1	0			600bps
			1	1	1			300bps

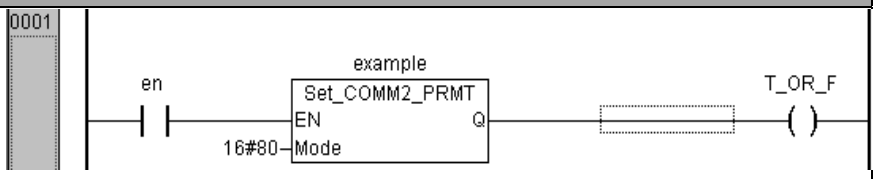
Application Example (LD and ST)

APPLICATION EXAMPLE (LD and ST)

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	Name	Address	Type	Initial value	Comment
0001	en		BOOL		
0002	example		Set_COMM2_PRMT		
0003	T_OR_F		BOOL		

Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	0001 example(EN:=en,Mode:=16#80); 0002 T_OR_F:=example.Q;

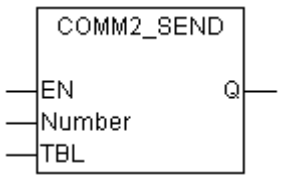
As shown in the table below, when Mode is 16#80(2#10000000), the free port is no parity, 8 bits and 38400bps.

Mode=16#80							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Parity		Bit number	Baud rate			No defination	
1	0	0	0	0	0	0	0

Remarks on the program:

- When EN is set and retained, RS485 is set as a free port with no parity, 8 bits and 38400bps.
- When EN is reset, the original setting is retained.

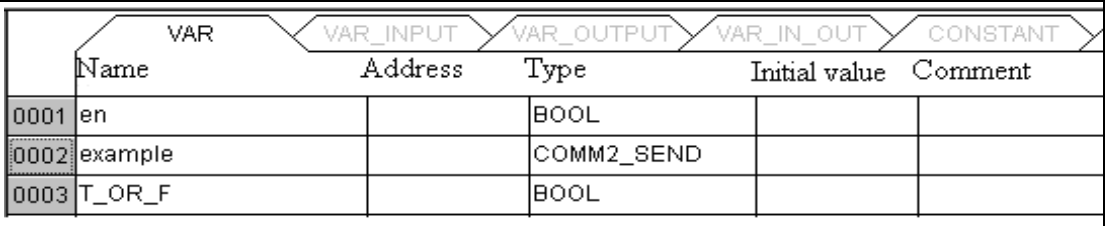
5.3.2 COMM2_SEND—Send RS485 Free Port Communication Data

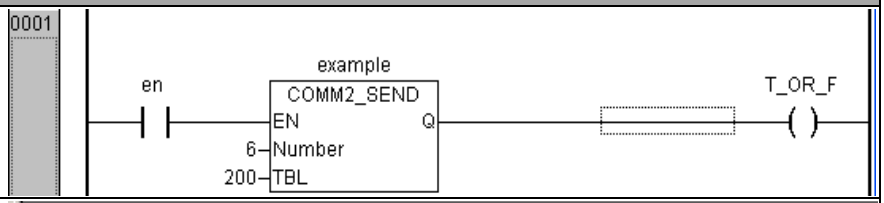
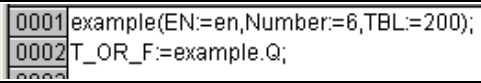
Example of Ladder Diagram Language


PARAMETER SPECIFICATIONS			
Input	Data Type	Function Description	Value
EN	enable	BOOL	0: disabled 1: rising edge enabled
Number	bytes sent	BYTE	
TBL	first address of send buffer in M area	INT	For: TBL=X, first address is %MBX
Output	Data Type	Function Description	Value
Q	indicate if sending is completed	BOOL	0: sending not completed 1: sending completed

APPLICATION EXAMPLE (LD and ST)

Variable Declaration

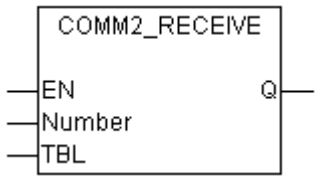


Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	

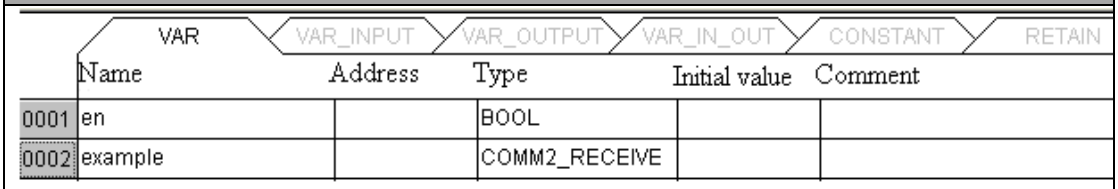
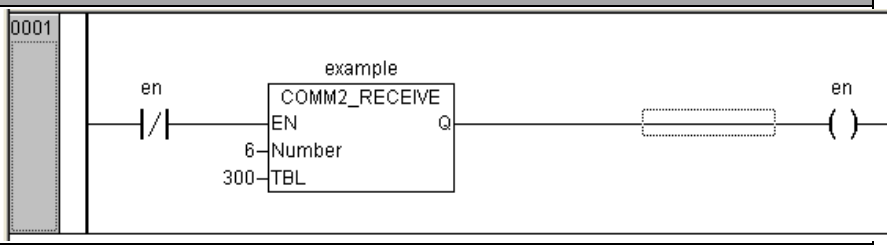
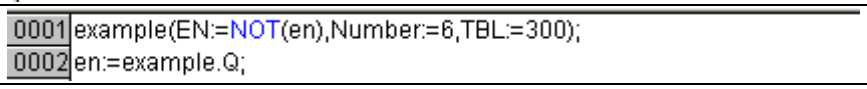
Remarks on the program:

- Under RS485 free port mode, when EN is set and retained, 6 consecutive bytes starting from %MB200 (%MB200-%MB205) are sent, Q is TRUE and retains.
- When EN is reset, no data is sent.

5.3.3 COMM2_RECEIVE—Receive RS485 Free Port Communication Data

Example of Ladder Diagram Language


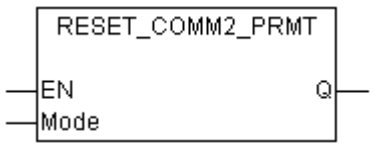
PARAMETER SPECIFICATIONS			
Input	Data Type	Function Description	Value
EN	BOOL	enable	0: disabled 1: rising edge enabled
Number	BYTE	bytes received	
TBL	INT	first address of receive buffer in M area	For: TBL=X, first address is %MBX
Output	Data Type	Function Description	Value
Q	BOOL	indicate if receiving is completed	0: receiving not completed 1: receiving completed

APPLICATION EXAMPLE (LD and ST)	
Variable Declaration	
	
Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	

Remarks on the program:

- Under RS232 free port mode, if EN equals to FALSE at power-on, in the first scanning period 6 bytes of data are stored in %MB300-%MB305 respectively, and Q is TRUE, which makes EN to be TRUE. In the second scanning period, no data is received, and Q is FALSE, which makes EN to be FALSE. In the third scanning period 6 bytes of data are stored in %MB300-%MB305 respectively. Data is received in such a cyclic manner.

5.3.4 Reset_COMM2_PRMT—Reset RS485 Protocol Settings

Example of Ladder Diagram Language


PARAMETER SPECIFICATIONS			
Input	Data Type	Function Description	Value
EN	BOOL	enable	0: disabled 1: rising edge enabled
Mode	BYTE	communication mode (7 bits no parity is not supported, default is 7 bits even parity)	Bits 7 6 5 4 3 2 1 0 parity | number of bits | baud rate | not defined See detailed Specification table of the Mode parameter below.
Output	Data Type	Function Description	Value
Q	BOOL	indicate if setting is completed	0: setting not completed 1: setting completed

TABLE OF MODE PARAMETER								
Parity		Number of Bits	Baud Rate			Not Defined		Comments
7	6	5	4	3	2	1	0	
0	0							keep current communication parameters(reset LM special protocol)
0	1							even
1	0							no parity (7 bits no parity is not supported; default to 7 bits even parity)
1	1							odd parity
		0						8 bits
		1						7 bits
			0	0	0			38400bps
			0	0	1			19200bps
			0	1	0			9600bps
			0	1	1			4800bps
			1	0	0			2400bps
			1	0	1			1200bps
			1	1	0			600bps
			1	1	1			300bps

Note:

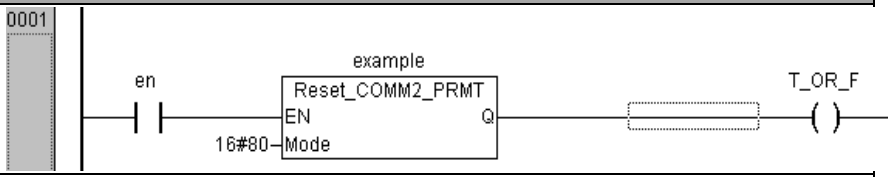
- The default protocol of RS485 is MODBUS, after a port is set to a free port by using the Set_COMM2_PRMT instruction, Reset_COMM2_PRMT needs to be used to reset to the original protocol.

APPLICATION EXAMPLE (LD and ST)

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT | RETAIN

	Name	Address	Type	Initial value	Comment
0001	en		BOOL		
0002	example		Reset_COMM2_PRI		
0003	T_OR_F		BOOL		

Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	0001 example(EN:= en,Mode:=16#80); 0002 T_OR_F:=example.Q;

As shown in the table below, when Mode is 16#80(2#10000000), the LM proprietary protocol is no parity, 8 bits and 38400bps.

Mode=16#80							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Parity		Number of Bits	Baud Rate			Not Defined	
1	0	0	0	0	0	0	0

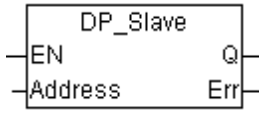
Remarks on the program:

- When EN is set and retained, RS485 is set to use ModBus protocol with no parity, 8 bits and 38400bps.

5.4 PROFIBUS-DP MODULE (HOLLYSYS_PLC_DPSLAVE.LIB)

5.4.1 DP_Slave—Call Profibus-DP Slave Module (LM3401)

Example of Ladder Diagram Language



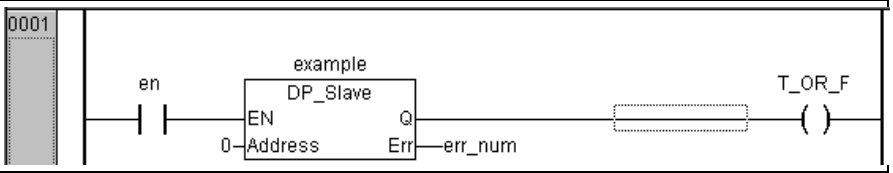
PARAMETER SPECIFICATIONS			
Storage Address of CPU Module Input Output Data			
%IWxx %QWxx	I and Q area address of DP salve module specified in PLC configuration		
Input	**Data Type**	**Function Specification**	**Value**
EN	BOOL	enable	0: disabled, can not scan DP module 1: enabled, can scan DP module
Address	BYTE	module node id	0-7(consistent with node id in PLC configuration)
Output	**Data Type**	**Function Specification**	**Value**
Q	BOOL	Indicate if valid data has been read	0: no valid data read 1: valid data read
Err	BYTE	communication error with slave module	0: no error 1: error waiting Ready from DP slave module 2: error reading interrupt from DP slave module 3: error reading response 4: error reading data length 5: error reading data 6: error writing data 7: error writing interrupt to DP slave module

APPLICATION EXAMPLE (LD and ST)

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT | RETAIN

	Name	Address	Type	Initial value	Comment
0001	en		BOOL		
0002	example		DP_Slave		
0003	err_num		BYTE		
0004	T_OR_F		BOOL		

Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	0001 example(EN:= en,Address:=0); 0002 T_OR_F:=example.Q; 0003 err_num:=example.Err;

The input of Address in the above example shall be consistent with the node id in PLC configuration.

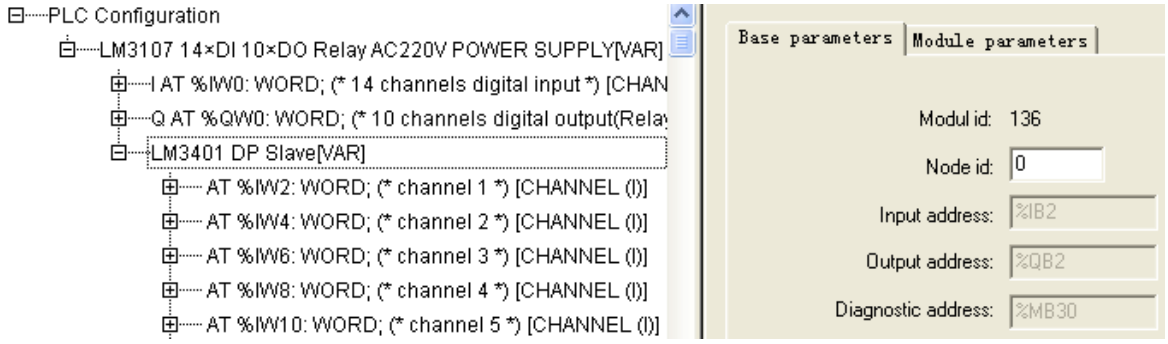


Figure 5-4-1

The input and output data sizes can be configured in the Value column of module parameters, as shown in figure 5-4-2, where the minimum value is 0, and the maximum is 64.

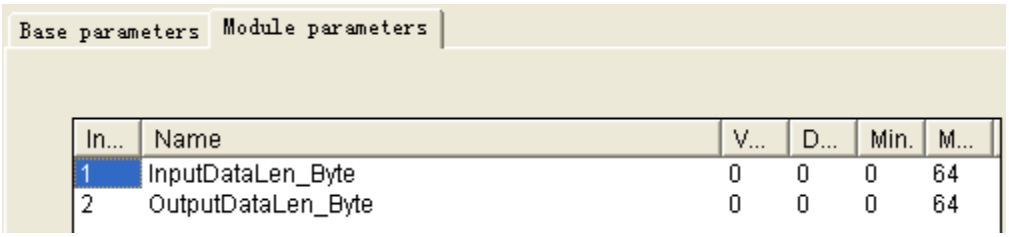


Figure 5-4-2

Remarks on the program:

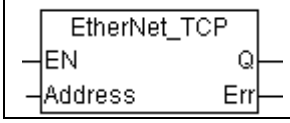
- When EN is set, DP module will be scanned, and when EN is reset, DP module will not be scanned.

NOTE:
Please refer to Appendix B for application examples on using DP module.

5.5 ETHERNET MODULE (HOLLYSYS_PLC_ETHERNET.LIB)

5.5.1 EtherNet_TCP—EtherNet Module (LM3403) Call

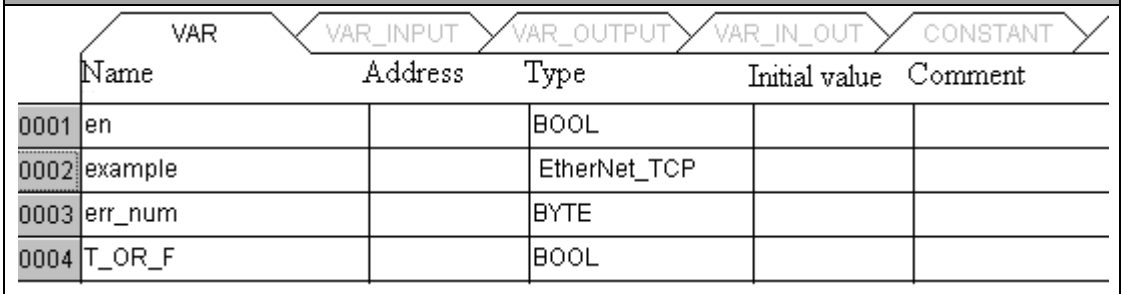
Example of Ladder Diagram Language

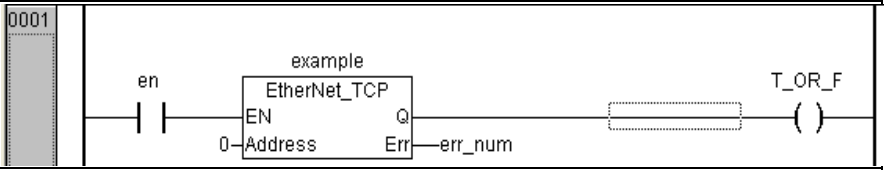
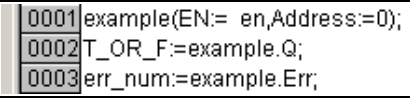


PARAMETER SPECIFICATIONS			
Storage Address of CPU Module Input Output Data			
%IWxx %QWxx	I and Q area address of EtherNet module specified in PLC configuration		
Input	Data Type	Function Description	Value
EN	BOOL	enable	0: disabled, can not scan EtherNet module 1: enabled, can scan EtherNet module
Address	BYTE	module node id	0-7(consistent with node id in PLC configuration)
Output	Data Type	Function Description	Value
Q	BOOL	valid data	0: no valid data read 1: valid data read
Err	BYTE	Communication error with EtherNet module	0: no error 1: error waiting Ready for EtherNet module 2: error reading interrupt from EtherNet module 3: error reading response 4: error reading data length 5: error reading data 6: error writing data 7: error writing interrupt to EtherNet module

APPLICATION EXAMPLE (LD and ST)

Variable Declaration



Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	


```
0001 example(EN:= en,Address:=0);
0002 T_OR_F:=example.Q;
0003 err_num:=example.Err;
```

The input of Address in the above example shall be consistent with the node id in PLC configuration in figure 5-5-1.

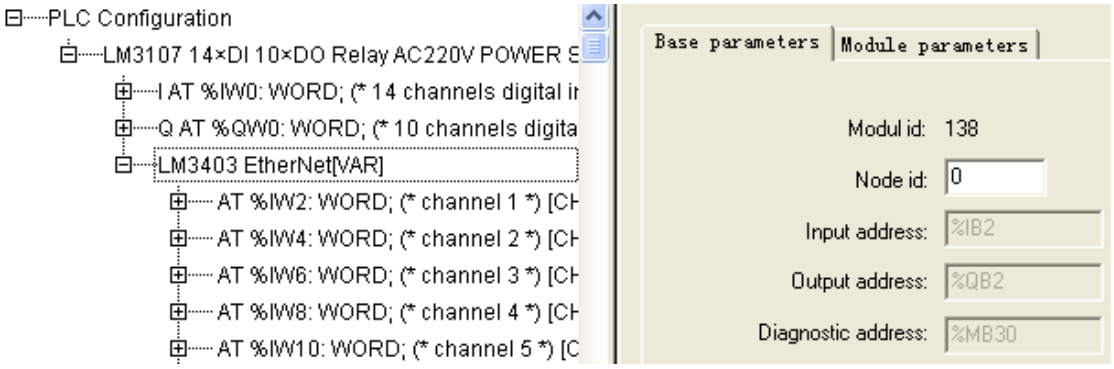
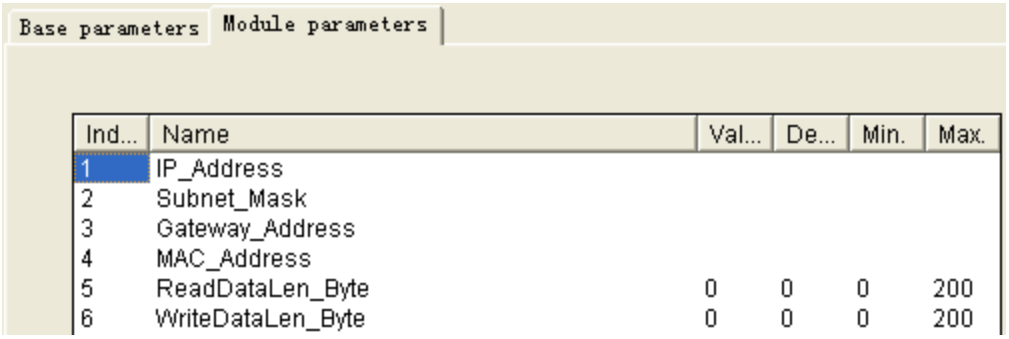


Figure 5-5-1

Figure 5-5-2 shows the configuring of IP_Address, Subnet_Mask, Gateway_Address, ReadDataLen_Byte, and WriteDataLen_Byte.



Ind...	Name	Val...	De...	Min.	Max.
1	IP_Address				
2	Subnet_Mask				
3	Gateway_Address				
4	MAC_Address				
5	ReadDataLen_Byte	0	0	0	200
6	WriteDataLen_Byte	0	0	0	200

Figure 5-5-2

Remarks on the Program:

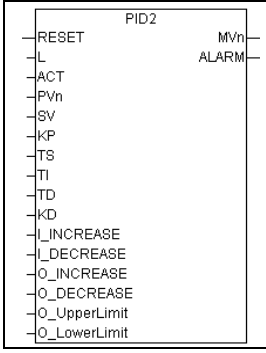
- When EN is set, EtherNet module will be scanned, and when EN is reset, EtherNet module will not be scanned.

Note:
Please refer to Appendix B for more application examples on using Ethernet module.

5.6 POSITIVE AND NEGATIVE ACTION OPTIONAL PID CONTROLLER (HOLLYSYS_PLC_UTIL.LIB)

5.6.1 PID2—Optional positive and negative action PID controller

Example of Ladder Diagram Language



PARAMETER SPECIFICATIONS			
Input	Data Type	Function Description	Value
RESET	BOOL	reset, valid high voltage	
L	REAL	filter coefficient (0-1)	
ACT	UINT	action/alarm setting (ACT.2 and ACT.5 shall be 1 synchronously.)	.0 = 0—positive, 1—negative .1 = 0—invalid input alarm 1—valid input alarm .2 = 0— invalid output alarm 1—valid output alarm .5 = 0—invalid output upper and lower limits setting 1—valid output upper and lower limits setting
PVn	REAL	process variable	
SV	REAL	setpoint value	
KP	REAL	proportional increase constant (KP>0)	
TS	REAL	sampling period (TS>0)	in second (S)
TI	REAL	integral constant (TI≥0)	in second (S)
TD	REAL	derivative constant (TD≥0)	in second (S)
KD	REAL	derivative increase constant (0-1)	
I_INCREASE	REAL	input increase alarm setpoint	
I_DECREASE	REAL	input decrease alarm setpoint	
O_INCREASE	REAL	output increase alarm setpoint	
O_DECREASE	REAL	output decrease alarm setpoint	
O_UpperLimit	REAL	output upper limit setpoint	
O_LowerLimit	REAL	output lower limit setpoint	
Output	Data Type	Function Description	Value
MVn	REAL	manipulator Variable	
ALARM	UINT	alarm	.0 = 0—input increase normal 1—input increase overflow .1 = 0—input decrease normal 1—input decrease overflow .2 = 0—output increase normal 1—output increase overflow .3 = 0—output decrease normal 1—output decrease overflow

BASIC PID FORMULA

Expression	Meaning		Expression	Meaning
EVn	present sampling bias		Dn	Nth derivative
EVn-1	previous sampling bias		Dn-1	N-1th derivative
SV	setpoint value		KP	proportional increase constant
PVnf	process value of nth sampling(after filter)		TS	sampling period
PVnf-1	process value of n-1th sampling(after filter)		TI	integral constant
PVnf-2	process value of n-2th sampling(after filter)		TD	derivative constant
ΔMV	manipulator variable changes		MVn	Nth manipulator variable
KD：	derivative constant			

Positive Action:

$$\Delta MV = K_p\{(EV_n - EV_{n-1}) + \frac{T_S}{T_I} * EV_n + D_n\}$$

$$EV_n = PV_{nf} - SV$$

$$D_n = \frac{T_D}{T_S + KD * T_D}(-2PV_{nf-1} + PV_{nf} + PV_{nf-2}) + \frac{KD * T_D}{T_S + KD * T_D} * D_{n-1}$$

$$MV_n = \sum \Delta MV$$

Negative Action:

$$\Delta MV = K_p\{(EV_n - EV_{n-1}) + \frac{T_S}{T_I} * EV_n + D_n\}$$

$$EV_n = SV - PV_{nf}$$

$$D_n = \frac{T_D}{T_S + KD * T_D}(2PV_{nf-1} - PV_{nf} - PV_{nf-2}) + \frac{KD * T_D}{T_S + KD * T_D} * D_{n-1}$$

$$MV_n = \sum \Delta MV$$

- PV_{nf} is the result calculated by reading the process variables in the following formula.
- Process variables after filter $PV_{nf} = PV_n + \mathrm{L}(PV_{nf-1} - PV_n)$
- PV_n: the process value of the nth sampling process.
- L: filter coefficient (0~1).
- PV_{nf-1}: the n-1th process variable1 (after filter).

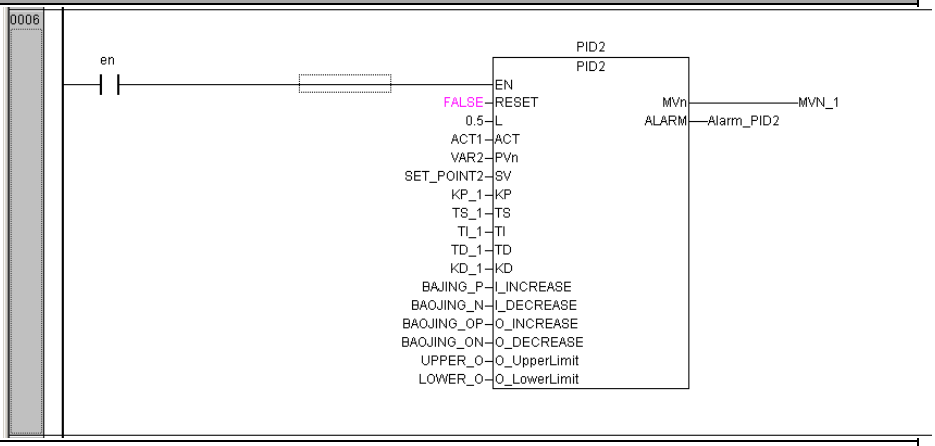
APPLICATION EXAMPLE (LD and ST)

Variable Declaration

	Name	Address	Type	Initial value	Comment
0001	PID2		PID2		
0002	en		BOOL		
0003	ACT1		UINT		
0004	VAR2		REAL		
0005	SET_POINT2		REAL	100	
0006	KP_1		REAL	0.1	
0007	TS_1		REAL	100	
0008	TI_1		REAL	100	
0009	TD_1		REAL	100	
0010	KD_1		REAL	0.2	
0011	BAJING_P		REAL	500	
0012	BAOJING_N		REAL	-500	
0013	BAOJING_OP		REAL	500	
0014	BAOJING_ON		REAL	-500	
0015	UPPER_O		REAL	1000	
0016	LOWER_O		REAL	-1000	
0017	MVN_1		REAL		
0018	Alarm_PID2		UINT		

Programming Language: Program

Ladder Diagram (LD)



Structured Text (ST)

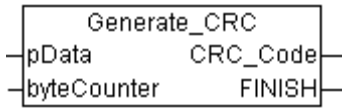
```
0001 IF en=TRUE THEN;
0002 PID2(RESET:=FALSE,L:=0.5,ACT:=ACT1,PVn:=VAR2,SV:=SET_POINT2,KP:=KP_1,TS:=TS_1,TI:=TI_1,TD:=TD_1,KD:=KD_1,I_INCREASE:=BAJING_
0003 I_DECREASE:=BAOJING_N,O_INCREASE:=BAOJING_OP,O_DECREASE:=BAOJING_ON,O_UpperLimit:=UPPER_O,O_LowerLimit:=LOWER_O);
0004 MVN_1:=PID2.MVn;
0005 Alarm_PID2:=PID2.ALARM;
0006 END_IF
```

Remarks on the Program:

- When EN is enabled, the PID2 instruction will complete the PID operations by calculating various parameters according to positive action or negative action formula; when EN is reset, the instruction will not be executed.

5.7 MODBUS CRC CHECK (HOLLYSYS_PLC_MODBUS_CRC.LIB)

5.7.1 Generate_CRC—Generate the Modbus CRC Codes.

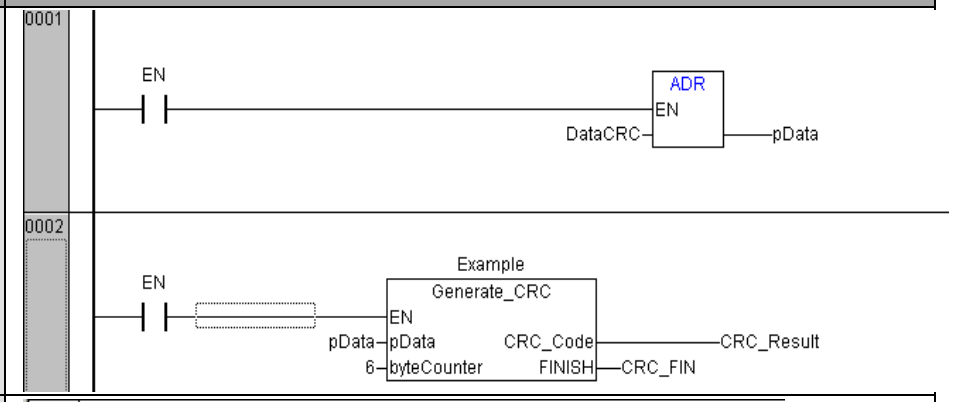
Example of Ladder Diagram Language


PARAMETER SPECIFICATIONS			
Input	Data Type	Function Description	Value
pData	POINTER TO BYTE	initial pointer of data needs to be CRC checked	
byteCounter	WORD	byte of data needs to be CRC checked	
Output	Data Type	Function Description	Value
CRC_Code	WORD	CRC check result	
FINISH	BOOL	Indicate if CRC check is completed	0: not completed 1: completed

APPLICATION EXAMPLE (LD and ST)

Variable Declaration

	Name	Address	Type	Initial value	Comment
0001	EN		BOOL		
0002	pData		POINTER TO BYTE		
0003	Example		Generate_CRC		
0004	CRC_Result	%MW206	WORD		
0005	CRC_FIN		BOOL		
0006	DataCRC	%MB200	ARRAY[1..6] OF BYTE	16#01,16#03,16#0C, 16#1C,16#00,16#01;	

Programming Language	Program
Ladder Diagram (LD)	

Structured Text (ST)

```
0001 IF EN=TRUE THEN
0002    pData:=ADR(DataCRC);
0003    Example(pData:=pData,byteCounter:=6);
0004    CRC_Result:=Example.CRC_Code;
0005    CRC_FIN:=Example.FINISH;
0006 END_IF
```

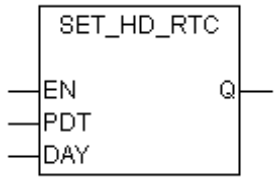
Remarks on the Program:

- After the setting of EN, first use ADR to calculate the pointer of array DataCRC, assign it to Generate_CRC and input in byteCounter the number of bytes 6 that needed to be CRC checked, then the Generate_CRC instruction will calculate the CRC_Codes 16#01, 16#03, 16#0C, 16#1C, 16#00, 16#01 and store them to CRC_Result (where the address is %MW206).

Note:
Enable operators must be used to call the Generate_CRC instruction.

5.8 HARDWARE REAL TIME CLOCK (HOLLYSYS_PLC_HDRTC.LIB)

5.8.1 Set_HD_RTC — Set Hardware Real Time Clock (data type: DT)

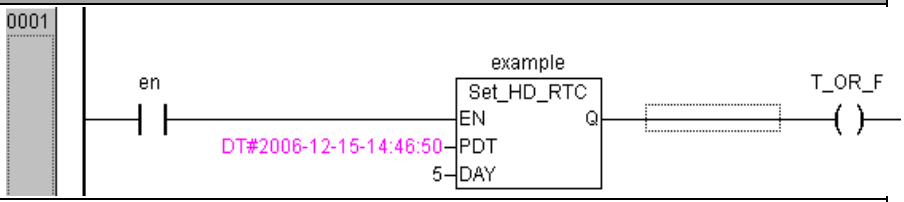
Example of Ladder Diagram Language


PARAMETER SPECIFICATIONS			
Input	Data Type	Function Description	Value
EN	BOOL	enable	0: disabled 1: rising edge enabled
PDT	DT	date/time	From DT#2000-01-01-00: 00: 00 To DT#2099-12-31-23:59:59
DAY	BYTE	day	1-7
Output	Data Type	Function Description	Value
Q	BOOL	indicate if the setting is completed	0: not completed 1: completed

APPLICATION EXAMPLE (LD and ST)

Variable Declaration

	Name	Address	Type	Initial value	Comment
0001	en		BOOL		
0002	T_OR_F		BOOL		
0003	example		Set_HD_RTC		

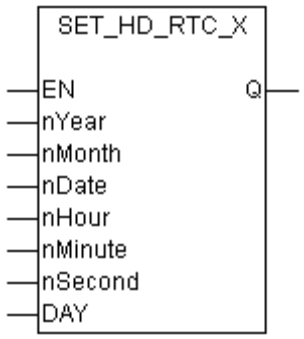
Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	0001 example(EN:= en,PDT:=DT#2006-12-15-14:46:50, DAY:=5); 0002 T_OR_F:=example.Q;

Remarks on the Program:

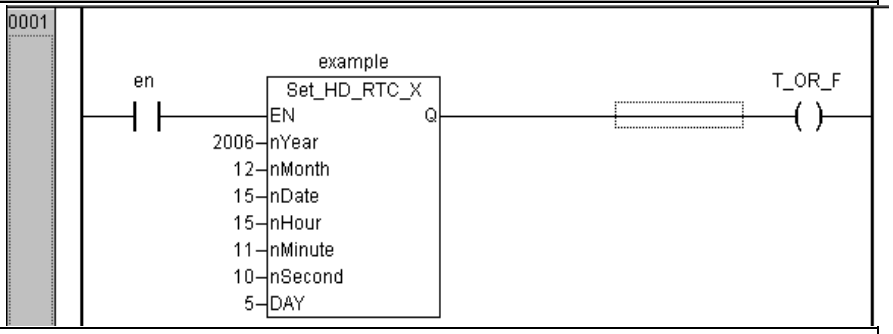
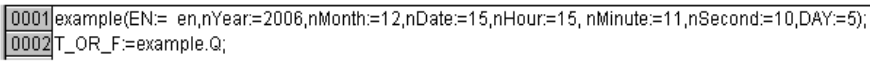
- When EN is set, the date, time and day shown in the above figure will be set to PLC hardware real time clock, the current HD_RTC is: 2006-12-15-14:46:50, Friday.

Note:
The date and time cannot exceed the specified range (DT#2000-01-01-00:00:00 to DT#2099-12-31-23:59:59).

5.8.2 Set_HD_RTC_X—Set Hardware Real Time Clock (data type: TP)

Example of Ladder Diagram Language


PARAMETER SPECIFICATIONS			
Input	Data Type	Function Description	Value
EN	BOOL	enable	0: disabled 1: rising edge enabled
nYear	INT	year	2000-2099
nMonth	BYTE	month	1-12
nDate	BYTE	day	1-31
nHour	BYTE	hour	0-24
nMinute	BYTE	minute	0-60
nSecond	BYTE	second	0-60
DAY	BYTE	day	1-7
Output	Data Type	Function Description	Value
Q	BOOL	Indicate if the setting is completed	0: not completed 1: completed

APPLICATION EXAMPLE (LD and ST)	
Variable Declaration	
0001 en — BOOL	
0002 T_OR_F — BOOL	
0003 example — Set_HD_RTC_X	
Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	


```
0001 example(EN:= en,nYear:=2006,nMonth:=12,nDate:=15,nHour:=15, nMinute:=11,nSecond:=10,DAY:=5);
0002 T_OR_F:=example.Q;
```

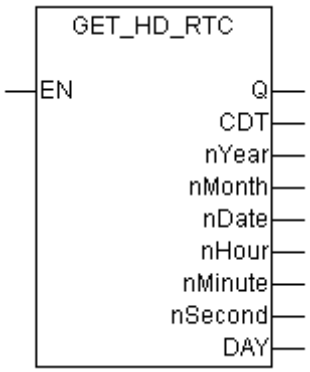
Remarks on the Program:

- When EN is set, the date, time and day shown in the above figure will be set to PLC hardware real time clock, the current HD_RTC is: 2006-12-15-15:11: 10, Friday.

Note:
The date and time cannot exceed the specified range (DT#2000-01-01-00:00:00 to DT#2099-12-31-23:59:59).

5.8.3 Get_HD_RTC—Get Hardware Real Time Clock: date/time/day

Example of Ladder Diagram Language



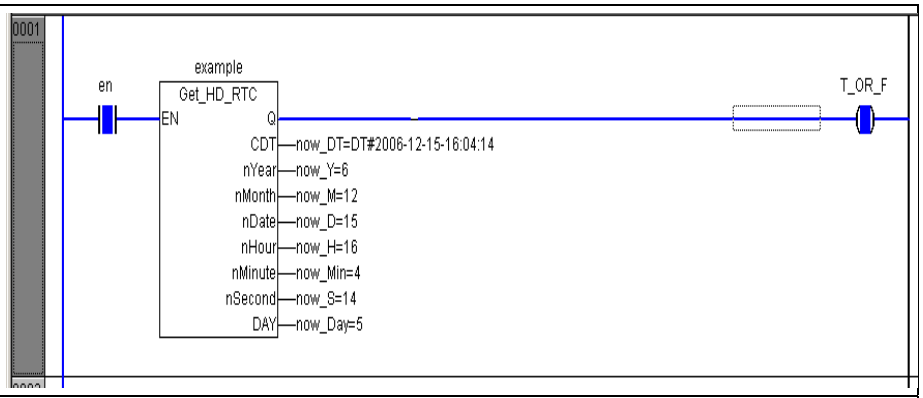
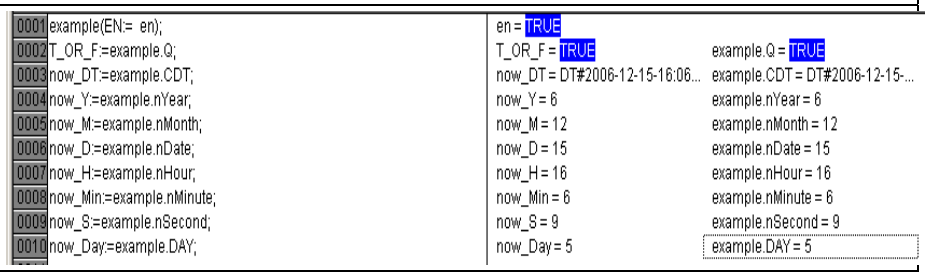
PARAMETER SPECIFICATIONS			
Input	Data Type	Function Description	Value
EN	BOOL	enable	0: disabled 1: enabled
Output	Data Type	Function Description	Value
Q	BOOL	indicate if valid data has been read	0: no valid data read 1: valid data read
CDT	DT	date/time	from DT#2000-01-01-00: 00: 00 to DT#2099-12-31-23:59:59
nYear	INT	year	0-99(the last two numbers)
nMonth	BYTE	month	1-12
nDate	BYTE	day	1-31
nHour	BYTE	hour	0-24
nMinute	BYTE	minute	0-60
nSecond	BYTE	second	0-60
DAY	BYTE	day	1-7

APPLICATION EXAMPLE (LD and ST)

Variable Declaration

	Name	Address	Type	Initial value
0001	en		BOOL	
0002	T_OR_F		BOOL	
0003	example		Get_HD_RTC	
0004	now_DT		DT	
0005	now_Y		INT	
0006	now_M		BYTE	
0007	now_D		BYTE	
0008	now_H		BYTE	
0009	now_Min		BYTE	
0010	now_S		BYTE	
0011	now_Day		BYTE	

Programming Language	Program

Ladder Diagram (LD)	
Structured Text (ST)	

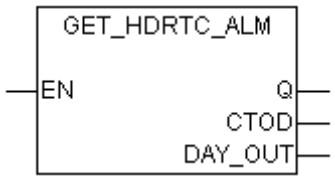
Parameter Comments:

- When EN is set, CDT will output the date, time of PLC HD_RTC in DT data type, nYear will output year in INT data type, and nMonth, nDate, nHour, nMinute, nSecond, DAY will output month, date, hour, minute, second, day in BYTE data type, Q equals TRUE, the current RTC is: 2006-12-15-16:04:14, Friday.
- When EN is reset, Q equals FALSE while CDT, nYear, nMonth, nDate, nHour, nMinute, nSecond, DAY retain the last output values.

5.9 HD_RTC ALARM (HOLLYSYS_PLC_HDRTCALM.LIB)

5.9.1 Get_HDRTC_ALM—Read HDRTC Alarm Time/Day

Example of Ladder Diagram Language



PARAMETER SPECIFICATIONS			
Input	Data Type	Function Description	Value
EN	BOOL	enable	0: disabled 1: enabled
Output	Data Type	Function Description	Value
Q	BOOL	indicate if the alarm data has been read	0: no alarm data read 1: alarm data read
CTOD	TOD	current alarm time	
DAY_OUT	BYTE	current alarm day	1-7

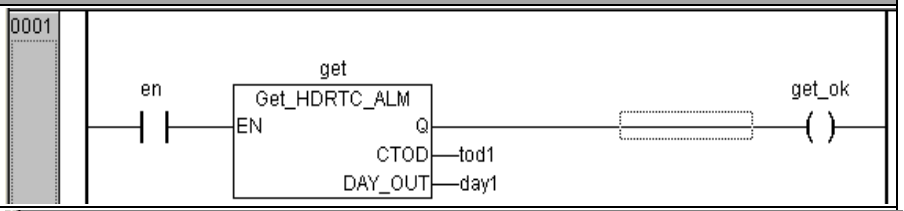
APPLICATION EXAMPLE (LD and ST)

Variable Declaration

	Name	Address	Type	Initial value	Comment
0001	en		BOOL		
0002	tod1		TOD		
0003	day1		BYTE		
0004	get		Get_HDRTC_ALM		
0005	get_ok		BOOL		

Programming Language: Ladder Diagram (LD)

Program:



Programming Language: Structured Text (ST)

Program:

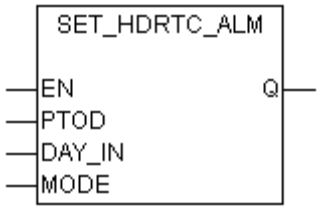
```
0001 get(EN:= en);
0002 get_ok:=get.Q;
0003 tod1:=get.CTOD;
0004 day1:=get.DAY_OUT;
```

Remarks on the Program:

- When EN is set, CTOD will output RTC alarm time of PLC, DAY_OUT will output RTC alarm day of PLC, Q equals to TRUE, and the current alarm time is 15:21: 22, Monday.
- When EN is reset, Q equals to FALSE while CTOD and DAY_OUT retain the last output values.

5.9.2 Set_HDRTC_ALM—Set HDRTC Alarm Time/Day

Example of Ladder Diagram Language



PARAMETER SPECIFICATION

Input	Data Type	Function Description	Value
EN	BOOL	enable	0: disabled 1: rising edge enabled
PTOD	TOD	time	
DAY_IN	BYTE	day	1-7
MODE	BYTE	alarm modes	0: alarm event closed 1: when day, hour, minute, second match, alarm event triggered (once a week) 2: when hour, minute, second match, alarm event triggered (once a day) 3: when minute, second match, alarm event triggered (once an hour) 4: when second matches, alarm event triggered (once each minute) 5: alarm event triggered once a second
Output	**Data Type**	**Function Description**	**Value**
Q	BOOL	Indicate if the alarm data has been correctly set	0: alarm data not set 1: alarm data correctly set

Alarm Associated Event

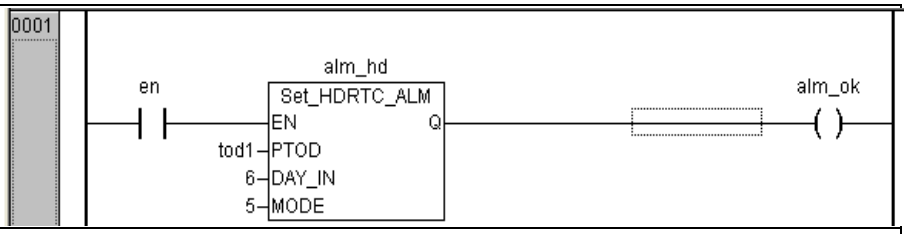
- HD_RTC_ALM 0 interrupt.

APPLICATION EXAMPLE (LD and ST)

Variable Declaration

	Name	Address	Type	Initial value	Comment
0001	en		BOOL		
0002	tod1		TOD		
0003	alm_hd		Set_HDRTC_ALM	T#12：05：0(	
0004	alm_ok		BOOL		

Programming Language	Program

Ladder Diagram (LD)	
Structured Text (ST)	0001 alm_hd(EN:= en,PTOD:=TOD1,DAY_IN:=6,MODE:=5); 0002 alm_ok:=alm_hd.Q;

Remarks on the Program:

- When EN is set, configure PLC real alarm time /day as shown in the above figure, Q equals to TRUE, the current alarm time is 12:05:00, Saturday. Since the mode 5 is selected, alarm event will be triggered once a second, the time and day will be invalid in this case.
- When EN is reset, Q equals to FALSE.

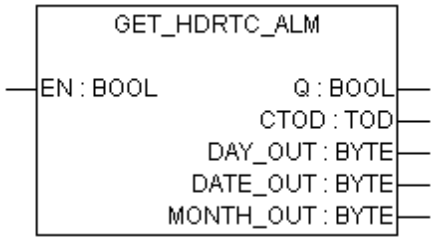
Note:

- After each alarm event trigger, HDRTC alarm data shall be re-read（re-call Get_HDRTC_ALM）, otherwise the next alarm cannot startup.
- Refer to appendix B for interruption relevancy.

5.10 LM3104/5 REAL TIME CLOCK ALARM (HOLLYSYS_PLC_HDRTCALM_N.LIB)

5.10.1 GET_HDRTC_ALM—Read the Real Time Clock Interrupt Time

Example of Ladder Diagram Language



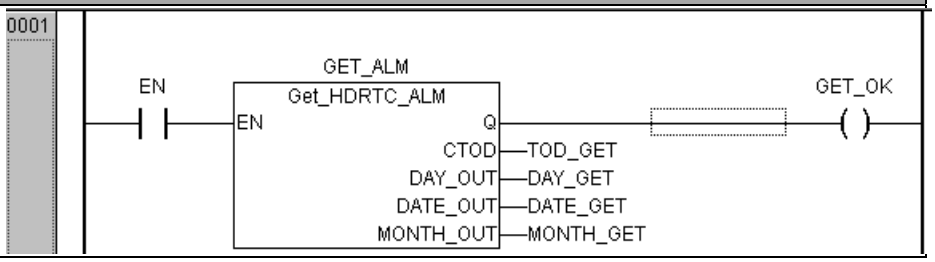
PARAMETER SPECIFICATIONS

Input	Data Type	Function Description	Value
EN	BOOL	enable	0: disable 1: enable
Output	**Data Type**	**Function Description**	**Value**
Q	BOOL	indicate if interrupt data has been read	0: no interrupt data read 1: interrupt data read
CTOD	TOD	current interrupt time	
DAY_OUT	BYTE	current interrupt day	1-7
DATE_OUT	BYTE	current interrupt date	1-31
MONTH_OUT	BYTE	current interrupt month	1-12

APPLICATION EXAMPLE (LD and ST)

Variable Declaration

	Name	Address	Type	Initial value	Comment
0001	EN		BOOL		
0002	GET_OK		BOOL		
0003	GET_ALM		Get_HDRTC_ALM		
0004	TOD_GET		BYTE		
0005	DAY_GET		BYTE		
0006	DATE_GET		BYTE		
0007	MONTH_GET		BYTE		

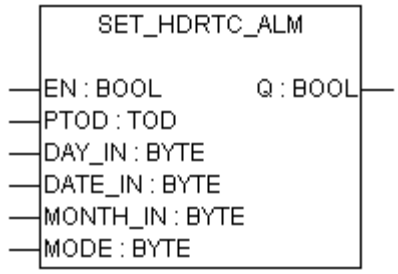
Programming Language	Program
Ladder Diagram (LD)	

Structured Text (ST)	0001 GET_ALM(EN:= EN); 0002 GET_OK:=GET_ALM.Q; 0003 TOD_GET:=GET_ALM.CTOD; 0004 DAY_GET:=GET_ALM.DAY_OUT; 0005 DATE_GET:=GET_ALM.DATE_OUT; 0006 MONTH_GET:=GET_ALM.MONTH_OUT;

Remarks on the Program:

- When EN is set, CTOD will output PLC HDRTC alarm time in TOD data type, MONTH_OUT, DATE_OUT, DAY_OUT will output HDRTC alarm month, date, day in BYTE data type, and Q equals to TRUE.
- When EN is reset, Q equals to FALSE, CTOD, MONTH_OUT, DATE_OUT, DAY_OUT retain the last output values.

5.10.2 SET_HDRTC_ALM—Set HDRTC Interrupt Time

Example of Ladder Diagram Language


PARAMETER SPECIFICATIONS			
Input	Data Type	Function Description	Value
EN	BOOL	enable	0: disabled 1: rising edge enabled
PTOD	TOD	time	
DAY_IN	BYTE	day	1-7
DATE_IN	BYTE	date	1-31
MONTH_IN	BYTE	month	1~12
MODE	BYTE	interrupt modes	See table 5-10-1 for details.
Output	Data Type	Function Description	Value
Q	BOOL	Indicate if the interrupt data has been set	0: interrupt data not set 1: interrupt data set

7	6	5	4	3	2	1	0
not defined	not defined	0: day match forbidden 1: day match	0: month match forbidden 1: month match	0: date match forbidden 1: date match	0: hour match forbidden 1: hour match	0: minute match forbidden 1: minute match	0: second Match forbidden 1: second match

Table 5-10-1

- If the interrupt mode is 2#00000011, interrupts will be generated when the minute and second of current clock match the set clock.

Alarm Associated Event

- HD_RTC_ALM 0 interrupt.

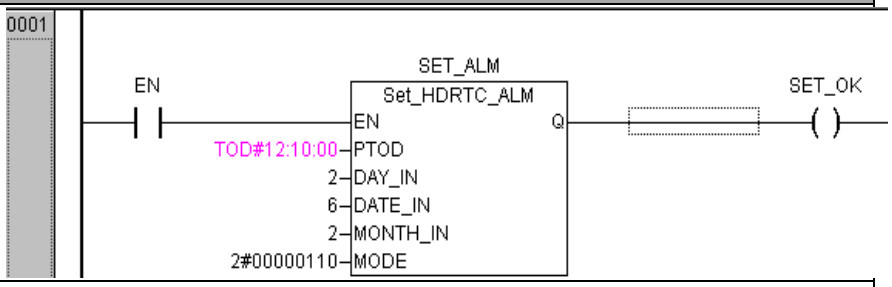
APPLICATION EXAMPLE (LD and ST)

Variable Declaration

	Name	Address	Type	Initial value	Comment
0001	EN		BOOL		
0002	SET_OK		BOOL		
0003	SET_ALM		Set_HDRTC_ALM		

Programming Language: Program

Ladder Diagram (LD)



Structured Text (ST)

```
0001 SET_ALM(EN:= EN,PTOD:=TOD#12:10:00,DAY_IN:=2,DATE_IN:=6,MONTH_IN:=2,MODE:=2#0000011
0002 SET_OK:=SET_ALM.Q;
0003
```

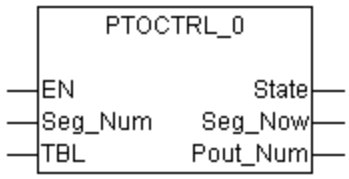
Remarks on the Program:

- When EN is set, configure the PLC real time alarm time as shown in the above figure; Q equals to TRUE, the current alarm time is 2-6-12: 10: 00, Tuesday. Because the mode is 2#00000110, so an alarm will be triggered at 12:10 everyday trigger, and the settings of month, date, second and day will be invalid in this case.
- When EN is reset, Q equals to FALSE.

Note:
Refer to Appendix B on the usage of interrupt events.

5.11 MULTI-SEGMENT PULSE TRANSMISSION INSTRUCTION (HOLLYSYS_PLC_PTOCTRL.LIB)

5.11.1 PTOCtrl_0 — Channel 1.1 Multi-segment Pulse Transmission

Example of Ladder Diagram Language


Applicable Module

- LM3106-CDT, LM3108-CDT.

Functions

- This instruction provides the function of generating pulse string of square wave with designated number of pulses (50% duty cycle). The period is designated in microsecond (ranging from 20 to 335,000). If the set value of the period is odd, the duty cycle would have a little distortion, pulse number output channel is Q1.1, range 0-4294967295.

Parameter Specifications

PARAMETER SPECIFICATIONS			
External terminal output	Function Description		
Q1.1	high-speed pulse output		
Input	Data Type	Functional Description	Value
EN	BOOL	Enable	0: disabled 1: enabled
Seg_Num	BYTE	number of segment	1-255
TBL	INT	The first address of segments in M area	100-8000
Output	Data Type	Functional Description	Value
State	BOOL	Indicate the pulse transmission status	0: transmission 1: in transmission
Seg_Now	BYTE	currently transmitting segment No	1-255
Pout_Num	DWORD	number of transmitted pulses	0-4294967295

ASSOCIATED INSTRUCTION CONFLICT			
Used Instruction A	Associated Instruction B	Associated Hardware	Availability
PTOCtrl_0	PTO_PWM0	T12	unavailable instruction B
	PTO_PWM0_Run		

Application Example of PTOCtrl_0 (LD and ST)

- For example, Q1.1 transmits pulses according to the parameters defined in Table 5-11- There are 4 segments in the table, and each segment contains totally 10 bytes consisted of one 32-bit initial period value, one 16-bit period increment, one 32-bit

number of pulses transmitted in this segment. Modification of period is made on each pulse.

Segment No.	Initial Period (μs, 32-bit)	Period Increment (μs, 16-bit)	Number of Pulses (32-bit)
#1	%MD200	%MW204	%MD206
	500	-2	100
#2	%MD210	%MW214	%MD216
	300	-3	50
#3	%MD220	%MW224	%MD226
	150	0	500
#4	%MD230	%MW234	%MD236
	150	5	100

Table 5-11-1 Pulse Transmission Parameters

The time-frequency relationship corresponding to the table above is shown in Fig. 5-11-1.

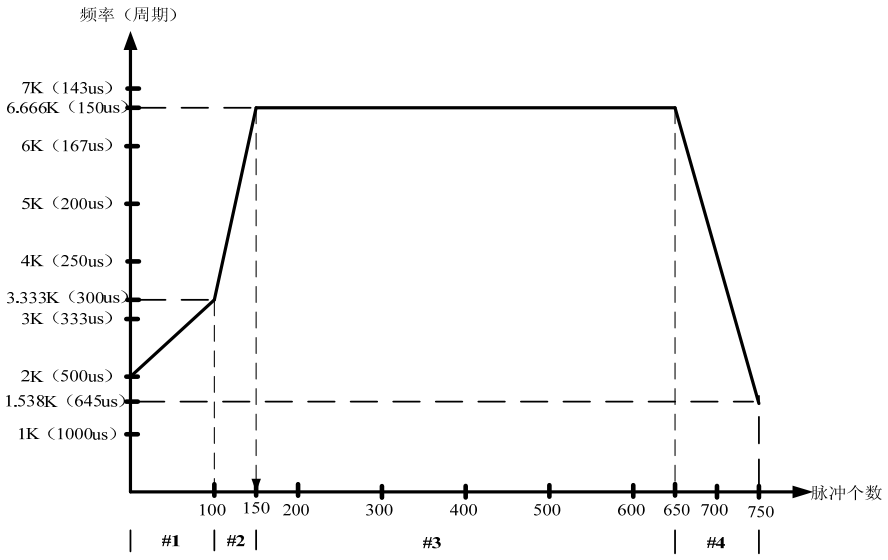


Figure 5-11-1 Time-frequency Relationship of Stepper Motor Application

Relationship between initial period, period increment and number of pulses is as follows:

- The period increment of a given segment = (ECT-ICT)/Q
- ECT=Period value at the end of the segment (i.e. period value at the beginning of the next segment)
- ICT=Initial period value of the segment
- Q=Number of pulses transmitted on this segment
- I.e.: # n period increment = (# n+1 Initial period - # n Initial period)/# n number of pulses
- For example: #1 period increment = (#2 Initial period - #1 Initial period)/#1 number of pulses.

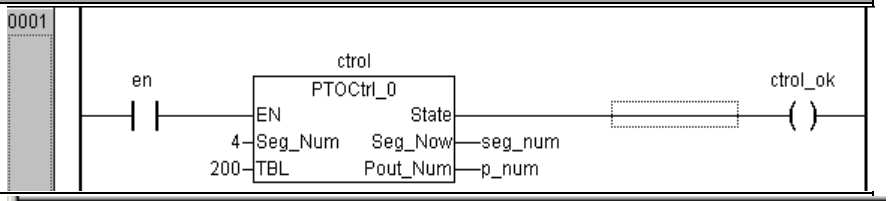
APPLICATION EXAMPLE (LD and ST)

Variable Declaration

	Name	Address	Type	Initial value	Comment
0001	ctrol_ok		BOOL		
0002	seg_num		BYTE		
0003	p_num		DWORD		
0004	en		BOOL		
0005	q11	%QX1.1	BOOL		
0006	ctrol		PTOCtrl_0		
0007	PTOSeg	%MW200	ARRAY [1..4] OF PTOCtrlSet	(P:=500,I:=-2,N:=100), (P:=300,I:=-3,N:=50), (P:=150,I:=0,N:=500), (P:=150,I:=5,N:=100);	

Programming Language: Ladder Diagram (LD)

Program:



Structured Text (ST)

```
0001 ctrol(EN:= en,Seg_Num:=4,TBL:=200);
0002 ctrol_ok:=ctrol.State;
0003 seg_num:=ctrol.Seg_Now;
0004 p_num:=ctrol.Pout_Num;
```

Remarks on the Program:

- When EN is enabled, CPU automatically reads the properties of each segment in the segment table from the designated address of M area (e.g. initial address of the table above is %MW200). Then Q.1.1 will start the transmission of a 500μs pulse width, which is reduced by 2μs for each pulse until the 100th pulse is transmitted. At this moment the transmission of the first segment is completed and the pulse width becomes 302μs. Then the second segment will be transmitted with a 300μs pulse width, which is reduced by 3μs for each pulse. When the 50th pulse is sent, the pulse width becomes 153μs. Then the third segment will be transmitted with a pulse width of 150μs. The pulse increment is 0 in the third segment; therefore the pulse width remains 150μs throughout the whole transmission of 500 pulses in this segment. Then the fourth segment will be transmitted with a pulse width of 150μs. The pulse width increases by 5μs for each pulse that makes the pulse width stop at 645μs after transmitting 100 pulses.
- This instruction is executed after EN is enabled, and pulses are transmitted according to the curves shown in Time-frequency relationship map of stepper motor applications. Seg_Num indicates the currently processing segment number, and Pout_Num shows the number of pulses that have been transmitted.
- When EN is reset, all variables are reset.

Specification of Segment Table Definition:

There are two methods to define the segment table in M area. One is to use the PTOCtrlSet structure in HOLLYSYS_PLC_PTOCtrl.lib library to define a structure array, or a self-defined

structure array, and assign values to this structure array. Another method is to assign values to the addresses in M area in a sequence.

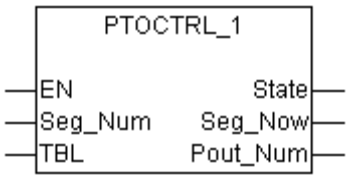
The Structure in HOLLYSYS_PLC_PTOCtrl.lib is shown as follows:

```
TYPE PTOCtrlSet :
STRUCT
    P: DWORD;
    I: INT;
    N: DWORD;
END_STRUCT
END_TYPE
```

In the above example, a structure array can be defined at %MW200 by the following method, simply filling the values in the table in the sequence.

```
PTOSeg AT %MW200: ARRAY [1..4] OF PTOCtrlSet:= (P:=500,I:=-2,N:=100),
(P:=300,I:=-3,N:=50),(P:=150,I:=0,N:=500),(P:=150,I:=5,N:=100);
```

5.11.2 PTOCtrl_1 — Channel 0.3 Multi-segment Pulse Transmission

Example of Ladder Diagram Language


Applicable module

- LM3104-CDT, LM3106-CDT, LM3108-CDT

Functions

- Similar to the function of PTOCtrl_0, the differences are: the period is in microsecond (range 20-5327), pulse output channel is Q0.3.

PARAMETER SPECIFICATIONS			
External terminal output	Parameter		
Q0.3	High-speed pulse output		
Input	Data Type	Function Description	Value
EN	BOOL	Enable	0: disabled 1: enabled
Seg_Num	BYTE	number of segment	1-255
TBL	INT	the first address of segments in M area	100-8000
Output	Data Type	Function Description	Value
State	BOOL	indicate the pulse transmission status	0: transmission stopped 1: in transmission
Seg_Now	BYTE	currently transmitting segment No.	1-255
Pout_Num	DWORD	number of the transmitted pulses	0-4294967295

ASSOCIATED INSTRUCTION CONFLICTS			
Used Instruction A	Associated Instruction B	Associated Hardware	Availability
PTOCtrl_1	PTO_PWM1	T13	Unavailable instruction B
	PTO_PWM1_Run		

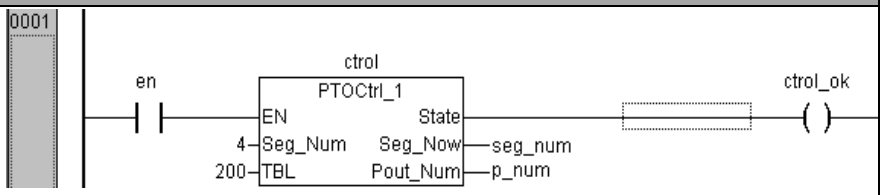
APPLICATION EXAMPLE (LD and ST)

Variable Declaration

	Name	Address	Type	Initial value	Comment
0001	ctrol_ok		BOOL		
0002	seg_num		BYTE		
0003	p_num		DWORD		
0004	en		BOOL		
0005	q03	%QX0.3	BOOL		
0006	ctrol		PTOCtrl_1		
0007	PTOSeg	%MW200	ARRAY [1..4] OF PTOCtrlSet	(P:=500,I:=-2,N:=100), (P:=300,I:=-3,N:=50), (P:=150,I:=0,N:=500), (P:=150,I:=5,N:=100);	

Programming Language: Ladder Diagram (LD)

Program:



Structured Text (ST)

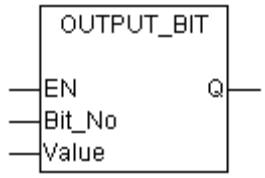
```
0001 ctrol(EN:= en,Seg_Num:=4,TBL:=200);
0002 ctrol_ok:=ctrol.State;
0003 seg_num:=ctrol.Seg_Now;
0004 p_num:=ctrol.Pout_Num;
```

Remarks on the Program:

- This instruction is executed when EN is enabled, and the pulses are transmitted according to the curves shown in the figure above. Seg_Num indicates the currently processing segment number, and Pout_Num shows the number of the pulses that have been transmitted.
- When EN is reset, all variables are reset.
- Definition method for segment table in M area is the same as PTOCtrl_0.

5.12 IMMEDIATE OUTPUT INSTRUCTION (HOLLYSYS_PLC_IO.LIB)

5.12.1 OutPut_Bit — Immediate Output

Example of Ladder Diagram Language


Functions

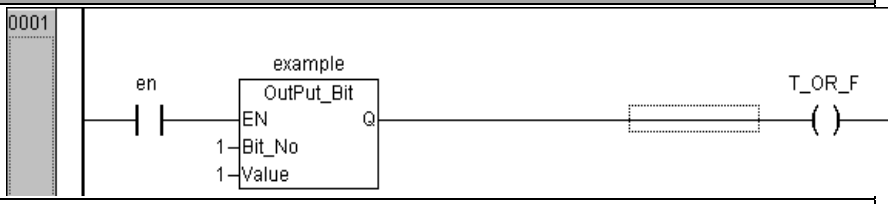
- This instruction outputs the channel status immediately before the next scanning period completed and the channel status renewed.

PARAMETER SPECIFICATIONS			
Input	Data Type	Function Description	Value
EN	BOOL	enable	0: disabled 1: enabled
Bit_No	BYTE	Number of the channel that will be immediately outputted	0-9 (correspond to Q0.0-Q1.1)
Value	BOOL	channel value	0: set Bit_No 0 1: set Bit_No 1
Output	Data Type	Function Description	Value
Q	BOOL	channel output	0: no output 1: output

APPLICATION EXAMPLE (LD and ST)

Variable Declaration

	Name	Address	Type	Initial value	Comment
0001	en		BOOL		
0002	example		OutPut_Bit		
0003	T_OR_F		BOOL		

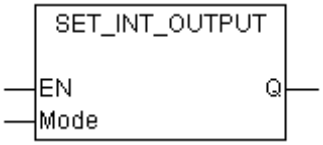
Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	0001 example(EN:= en,Bit_No:=1,Value:=1); 0002 T_OR_F:=example.Q;

Remarks on the Program:

- When EN is set, OutPut_Bit is enabled to set Q0.1 as 1. When EN is reset, OutPut_Bit will not be executed, channel value will be outputted after the scanning period.

Note:
If OutPut_Bit is used to output certain channel value in interrupts, it shall be set in the main program by Set_INT_OutPut so that the channel value will not be outputted after the scanning period to prevent conflict with the immediate output in interrupt.

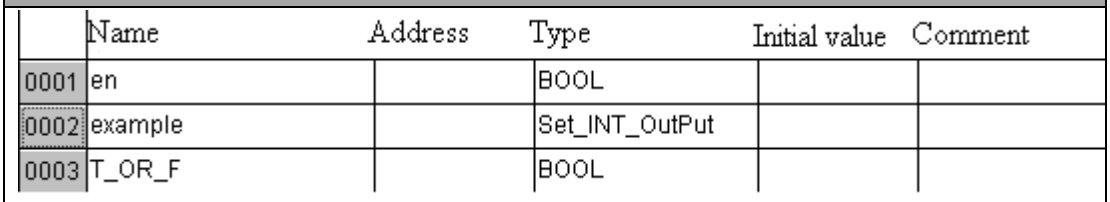
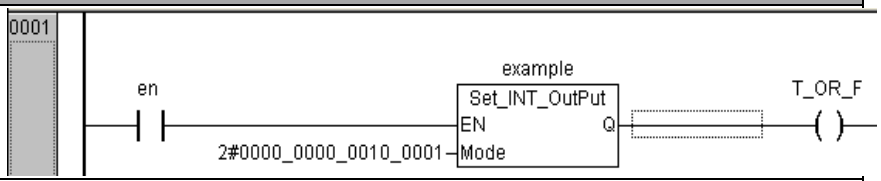
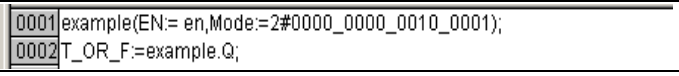
5.12.2 Set_INT_OutPut—Set Interrupt Immediate Output

Example of Ladder Diagram Language


Functions

- After Set_INT_OutPut is executed, the channel that is set in the “Mode” field will not be renewed after the scanning period. To renew the channel value, OutPut_Bit shall be called.

PARAMETER SPECIFICATIONS									
Input	Data Type	Function Description	Value						
EN	BOOL	enable	0: disabled 1: enabled						
Mode	WORD	channel not to be refreshed	Bit 0-7						
			7	6	5	4	3	2	1	0
			Q0.7	Q0.6	Q0.5	Q0.4	Q0.3	Q0.2	Q0.1	Q0.0
			Bits 8-15							
			15	14	13	12	11	10	9	8
			Invalid						Q1.1	Q1.0
			When a bit of Mode is set as 1, the corresponding channel will not be renewed after the scanning period. To renew the channel value, OutPut_Bit shall be called.							
Output	Data Type	Function Description	Value							
Q	BOOL	indicate if the setting is completed	0: not completed 1: completed							

APPLICATION EXAMPLE (LD and ST)	
Variable Declaration	
	
Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	

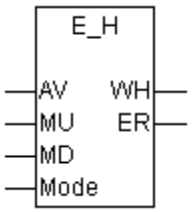
Remarks on the Program:

- When EN is set, the value of Mode is 2#0000000000100001, the first and the sixth bits from the right are 1, the corresponding channels Q0.0 and Q0.5 will not be renewed after the scanning period. To renew the channel values, OutPut_Bit shall be called.
- When EN is reset, Set_INT_OutPut will not be executed, and the values of each channel will be outputted normally after each scanning period.

Note:
Generally, when OutPut_Bit is used in the interrupts, Set_INT_OutPut shall be called in main program. If Set_INT_OutPut is not called in main program to shield the channels involved in the interrupt immediate ouputs, conflict may occur. For example, when a scanning period is over and the physical points are being renewed, if an interrupt occurs to call the OutPut_Bit function block for immediate output, conflict may arise.

5.13 ENGINEERING UNIT CONVERSION (HOLLYSYS_PLC_CNVT.LIB)

5.13.1 E_H—Convert EU to Hexadecimal

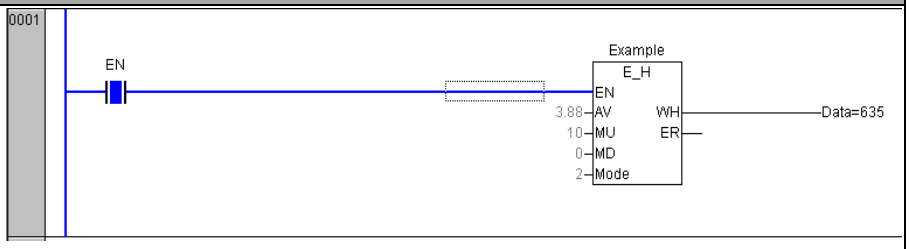
Example of Ladder Diagram Language


PARAMETER SPECIFICATIONS			
Input	Data Type	Function Description	Value
AV	REAL	engineering variable	
MU	REAL	Upper limit	
MD	REAL	Lower limit	
Mode	BYTE	conversion mode	0: normal mode, MD-MU corresponds to 0-65535 1: current 0-20mA/4-20mA corresponds to 0-3277(in LM3320) 2: voltage 0-10V corresponds to 0-1638(in LM3320)
Output	Data Type	Function Description	Value
WH	WORD	hexadecimal data output	
ER	BOOL	error code	0: correct 1: error

APPLICATION EXAMPLE (LD and ST)

Variable Declaration

	Name	Address	Type	Initial value	Comment
0001	EN		BOOL		
0002	T_OR_F		BOOL		
0003	Example		E_H		
0004	Data		WORD		

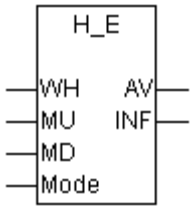
Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	0001 IF 0002 en=1 0003 THEN 0004 example(av:=3.88,mu:=10,md:=0,mode:=2); 0005 data:=example.WH; 0006 ELSE 0007 aa:=1; 0008 END_IF

Remarks on the Program:

- When EN is set, E_H conversion will be executed. When the AV is 3.88V with the MU of 10V, MD of 0V and the Mode 2 is select corresponding to 0-1638 (LM3320), then the WH is 635.
- When EN is reset, output value remains 635.

Note:
E_H shall be called through the Box with EN.

5.13.2 H_E—Convert Hexadecimal to EU

Example of Ladder Diagram Language


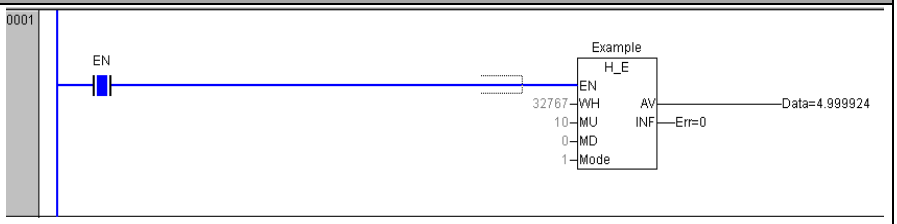
Parameter Specifications

Input	Data Type	Function Description	Value
WH	WORD	hexadecimal data input	
MU	REAL	upper limit	
MD	REAL	lower limit	
Mode	BOOL	electrical signal mode	0: current, corresponds to 0-65535 1: voltage, corresponds to 0-65535
Output	**Data Type**	**Function Description**	**Value**
AV	REAL	engineering variable output	
INF	BYTE	information code	0: normal 1: failure

Application Example (LD and ST)

Variable Declaration

	Name	Address	Type	Initial value	Comment
0001	EN		BOOL		
0002	Example		H_E		
0003	T_OR_F		BOOL		
0004	Data		REAL		
0005	Err		BYTE		

Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	0001 IF 0002 en=1 0003 THEN 0004 example(wh:=32767,mu:=10,md:=0,mode:=1); 0005 data:=example.AV; 0006 err:=example.INF; 0007 END_IF

Remarks on the Program:

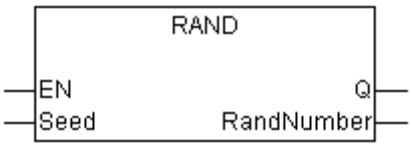
- When EN is set, H_E conversion will be executed. when the AV is 32767 with the MU of 10V, the MD of 0V and the Mode 1 is selected corresponding to 0-65535, then the output is 4.999924.
- When EN is reset, output value retains 4.999924.

Note:
H_U shall be called through the Box with EN.

5.14 RANDOM NUMBER GENERATING INSTRUCTION (HOLLYSYS_PLC_MATH.LIB)

5.14.1 Rand–Generate a Random Number

Example of Ladder Diagram Language



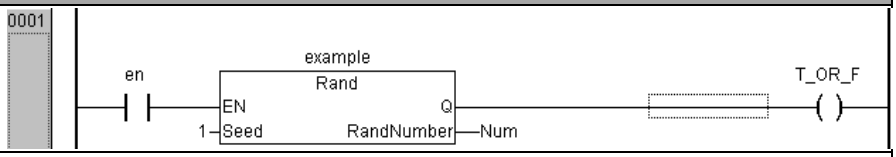
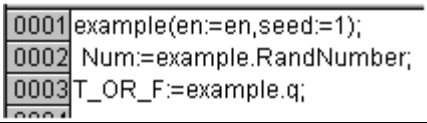
Parameter Specifications

PARAMETER SPECIFICATIONS			
Input	Data Type	Function Description	Value
EN	BOOL	enable	0: disabled 1: enabled
Seed	UINT	seed	0-65535
Output	Data Type	Function Description	Value
Q	BOOL	Indicate if the data is valid	0: invalid data 1: valid data
RandNumber	UINT	random number	0-65535

APPLICATION EXAMPLE (LD and ST)

Variable Declaration

	Name	Address	Type	Initial value	Comment
0001	en		BOOL		
0002	example		RAND		
0003	T_OR_F		BOOL		
0004	Num		UINT		

Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	

Remarks on the Program:

- When EN is set, a random number will be generated in each scanning period.
- When EN is reset, the generation of random numbers will be stopped, and the RandNumber retains the last value.

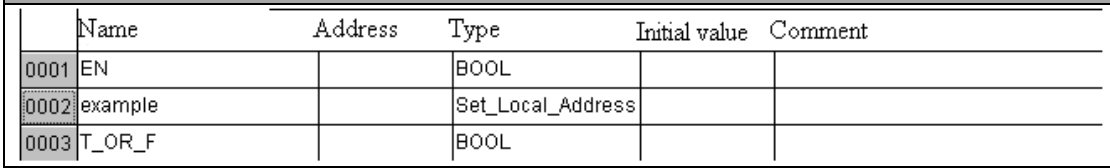
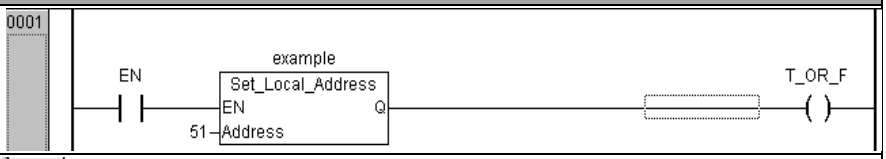
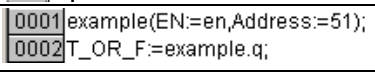
5.15 MODBUS LOCAL ADDRESS (HOLLYSYS_PLC_EX.LIB)

5.15.1 SET_LOCAL_ADDRESS—Set Modbus Local Communication Address

Example of Ladder Diagram Language


Parameter Specifications

PARAMETER SPECIFICATIONS			
Input	Data Type	Function Description	Value
EN	BOOL	enable	0: disabled 1: enabled (the first setting instruction valid after power-on or download)
Address	BYTE	Modbus local address	1-247: local number
Output	Data Type	Function Description	Value
Q	BOOL	indicate if the setting is completed	0: EN is reset 1: setting completed, retains

APPLICATION EXAMPLE (LD and ST)	
Variable Declaration	
	
Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	

Remarks on the Program:

- After power-on or download, the SET_LOCAL_ADDRESS is enabled when EN is set. When the local setting is completed, Q equals to TRUE.
- When EN is reset, Q equals to FALSE, but the local address retains the previously set value.

NOTE:
If the needed communication parameters are not set as 38400bps, 8 bits, no parity, Reset_COMM_PRMT/Reset_COMM2_PRMT shall be used to set the communication parameters, and then SET_LOCAL_ADDRESS will be called to configure the local address and establish the communication. After that, turn the RUN/STOP switch to STOP position to download, and then turn the switch to RUN to run the program.

5.15.2 GET_LOCAL_ADDRESS—Read Modbus Local Communication Address

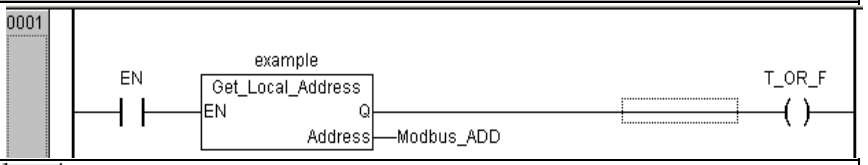
Example of Ladder Diagram Language


PARAMETER SPECIFICATIONS			
Input	Data Type	Function Description	Value
EN	BOOL	enable	0: disabled 1: enabled
Output	Data Type	Function Description	Value
Q	BOOL	Indicate if the local address is read	0: EN is reset 1: read Modbus local address and retain
Address	BYTE	Modbus local address	1-247: local number

APPLICATION EXAMPLE (LD and ST)

Variable Declaration

	Name	Address	Type	Initial value	Comment
0001	EN		BOOL		
0002	example		Get_Local_Address		
0003	T_OR_F		BOOL		
0004	Modbus_ADD		BYTE		

Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	0001 example(EN:=EN); 0002 T_OR_F:=example.Q; 0003 Modbus_ADD:=example.Address;

Remarks on the Program:

- When EN is set, GET_LOCAL_ADDRESS is enabled to read the Modbus local address, and Q equals to TRUE.
- When EN is reset, Q equals to FALSE, the local address output retains the last value.

5.16 ANALOG POTENTIOMETER (HOLLYSYS_PLC_EX.LIB)

5.16.1 POT–Read Analog Potentiometer

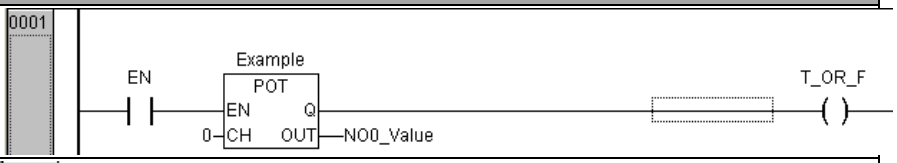
Example of Ladder Diagram Language


PARAMETER SPECIFICATIONS			
Input	Data Type	Function Description	Value
EN	BOOL	enable	0: disabled 1: enabled
CH	BYTE	channel number of analog potentiometer	0: channel 0 1: channel 1
Output	Data Type	Function Description	Value
Q	BOOL	Indicate if the data is read	0: EN is reset or no data is read 1: read data and retain
OUT	BYTE	current value of the Analog potentiometer	0-255

APPLICATION EXAMPLE (LD and ST)

Variable Declaration

	Name	Address	Type	Initial value	Comment
0001	EN		BOOL		
0002	Example		POT		
0003	T_OR_F		BOOL		
0004	NO0_value		BYTE		

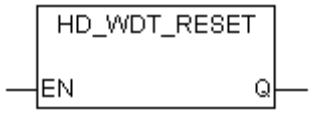
Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	0001 Example(EN:=EN,CH:=0); 0002 NO0_Value:=Example.OUT; 0003 T_OR_F:=Example.Q;

Remarks on the Program:

- When EN is set, POT is enabled to read data, Q outputs TRUE and retains, the value of OUT is the analog potentiometer value of channel 0. When analog potentiometer is dextrorotation to the maximum position, OUT equals 255.
- When EN is reset, Q equals FALSE, but OUT retains the last analog potentiometer value.

5.17 SYSTEM WATCH-DOG RESET (HOLLYSYS_PLC_EX.LIB)

5.17.1 HD_WDT_Reset–Reset the System Watch-Dog

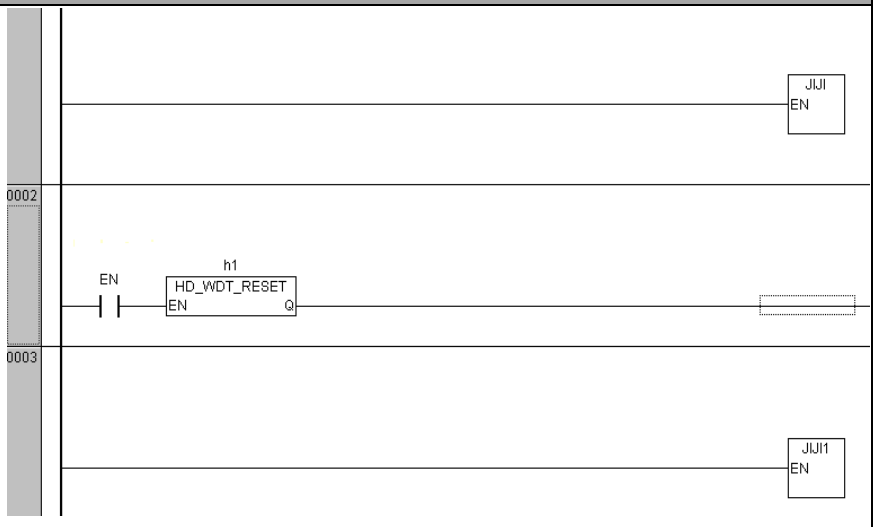
Example of Ladder Diagram Language


PARAMETER SPECIFICATIONS			
Input	Data Type	Function Description	Value
EN	BOOL	enable	0: disabled 1: enabled
Output	Data Type	Function Description	Value
Q	BOOL	indicate if the reset is completed	0: EN is resetting 1: reset completed, retain

APPLICATION EXAMPLE (LD and ST)

Variable Declaration

	Name	Address	Type	Initial value
0001	EN		BOOL	
0002	h1		HD_WDT_Reset	

Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	0001 jiji; 0002 h1(en:=1); 0003 jiji1;

Remarks on the Program:

- In the program, two subprograms will be called to executing tens of thousand times of two FOR circles. If there are no HD_WDT_Resets instructions in this two subprograms, or EN is FALSE, the PLC will hang because the scanning period is too long scanning period.

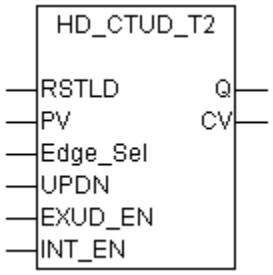
- The purpose of adding an HD_WDT_Reset instruction in the program is to clear the time recorded in PLC hardware timer, so that the System Watch-Dog will not alarm.

Note:
The System Watch-Dog gives alarms by flashing the PLC error indicator light and glittering the operation indicator light at the same time. Generally, the HD_WDT_Reset instruction is not ver necessary for the scanning period will not be too long, usually last only several ten milliseconds. When the PLC hardware clock records a time of 300ms, the Watch-Dog will give the alarm and stop the program.

5.18 MONO-PHASE COUNTER (HOLLYSYS_PLC_EX_CT.LIB)

5.18.1 HD_CTUD_T2—T2 High Speed Counter

Example of Ladder Diagram Language



PARAMETER SPECIFICATIONS			
Channel Specifications			
I0.0	Input of external high-speed count pulse		
I0.1	Input of external high-speed direction control		
Input	Data Type	Function Description	Value
RSTLD	BOOL	enable	0: disabled 1: rising edge enabled
PV	UINT	set value of the counter	0—65535
Edge_Sel	BYTE	I0.0 input pulse trigger edge selection	0: disabled 1: rising edge 2: falling edge 3: rising or falling edge
UPDN	BOOL	two functions: 1.Up/Down count selection (valid when EXUD_EN = 0) 2. reset/download in the initialization	0: down count/download CV=PV 1: up count/reset CV=0
EXUD_EN	BOOL	external I0.1 direction control enable I0.1 high voltage: up count I0.1 low voltage: down count	0: I0.1 direction control disabled 1: I0.1 direction control enabled (at the time UPDN does not control up/down count)
INT_EN	BOOL	Enable interrupts when the count value has been reached Interrupt occurs when UPDN=0: CV=0 Interrupt occurs when UPDN=1: CV=PV	0: T2 interrupt when count value has been reached disabled 1: T2 interrupt when count value has been reached enabled
Output	Data Type	Function Description	Value
Q	BOOL	indicate if the count value has been reached	0: current value CV≤PV(UPDN=1)or CV≥0(UPDN=0) 1: current value CV≥PV(UPDN=1)or CV≤0(UPDN=0)
CV	UINT	current value	0—65535

ASSOCIATED INSTRUCTION CONFLICT			
Used Instruction A	Associated Instruction B	Associated Hardware	Availability
HD_CTUD_T2	HD_DCTUD_T2	T2 internal counter, I0.0, I0.1 pulse input	Can not use instruction B

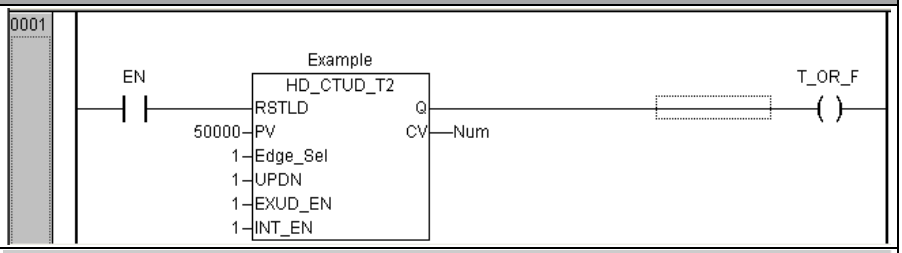
SPECIFICATION OF PORTS OCCUPIED BY THREE MONO-PHASE HIGH-SPEED COUNTERS		
High-speed Counters	External Pulse Input	External Direction Control Input
HD_CTUD_T2	I0.0	I0.1
HD_CTUD_T3	I0.2	I0.3
HD_CTUD_T7	I0.6	—

APPLICATION EXAMPLE (LD and ST)

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT | RETAIN

	Name	Address	Type	Initial value	Comment
0001	EN		BOOL		
0002	T_OR_F		BOOL		
0003	Example		HD_CTUD_T2		
0004	Num		UINT		

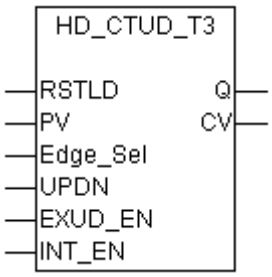
Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	0001 example(rstld:=en,pv:=50000,edge_sel:=1,updn:=1,exud_en:=1,int_en:=1); 0002 T_OR_F:=example.Q; 0003 Num:=example.CV;

Remarks on the Program:

- UPDN = 1, after the power-on reset, CV=0, because EXUD_EN = 1 at the time, UPDN does not control the up/down count.
- EXUD_EN = 1, UPDN does not control the up/down count at the time; and the I0.1 direction control will enable up count at I0.1 high voltage and down count at I0.1 low voltage.
- INT_EN = 1, for UPDN = 1, so when CV=PV, interrupt will be triggered by calling HD_TC2 interrupt event.
- When EN is set and retained, HD_CTUD_T2 will be executed to reset the counting value CV = 0, UPDN does not control up/down count, at the time it will count whenever I0.0 reaches a rising edge, Num increase or decrease by 1 (related with I0.1 external direction control), when count value reaches 50000, HD_TC2 interrupt event will be triggered, Q outputs 1.
- When EN is reset, the counting will be stopped, Q equals 0, CV retains the last value.

Note:
Refer to Appendix B for Interrupt Call.

5.18.2 HD_CTUD_T3—T3 High-Speed Counter

Example of Ladder Diagram Language


PARAMETER SPECIFICATIONS			
Channel Specification			
I0.2	Input of external high-speed count pulse		
I0.3	Input of external high-speed direction control		
Input	Data Type	Function Description	Value
RSTLD	BOOL	enable	0: disabled 1: rising edge enabled
PV	UINT	set value	0-65535
Edge_Sel	BYTE	I0.2 input pulse trigger edge selection	0: disabled 1: rising edge 2: falling edge 3: rising or falling edge
UPDN	BOOL	two functions: 1.Up/Down count select (valid when EXUD_EN = 0) 2. reset/download in the initialization	0: down count/download CV=PV 1: up count/reset CV=0
EXUD_EN	BOOL	enable external I0.3 direction control I0.3 high voltage: up count I0.3 low voltage: down count	0: I0.3 direction control disabled 1: I0.3 direction control enabled (at the time UPDN does not control up/down count)
INT_EN	BOOL	enable interrupts when the count value has been reached interrupt occurs when UPDN=0: CV=0 interrupt occurs when UPDN=1: CV=PV	0: T3 interrupt when count value has been reached disabled 1: T3 interrupt when count value has been reached enabled
Output	Data Type	Function Description	Value
Q	BOOL	Indicate if the count value has been reached	0: current value CV≤PV(UPDN=1) or CV≥0(UPDN=0) 1: current value CV≥PV(UPDN=1) or CV≤09(UPDN=0)
CV	UINT	current value	0-65535

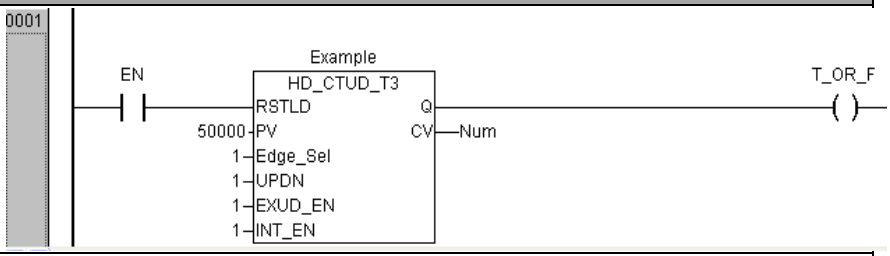
ASSOCIATED INSTRUCTION CONFLICT			
Used Instruction A	Associated Instruction B	Associated Hardware	Availability
HD_CTUD_T3	HD_DCTUD_T3	T3 internal counter, I0.2, I0.3 pulse input	Can not use instruction B
	HD_DCTUD32_T3		

APPLICATION EXAMPLE (LD and ST)

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT | RETAIN

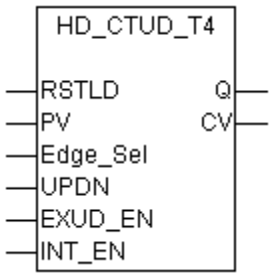
	Name	Address	Type	Initial value	Comment
0001	EN		BOOL		
0002	T_OR_F		BOOL		
0003	Example		HD_CTUD_T3		
0004	Num		UINT		

Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	0001 Example(rstld:=en,pv:=50000,edge_sel:=1,updn:=1,exud_en:=1,int_en:=1); 0002 T_OR_F:=Example.Q; 0003 Num:=Example.CV;

Remarks on the Program:

- UPDN = 1, after the power-on reset, CV=0, because EXUD_EN = 1 at the time, UPDN does not control the up/down count.
- EXUD_EN = 1, UPDN does not control the up/down count at the time, and I0.3 direction control will enable up count at I0.3 high voltage and down count at I0.3 low voltage.
- INT_EN = 1, for UPDN = 1, so when CV=PV, interrupt will be triggered by calling HD_TC3 interrupt event.
- When EN is set and retained, HD_CTUD_T3 will be executed to reset the counting value CV = 0, UPDN does not control the up/down count, at the time it will count whenever I0.2 reaches a rising edge, Num increase or decrease by 1 (related with I0.3 external direction control), when count value reaches 50000, HD_TC3 interrupt event will be triggered, Q outputs 1.
- When EN is reset, the counting will be stopped, Q equals 0, CV retains the last value.

5.18.3 HD_CTUD_T4—T4 Normal Counter

Example of Ladder Diagram Language


PARAMETER SPECIFICATIONS			
Channel Specification			
I0.4	Input of external count pulse		
I0.5	Input of external direction control		
Input	Data Type	Function Description	Value
RSTLD	BOOL	enable	0: disabled 1: rising edge enabled
PV	UINT	set value	0—65535
Edge_Sel	BYTE	I0.4 input pulse trigger edge selection	0: disabled 1: rising edge 2: falling edge 3: rising or falling edge
UPDN	BOOL	two functions: 1.Up/Down count selection (valid when EXUD_EN = 0) 2. reset/download in the initialization	0: down count/download CV=PV 1: up count/reset CV=0
EXUD_EN	BOOL	enable external I0.5 direction control I0.5 high voltage: up count I0.5 low voltage: down count	0: I0.5 direction control disabled 1: I0.5 direction control enabled (at the time UPDN does not control up/down count)
INT_EN	BOOL	enable interrupts when the count value has been reached interrupt occurs when UPDN=0: CV=0 interrupt occurs when UPDN=1: CV=PV	0: T3 interrupt when count value has been reached disabled 1: T3 interrupt when count value has been reached enable
Output	Data Type	Function Description	Value
Q	BOOL	indicates if the count value has been reached	0: current value CV≤PV(UPDN=1) or CV≥0(UPDN=0) 1: current value CV≥PV(UPDN=1) or CV≤0(UPDN=0)
CV	UINT	current value	0-65535

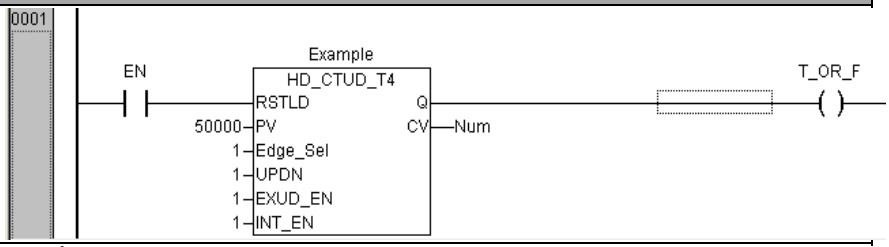
ASSOCIATED INSTRUCTION CONFLICT			
Used Instruction A	Associated Instruction B	Associated Hardware	Availability
HD_CTUD_T4	HD_DCTUD_T4	T4 internal counter, I0.4, I0.5 pulse input	Can not use instruction B
	HD_DCTUD32_T3	T4 internal counter	

APPLICATION EXAMPLE (LD and ST)

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT | RETAIN

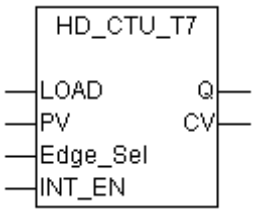
	Name	Address	Type	Initial value	Comment
0001	EN		BOOL		
0002	T_OR_F		BOOL		
0003	Example		HD_CTUD_T4		
0004	Num		UINT		

Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	0001 Example(rstld:=en,pv:=50000,edge_sel:=1,updn:=1,exud_en:=1,int_en:=1); 0002 T_OR_F:=Example.Q; 0003 Num:=Example.CV;

Remarks on the Program:

- UPDN = 1, after the power-on reset, CV=0, because EXUD_EN = 1 at the time, UPDN does not control the up/down count.
- EXUD_EN = 1, UPDN does not control the up/down count, and I0.5 direction control will enable up count at I0.5 high voltage and down count at I0.5 low voltage.
- INT_EN = 1, for UPDN = 1, so when CV=PV, interrupt will be triggered by calling HD_TC4 interrupt event.
- When EN is set and retained, HD_CTUD_T4 will be executed to reset the counting value CV = 0, UPDN does not control the up/down count， at the time it will count whenever I0.4 reaches a rising edge, Num increase or decrease by 1 (related with I0.5 external direction control), when the count value reaches 50000, HD_TC4 interrupt event will be triggered, Q outputs 1.
- When EN is reset, the counting will be stopped, Q equals 0, CV retains the last value.

5.18.4 HD_CTU_T7—T7 High Speed Counter

Example of Ladder Diagram Language


PARAMETER SPECIFICATIONS			
Channel Specification			
I0.6	input of external high-speed count pulse		
Input	Data Type	Function Description	Value
LOAD	BOOL	enable	0: disabled 1: rising edge enabled
PV	UINT	set value	0-65535
Edge_Sel	BYTE	I0.6 input pulse trigger edge Selection	0: disabled 1: rising edge 2: falling edge 3: rising or falling edge
INT_EN	BOOL	enable count value reaches interrupt enabled	0: T7 Count value reaches interrupt disabled 1: T7 Count value reaches interrupt enable
Output	Data Type	Function Description	Value
Q	BOOL	indicates if the count value has been reached	0: current value CV≤PV(UPDN=1) or CV≥0(UPDN=0) 1: current value CV≥PV(UPDN=1) or CV≤0(UPDN=0)
CV	UINT	current value	0-65535

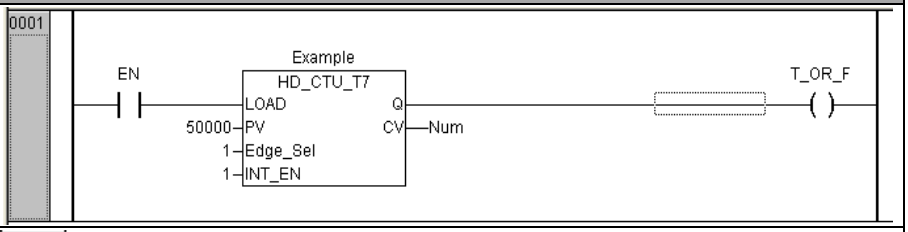
ASSOCIATED INSTRUCTION CONFLICT			
Used Instruction A	Associated Instruction B	Associated Hardware	Availability
HD_CTU_T7	HD_DCTUD_T2	I0.6 Pulse Input and I0.6 External Reset pulse Input	Can not use B External Reset pulse Input
	Fast_ExINT	I0.6 Pulse Input and I0.6 Fast External Interrupt 3 Pulse Input	Can not use B Fast External Interrupt 3 Pulse Input Channel
	Fast_ExINT_E		
	HD_TIMER_T7	T7	Can not use instruction B

APPLICATION EXAMPLE (LD and ST)

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT | RETAIN

	Name	Address	Type	Initial value	Comment
0001	EN		BOOL		
0002	T_OR_F		BOOL		
0003	Example		HD_CTU_T7		
0004	Num		UINT		

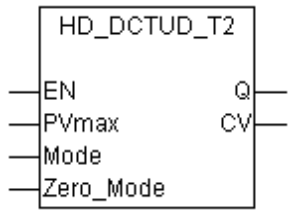
Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	0001 example(load:=en,pv:=50000,edge_sel:=1,int_en:=1); 0002 T_OR_F:=example.Q; 0003 NUM:=example.CV;

Remarks on the Program:

- Edge_Sel = 1, count by 1 when I0.6 input pulse s a rising edge.
- INT_EN = 1, when the count value has been reached, INT_EN will trigger the HD_TC7 interrupt event.
- When EN is set and retained, HD_CTUD_T7 will be executed to reset the count value CV = 0， at the time it will count whenever I0.6 reaches a rising edge, Num increase by 1, when the count value is greater than or equals to 50000, the HD_TC7 interrupt event will triggered, Q outputs 1.
- When EN is reset, the counting will be stopped, Q equals 0 and CV retains the last value.

5.19 BI-PHASE COUNTER (HOLLYSYS_PLC_EX_DCT.LIB)

5.19.1 HD_DCTUD_T2—T2 High Speed Bi-phase Counter

Example of Ladder Diagram Language


PARAMETER SPECIFICATIONS			
Channel Specification			
I0.0	external high-speed count pulse input A		
I0.1	external high-speed count pulse input B		
I0.6	external reset pulse input		
Input	Data Type	Function Description	Value
EN	BOOL	enable	0: the counting disabled and the CV reset 1: the counting enabled at CV=0. CV increases in up count and decreases in down count
Mode	BYTE	select mode	See Table 5-19-1 for details
Zero_Mode	BYTE	enable reset pulse	See Table 5-19-1 for details
PVmax	UINT	set value	range 0-65535
Output	Data Type	Function Description	Value
Q	BOOL	indicate if the count is completed	0: up count CV<PV or down count CV>PV; 1: up count CV≥PV or down count CV≤PV;
CV	UINT	current count value	0-65535

Parameter	Parameter Specification
Mode	00: Disabled. 01: Count on the rising edge at phase A and phase B. 10: Count on the falling edge at phase A and phase B. 11: Count of I0.0 or I0.1 on the rising or falling edge.
Zero_Mode	00: Disabled. PV is valid. When CV≥PV, CV is reset (I0.6 invalid). 01: Reset pulse on the rising edge at I0.6 triggers CV reset (PVmax invalid). 10: Reset pulse on the falling edge at I0.6 triggers CV reset (PVmax invalid). 11: Reset pulse on the rising or falling edge at I0.6 triggers the reset of CV (PVmax invalid).

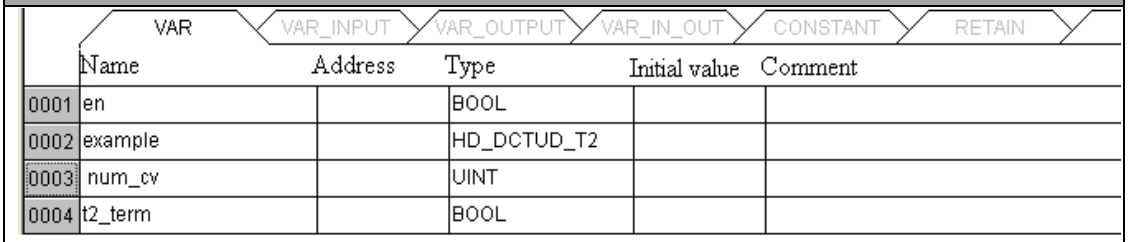
Table 5-19-1 Detailed Specifications of Mode and Zero_Mode Parameters

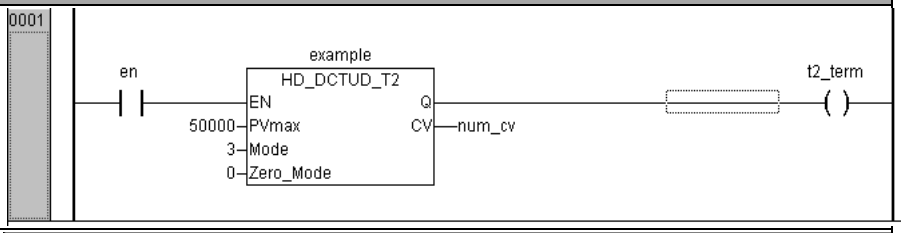
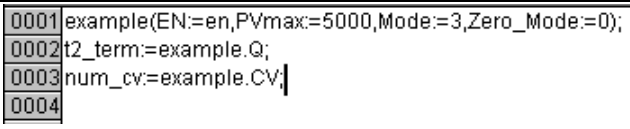
ASSOCIATED INSTRUCTION CONFLICT			
Used Instruction A	Associated Instruction B	Associated Hardware	Availability
HD_DCTUD_T2	HD_CTUD_T2	T2 internal counter, I0.0, I0.1 pulse input	Can not use instruction B
	HD_CTU_T7	I0.6 input of external reset pulse, I0.6 pulse input	Can not use instruction B when using the external reset pulse input A.
	Fast_ExINT	I0.6 input of external reset pulse, I0.6 pulse input of fast external interrupt 3	Can not use instruction B for the fast external interrupt of the 3 pulse input channels when using the external reset pulse input A.
	Fast_ExINT_E		

SPECIFICATIONS OF PORT OCCUPIED BY TWO HIGH-SPEED Bi-phase counter			
High-speed Counter	A Input of External Pulse	B Input of External Pulse	Input of External Reset pulse
HD_DCTUD_T2	I0.0	I0.1	I0.6
HD_DCTUD_T3	I0.2	I0.3	I0.7

APPLICATION EXAMPLE (LD and ST)

Variable Declaration



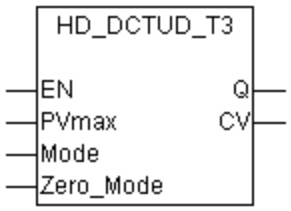
Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	

Remarks on the Program:

- When Mode=3, the rising or falling edge of I0.0 or I0.1 will trigger the counting.
- When Zero_Mode=0, I0.6 external reset pulse input will be invalid. When CV≥PVmax, CV = 0.
- When EN is set and retained (on the rising edge), this instruction will be executed to reset count value CV=0, the rising or falling edge of I0.0 or I0.1 will trigger the counting. When CV≥PVmax and CV = 0, the counting will be restarted.
- When EN is reset, the counting will be stopped, Q equals to 0 and CV retains the last value.

5.19.2 HD_DCTUD_T3—T3 High Speed Bi-phase Counter

Example of Ladder Diagram Language



PARAMETER SPECIFICATIONS			
Channel Specification			
I0.2	external high-speed count pulse input A		
I0.3	external high-speed count pulse input B		
I0.7	external reset pulse input		
Input	Data Type	Function Description	Value
EN	BOOL	enable	0: the counting disabled and the CV reset 1: the counting enabled at CV=0. CV increases in up count and decreases in down count
Mode	BYTE	select mode	See Table 5-19-2 for details.
Zero_Mode	BYTE	enable reset pulse	See Table 5-19-2 for details.
PVma	UINT	set value	range 0-65535
Output	Data Type	Function Description	Value
Q	BOOL	Indicate if the count is completed	0: up count CV<PV or down count CV>PV; 1: up count CV≥PV or CV≤PV down count;
CV	UINT	current count value	0-65535

Parameter	Parameter Specification
Mode	00: Disabled. 01: Count on the rising edge at phase A and phase B. 10: Count on the falling edge at phase A and phase B. 11: Count of I0.0 or I0.1 on the rising or falling edge.
Zero_Mode	00: Disabled. PV is valid. When CV≥PV, CV is reset (I0.7 invalid) 01: Reset pulse on the rising edge at I0.7 triggers CV reset (PVmax invalid) 10: Reset pulse on the falling edge at I0.7 triggers CV reset (PVmax invalid) 11: Reset pulse on the rising or falling edge at I0.7 triggers CV reset (PVmax invalid)

Table 5-19-2 Detailed Specification of Mode and Zero_Mode Parameters

ASSOCIATED INSTRUCTION CONFLICT			
Used Instruction A	Associated Instruction B	Associated Hardware	Availability
HD_DCTUD_T2	HD_CTUD_T2	T2 internal counter, I0.0, I0.1 pulse input	Can not use instruction B
	HD_CTU_T7	I0.6 external reset pulse input, I0.6 pulse input	Can not use instruction B when using the external reset pulse input A.
	Fast_ExINT	I0.6 external reset pulse input, I0.6 pulse input of fast external interrupt 3	Can not use instruction B for the fast external Interrupt of 3 pulse input channels when using the external reset pulse input A.
	Fast_ExINT_E		

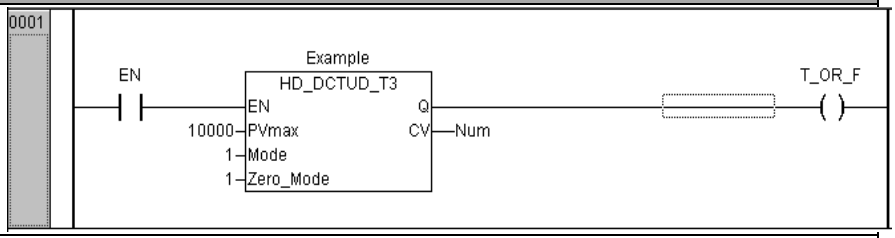
SPECIFICATION OF PORT OCCUPIED BY TWO HIGH-SPEED BI-PHASE COUNTERS			
High Speed Counter	A Input of External Pulse	B Input of External Pulse	Input of External Reset pulse
HD_DCTUD_T2	I0.0	I0.1	I0.6
HD_DCTUD_T3	I0.2	I0.3	I0.7

APPLICATION EXAMPLE (LD and ST)

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT | RETAIN

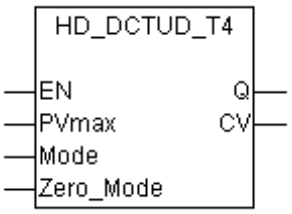
	Name	Address	Type	Initial value	Comment
0001	EN		BOOL		
0002	T_OR_F		BOOL		
0003	Example		HD_DCTUD_T3		
0004	Num		UINT		

Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	0001 example(EN:=en,pvmax:=10000,mode:=1,zero_mode:=1); 0002 T_OR_F:=example.Q; 0003 NUM:=example.CV;

Remarks on the Program:

- When Mode=3, the rising or falling edge of I0.3 will trigger the counting.
- When Zero_Mode = 1, PVmax will be invalid, when I0.7 reaches the rising ede, the reset of CV will be triggered.
- When EN is set and retained (on the rising edge), this instruction is executed to reset the count value CV=0, the rising or falling edge of I0.3 will trigger the counting. When the rising edge at I0.7 triggers the reset of CV, counting will be restarted.
- When EN is reset, the counting will be stopped, Q equals to 0 and CV retains the last value.

5.19.3 HD_DCTUD_T4—T4 Bi-phase Counter

Example of Ladder Diagram Language


PARAMETER SPECIFICATIONS			
Channel Specification			
I0.4	external count pulse input A		
I0.5	external count pulse input B		
I1.0	external reset pulse input		
Input	Data Type	Function Description	Value
EN	BOOL	enable	0: the counting disabled and the CV reset 1: the counting enabled at CV=0. CV increases in up count and decreases in down count
Mode	BYTE	select mode	See Table 5-19-3 for details.
Zero_Mode	BYTE	enable reset pulse	See Table 5-19-3 for details.
PVma	UINT	set value	range 0-65535
Output	Data Type	Function Description	Value
Q	BOOL	indicate if the count is completed	0: up count CV<PV or down count CV>PV; 1: up count CV≥PV or down count CV≤PV;
CV	UINT	current count value	0-65535

Parameter	Parameter Specification
Mode	00: Disabled 01: Count on the rising edge at phase A and phase B 10: Count on the falling edge at phase A and phase B 11: Count of I0.4 or I0.5 on the rising or falling edge
Zero_Mode	00: Disabled. PV is valid. When CV≥PV, CV is reset (I1.0 invalid) 01: Reset pulse on the rising edge at I1.0 triggers CV reset (PVmax invalid) 10: Reset pulse on the falling edge at I1.0 triggers CV reset (PVmax invalid) 11: Reset pulse on the rising or falling edge at I1.0 triggers the reset of CV (PVmax invalid)

Table 5-19-3 Detailed Specifications of Mode and Zero_Mode Parameters

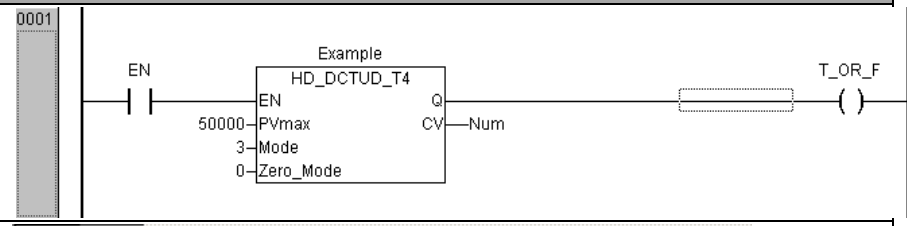
ASSOCIATED INSTRUCTION CONFLICT			
Used Instruction A	Associated Instruction B	Associated Hardware	Availability
HD_DCTUD_T4	HD_CTUD_T4	T4 internal counter, I0.4, I0.5 pulse input	Can not use instruction B
	HD_DCTUD32_T3	T4 internal counter	
	Fast_ExINT	I1.0 external reset pulse input, I1.0 pulse input of fast external interrupt 1	Can not use instruction B for the east external interrupt of 1 pulse input channel when using the external reset pulse input A.
	Fast_ExINT_E		

APPLICATION EXAMPLE (LD and ST)

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT | RETAIN

	Name	Address	Type	Initial value	Comment
0001	EN		BOOL		
0002	T_OR_F		BOOL		
0003	Example		HD_DCTUD_T4		
0004	Num		UINT		

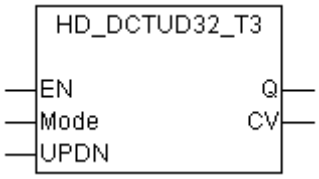
Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	0001 example(EN:=en,pvmax:=50000,mode:=3,zero_mode:=0); 0002 T_OR_F:=example.Q; 0003 NUM:=example.CV;

Remarks on the Program:

- When Mode=3, the rising or falling edge of I0.4 or I0.5 will trigger the counting.
- When Zero_Mode=0, input of I1.0 external reset pulse will be invalid. When CV≥PVmax, CV = 0.
- When EN is set and retained, this instruction is executed to reset the count value CV=0, the rising or falling edge of I0.4 or I0.5 will trigger the counting. When CV≥PVmax, CV is 0, the counting will be restarted.
- When EN is reset, the counting will be stopped, Q equals to 0 and CV retains the last value.

5.20 BI-PHASE 32-BIT COUNTER (HOLLYSYS_PLC_EX_DCT32.LIB)

5.20.1 HD_DCTUD32_T3 – 32-bit High Speed Counter

Example of Ladder Diagram Language


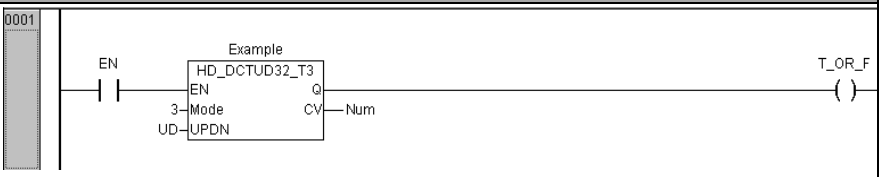
PARAMETER SPECIFICATIONS			
Channel Specification			
I0.2	external high-speed count pulse input A		
I0.3	external count pulse input B		
Input	Data Type	Function Description	Value
EN	BOOL	enable	0: disabled 1: rising edge enabled
Mode	BYTE	select mode	0: disabled 1: the rising edge or falling edge at I0.3 2: the rising edge or falling edge at I0.2 3: the rising edge or falling edge at I0.2 or I0.3
UPDN	BOOL	direction of the carry (low 16-bit to high16-bit) (no need to set direction of the carry for counting value between -32767 – 32768)	0: up 1: down
Output	Data Type	Function Description	Value
Q	BOOL	indicate if the counting has been started	0: counting stopped 1: counting started
CV	DWORD	current count value	-2147483648-2147483647

ASSOCIATED INSTRUCTION CONFLICT			
Used Instruction A	Associated Instruction B	Associated Hardware	Availability
HD_DCTUD32_T3	HD_CTUD_T3	T3 internal counter, I0.2, I0.3 pulse input	Can not use instruction B
	HD_CTUD_T4	T4 internal counter	
	HD_DCTUD_T3	T3 internal counter, I0.2, I0.3 pulse input	
	HD_DCTUD_T4	T4 internal counter	

APPLICATION EXAMPLE (LD and ST)

Variable Declaration

	Name	Address	Type	Initial value
0001	EN		BOOL	
0002	T_OR_F		BOOL	
0003	Example		HD_DCTUD32_T3	
0004	UD		BOOL	
0005	Num		DWORD	

Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	0001 Example(EN:=EN,mode:=3,UPDN:=UD); 0002 Num:=Example.CV; 0003 T_OR_F:=Example.Q;

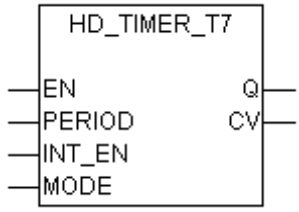
Remarks on the Program:

- When Mode=3, the rising or falling edge of I0.2 or I0.3 will trigger the counting.
- UD controls direction of the carry from low 16-bit to high 16-bit.
- When EN is set and retained, this instruction will be executed to reset the count value CV=0, the rising or falling edge of I0.2 or I0.3 will trigger the counting. When CV reaches 32767, if another up count is needed the UD shall be ensured to be equal to 1, otherwise error will occur in the counting.
- If EN is reset, the counting will be stopped, Q equals to 0 and CV retains the last value.

5.21 INTERRUPT TIMER (HOLLYSYS_PLC_EX_TIMER.LIB)

5.21.1 HD_TIMER_T7—Interrupt Timer

Example of Ladder Diagram Language



PARAMETER SPECIFICATIONS			
Input	Data Type	Function Description	Value
EN	BOOL	enable	0: disabled 1: rising edge enabled
PERIOD	DWORD	set value(us)	100-2680000
INT_EN	BOOL	enable T7 Interrupt	0: set time value not to generate interrupt 1: set time value to generate interrupt
MODE	BOOL	select mode	0: rising edge triggered 1: level triggered
Output	Data Type	Function Description	Value
Q	BOOL	indicate if the setting has been completed	0: EN is reset 1: setting completed and retained
CV	DWORD	current value(ns)	0-2680000000

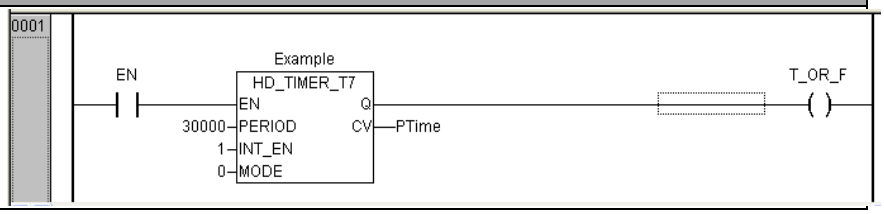
ASSOCIATED INSTRUCTION CONFLICT			
Used Instruction A	Associated Instruction B	Associated Hardware	Availability
HD_TIMER_T7	HD_CTU_T7	T7	Can not use instruction B

APPLICATION EXAMPLE (LD and ST)

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT | RETAIN

	Name	Address	Type	Initial value	Comment
0001	EN		BOOL		
0002	Example		HD_TIMER_T7		
0003	PTime		DWORD		
0004	T_OR_F		BOOL		

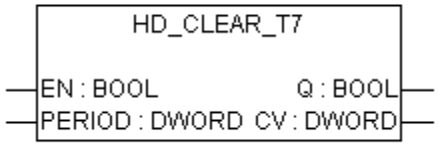
Programming Language	Program
Ladder Diagram (LD)	

Structured Text (ST)	0001 example(en:=en,period:=30000,int_en:=1,mode:=0); 0002 T_OR_F:=example.q; 0003 PTIME:=example.CV;

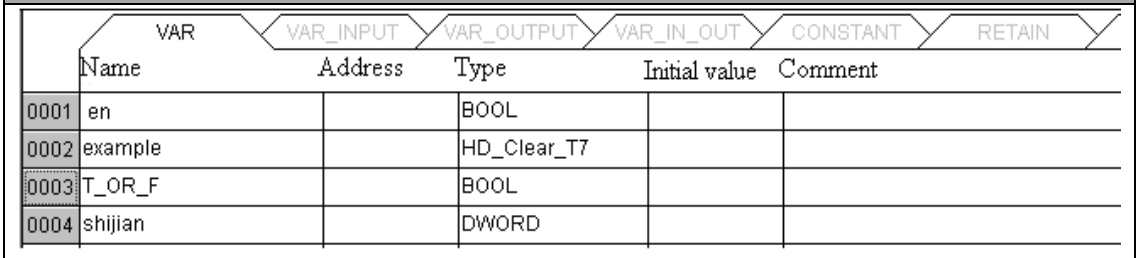
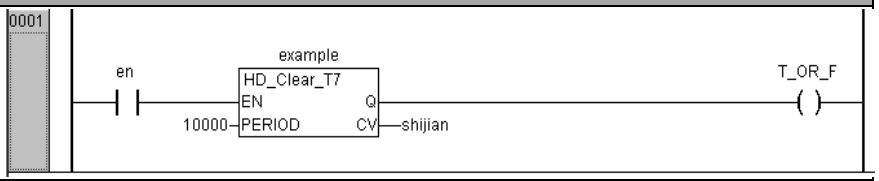
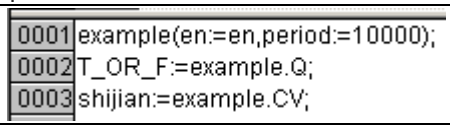
Remarks on the Program:

- When EN is set and retained (on the rising edge), Q equals 1; because INT_EN is 1, an HD_TC7 interrupt event will be triggered every 30000us. Call the execution program of HD_TC7 interrupt event through the system events of PowerPro, CV will show values of the current passed time in ns.
- When EN is reset, the generating of HD_TC7 interrupt event will be stopped, Q equals to 0.

5.21.2 HD_CLEAR_T7—Reset Timer

Example of Ladder Diagram Language


PARAMETER SPECIFICATIONS			
Input	Data Type	Function Description	Value
EN	BOOL	enable	0: disabled 1: enabled
PERIOD	DWORD	Set maximum time (us)	100-2680000
Output	Data Type	Function Description	Value
Q	BOOL	indicate if the reset has been completed	0: not completed 1: completed
CV	DWORD	passed time before clear (ns)	0-2680000000

APPLICATION EXAMPLE (LD and ST)	
Variable Declaration	
	
Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	

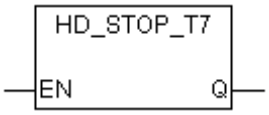

```
0001 example(en:=en,period:=10000);
0002 T_OR_F:=example.Q;
0003 shijian:=example.CV;
```

Remarks on the Program:

- Operated with HD_TIMER_T7 and fast external interrupt, this instruction is used to measure pulse width. For example, start up the HD_TIMER_T7 and the fast external interrupt in the main program, then define I0.6 to capture the rising edge and falling edge of fast external interrupt, call HD_CLEAR_T7 in the interrupt. When the rising edge at I0.6 and the interrupt is generated, the instruction can be called to initialize T7, and interrupt will be re-generated when the falling edge of I0.6 occurs again, the pulse width will be stored to CV (in ns), and T7 will be reset.

- The values of PERIOD in HD_CLEAR_T7 and HD_TIMER_T7 should be the same, otherwise the value of CV may appear errors.

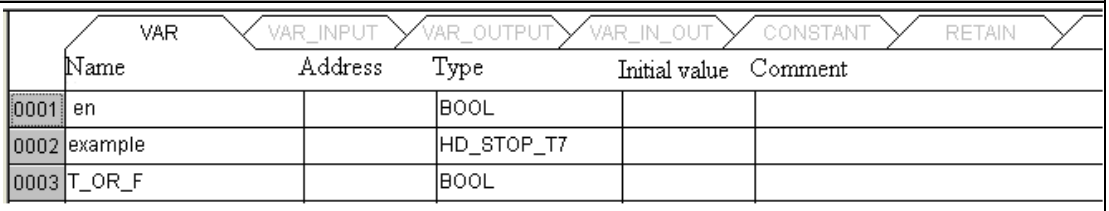
5.21.3 HD_STOP_T7—Stop Timer

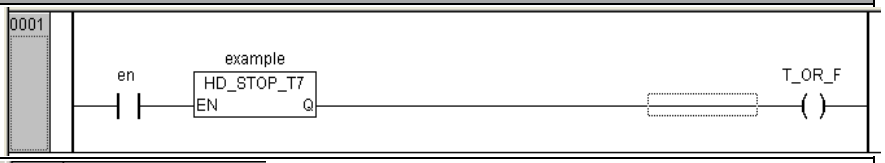
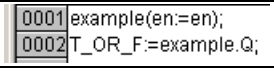
Example of Ladder Diagram Language


PARAMETER SPECIFICATIONS			
Input	Data Type	Function Description	Value
EN	BOOL	enable	0: disabled 1: enabled
Output	Data Type	Function Description	Value
Q	BOOL	Indicate the stop	0: not stopped or EN disabled 1: stopped

APPLICATION EXAMPLE (LD and ST)

Variable Declaration



Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	

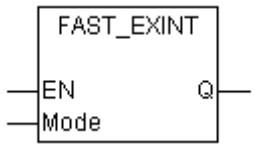

```
0001 example(en:=en);
0002 T_OR_F:=example.Q;
```

Remarks on the Program:

- Operated with HD_TIMER_T7, this instruction is used to stop HD_STOP_T7.
- T7 timer is started by the calling of HD_TIMER_T7 and stopped by the execution of HD_STOP_T7.

5.22 EXTERNAL INTERRUPT (HOLLYSYS_PLC_EX_EXINT.LIB)

5.22.1 Fast_ExINT—Fast External Interrupt

Example of Ladder Diagram Language


PARAMETER SPECIFICATIONS			
Channel Specification			
I1.0	fast external interrupt 1 pulse input (inapplicable to LM3104, LM3105)		
I0.7	fast external interrupt 2 pulse input		
I0.6	fast external interrupt 3 pulse input		
Input	Data Type	Function Description	Value
EN	BOOL	enable	0: disabled 1: rising edge enabled
Mode	BYTE	Set interrupt mode	See table 5-22-1 for details.
Output	Data Type	Function Description	Value
Q	BOOL	Indicate if the interrupt is correctly set	0: incorrect setting 1: correct setting

Bits							
7	6	5	4	3	2	1	0
interrupt 3		interrupt 2		interrupt 1		null	
Setting of Interrupt n (n=1、2、3)							
Disabled: 0 0 Rising edge trigger: 0 1 Falling edge trigger: 1 0 Rising or falling edge trigger: 1 1							

Table 5-22-1 Detailed Specifications of Mode Parameters

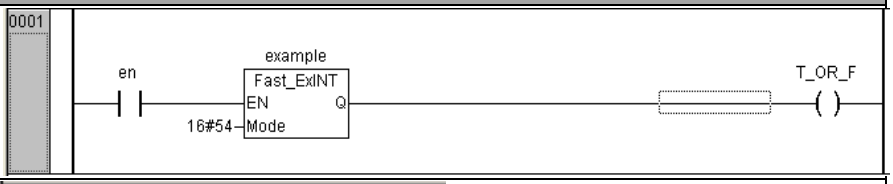
ASSOCIATED INSTRUCTION CONFLICT			
Used Instruction A	Associated Instruction B	Associated Hardware	Availability
Fast_ExINT	HD_CTU_T7	I0.6 pulse input of fast external interrupt 3 I0.6 pulse input	Can not use instruction B when using the pulse input A of fast external interrupt 3
	HD_DCTUD_T2	I0.6 pulse input of fast external interrupt 3 I0.6 external reset pulse input	Can not use the external reset pulse input channel B when using the pulse input of fast external interrupt 3
	HD_DCTUD_T3	I0.7 pulse input of fast external interrupt 2 I0.7 external reset pulse input	Can not use the external reset pulse input channel B when using the pulse input of fast external interrupt 2
	HD_DCTUD_T4	I1.0 pulse input of fast external interrupt 1 I1.0 external reset pulse input	Can not use the external reset pulse input channel B when using the pulse input of fast external interrupt 1

APPLICATION EXAMPLE (LD and ST)

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT | RETAIN

	Name	Address	Type	Initial value	Comment
0001	en		BOOL		
0002	example		Fast_ExINT		
0003	T_OR_F		BOOL		

Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	0001 example(en:=en,mode:=16#54); 0002 T_OR_F:=example.q;

Remarks on the Program:

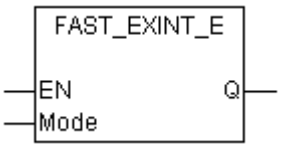
- When EN is set and retained, Q equals 1; as shown in the table below, whenever I0.6, I0.7, I1.0 reach a rising edge, the setting value of Mode is 16#54 (2#01 01 01 00), the three fast external interrupts are all set as rising edge triggers.

Mode = 16#54							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
interrupt 3		interrupt 2		interrupt 1		interrupt 0	
0	1	0	1	0	1	null	

- Call the corresponding interrupt program by system event of PowerPro to trigger the related fast external interrupt.
- When EN is reset, I0.6, I0.7, I1.0 stop receiving interrupt pulse, and Q equals 0.

Note:
FAST_ExINT is applicable to modules that have a 28K program memory space.

5.22.2 Fast_ExINT_E—Fast External Interrupt

Example of Ladder Diagram Language


PARAMETER SPECIFICATIONS			
Channel Specification			
I1.1	fast external interrupt 0 pulse input (inapplicable to LM3104, LM3105)		
I1.0	fast external interrupt 1 pulse input (inapplicable to LM3104, LM3105)		
I0.7	fast external interrupt 2 pulse input		
I0.6	fast external interrupt 3 pulse input		
Input	Data Type	Function Description	Value
EN	BOOL	enable	0: disabled 1: rising edge enabled
Mode	BYTE	Set interrupt mode	See table 5-22-2 for details.
Output	Data Type	Function Description	Value
Q	BOOL	indicate if the interrupt is correctly set	0: incorrect setting 1: correct setting

Bits							
7	6	5	4	3	2	1	0
interrupt 3		interrupt 2		interrupt 1		interrupt 0	
Setting of Interrupt n (n=1, 2, 3)							
Disabled: 0 0 Rising edge trigger: 0 1 Falling edge trigger: 1 0 Rising or falling edge trigger: 1 1							

Table 5-22-2 Detailed Specification of Mode Parameters

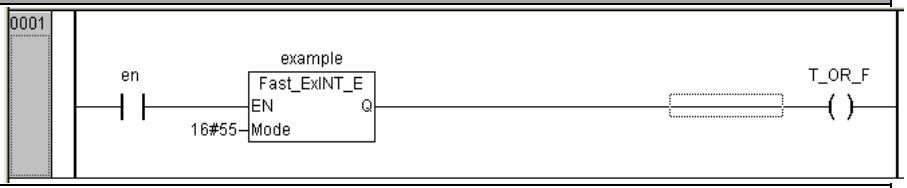
ASSOCIATED INSTRUCTION CONFLICT			
Used Instruction A	Associated Instruction B	Associated Hardware	Availability
Fast_ExINT_E	HD_CTU_T7	I0.6 pulse input of fast external interrupt 3 I0.6 pulse input	Can not use instruction B when using the pulse input A of fast external interrupt 3
	HD_DCTUD_T2	I0.6 pulse input of fast external interrupt 3 I0.6 external reset pulse input	Can not use the external reset pulse input channel B when using the pulse input of fast external interrupt 3
	HD_DCTUD_T3	I0.7 pulse input of fast external interrupt 2 I0.7 external reset pulse input	Can not use the external reset pulse input channel B when using the pulse input of fast external interrupt 2
	HD_DCTUD_T4	I1.0 pulse input of fast external interrupt 1 I1.0 external reset pulse input	Can not use the external reset pulse input channel B when using the pulse input of fast external interrupt 1

APPLICATION EXAMPLE (LD and ST)

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT | RETAIN

	Name	Address	Type	Initial value	Comment
0001	en		BOOL		
0002	example		Fast_ExINT_E		
0003	T_OR_F		BOOL		

Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	example(en:=en,mode:=16#55); T_OR_F:=example.Q;

Remarks on the Program:

- The setting value of Mode is 16#55 (2#01 01 01 01) (see the table below), the four fast external interrupts are set as rising edge triggers.

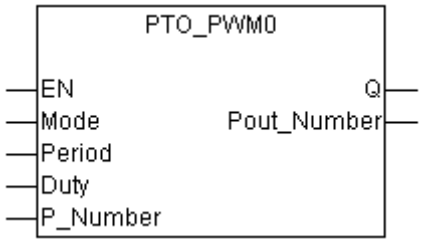
Mode = 16#55							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
interrupt 3		interrupt 2		interrupt 1		interrupt 0	
0	1	0	1	0	1	0	1

- When EN is set and retained (on the rising edge), Q equals to 1; whenever I0.6, I0.7, I1.0, I1.1 reach a rising edge, the corresponding interrupt program will be called by the system event of PowerPro to trigger the related fast external interrupt.
- When EN is reset, I0.6, I0.7, I1.0, I1.1 stop receiving interrupt pulse, and Q equals 0.

NOTE:
Fast_ExINT_E is applicable to modules that have a 120K program memory space.

5.23 PULSE OUTPUT (HOLLYSYS_PLC_EX_PT.LIB)

5.23.1 PTO_PWM0—PTO/PWM Pulse Output

Example of Ladder Diagram Language


PARAMETER SPECIFICATIONS			
Channel Specification			
Q1.1	high speed pulse output (Mode=0, 1, 2, 3)		
Q1.0	high speed pulse complementary output (Mode=2, 3)		
Input	Data Type	Function Description	Value
EN	BOOL	enable	0: disabled 1: rising edge enabled
Mode	BYTE	select the mode of pulse output (when Mode=1、3, P_Number invalid, Pout_Number outputs 0)	0: Q1.1 outputs PTO 1: Q1.1 outputs PWM 2: Q1.1 and Q1.0 complementary outputs PTO 3: Q1.1 and Q1.0 complementary outputs PWM
Period	DWORD	Set period (μs)	See table 5-23-1 for details.
Duty	BYTE	duty	0-100 (always 50 for PTO)
P_Number	DWORD	number of pulse	0-4294967295
Output	Data Type	Function Description	Value
Q	BOOL	indicate the pulse transmission	0: transmission stopped 1: in transmission
Pout_Number	DWORD	number of pulse transmitted	0-4294967295

Module Type	PTO	PWM
LM3106	50-335000	50-335000
LM3106A	20-335000	10-335000
LM3108	50-335000	50-335000

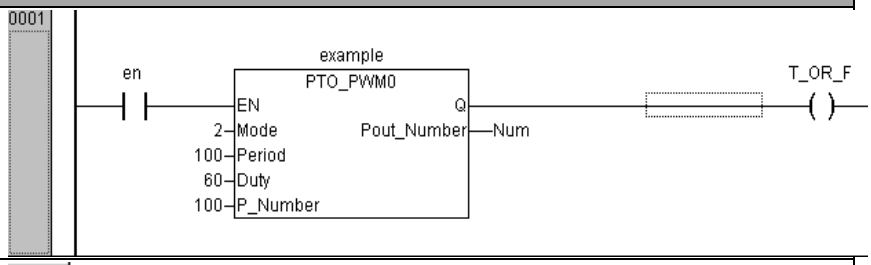
Table 5-23-1 Specification of Period Parameters

ASSOCIATED INSTRUCTION CONFLICT			
Used Instruction A	Associated Instruction B	Associated Hardware	Availability
PTO_PWM0	PTO_PWM0_Run	T12	Can not use instruction B.
	PTOCtrl_0		

APPLICATION EXAMPLE (LD and ST)

Variable Declaration

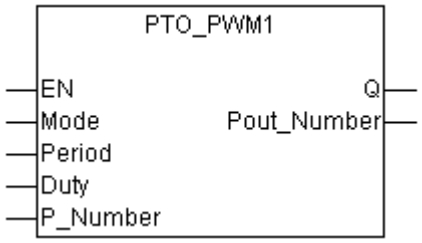
0001	en		BOOL		
0002	example		PTO_PWM0		
0003	Num		DWORD		
0004	T_OR_F		BOOL		

Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	0001 example(en:=en,mode:=2,period:=100,duty:=60,p_number:=100); 0002 T_OR_F:=example.Q; 0003 Num:=example.Pout_Number;

Remarks on the Program:

- When EN is set and retained, Q1.1 and Q1.0 will transmit pulse in reverse voltage (Mode=2) at a pulse frequency of 1s / 100μs = 10K (Period=100), Num is the number of pulse transmitted, Q equals 1 and retains, totally 10000 pulses will be transmitted in a round and it will stop when transmission is over, Q equals 0.
- If EN is reset during the transmission, Q1.1 and Q1.0 stop transmitting pulses, Q equals 0, and Num retains the current value.
- When Mode=2, PTO will be outputted with 50 duty (at the time, duty shall always be50, no matter what is the input value of Duty).

5.23.2 PTO_PWM1—PTO/PWM Pulse Output

Example of Ladder Diagram Language


PARAMETER SPECIFICATIONS			
Channel Specification			
Q0.3	high speed pulse out (Mode=0, 1)		
Input	Data Type	Function Description	Value
EN	BOOL	enable	0: disabled 1: rising edge enabled
Mode	BYTE	Select the mode of pulse output (when Mode=1, P_Number invalid, Pout_Number outputs 0)	0: Q0.3 outputs PTO 1: Q0.3 outputs PWM
Period	DWORD	set period (μs)	See table 5-23-2 for details.
Duty	BYTE	duty	0-100 (always 50 for PTO mode)
P_Number	DWORD	number of pulse	0-4294967295
Output	Data Type	Function Description	Value
Q	BOOL	Indicate the pulse transmission	0: transmission stopped 1: in transmission
Pout_Number	DWORD	number of pulse transmitted	0-4294967295

Module Type	PTO	PWM
LM3104-CDT	50-2630000	50-2630000
LM3106-CDT	50-2630000	50-2630000
LM3108-CDT	50-2630000	50-2630000

Table 5-23-2 Specification of Period Parameters

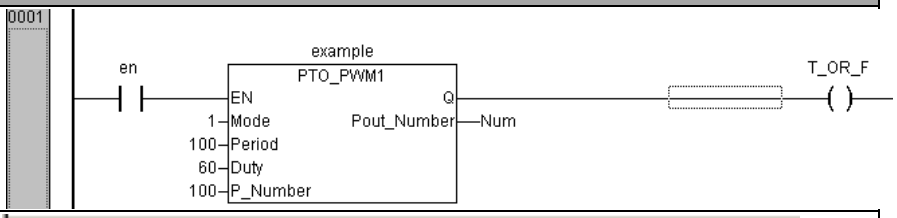
ASSOCIATED INSTRUCTION CONFLICT			
Used Instruction A	Associated Instruction B	Associated Hardware	Availability
PTO_PWM1	PTO_PWM1_Run	T8, T13	Can not use instruction B
	PTOCtrl_1	T13	

APPLICATION EXAMPLE (LD and ST)

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT | RETAIN

	Name	Address	Type	Initial value	Comment
0001	en		BOOL		
0002	example		PTO_PWM1		
0003	T_OR_F		BOOL		
0004	Num		DWORD		

Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	0001 example(en:=en,mode:=1,period:=100,duty:=60,p_number:=100); 0002 T_OR_F:=example.Q; 0003 Num:=example.Pout_Number;

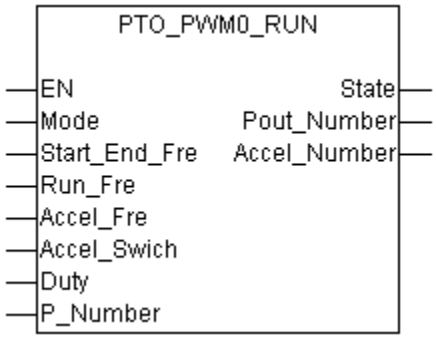
Remarks on the Program:

- When EN is set and retained, Q0.3 starts to transmit pulses at a pulse frequency of 1s / 100μs = 10K and 60 Duty, Num is the number of pulse transmitted, Q equals 1 and retains.
- When EN is reset, Q0.3 stop transmitting pulses, Q equals 0, Num retains the current value.
- When Mode=1, PTO will be outputted with 50 duty.

5.24 ACCELERATED AND DECELERATED PULSE OUTPUT (HOLLYSYS_PLC_EX_PTRUN.LIB)

5.24.1 PTO_PWM0_RUN—PTO_PWM Accelerated and Decelerated Pulse Out

Example of Ladder Diagram Language



Function Description

- If Start_End_Fre>Run_Fre, decelerated running and in any mode run as decelerated---constant speed.
- If Start_End_Fre<Run_Fre, accelerated running, and in PTO mode run as accelerated—constant speed—decelerated (accelerated and decelerated are symmetrical), in PWM mode run as accelerated—constant speed.
- If Start_End_Fre=Run_Fre, constant speed, and Accel_Fre, Accel_Swich are invalid at the time.

PARAMETER SPECIFICATIONS			
Input	Data Type	Function Description	Value
EN	BOOL	enable	0: disabled 1: rising edge enabled
Mode	BYTE	Select the mode of pulse output (when Mode=1, 3, P_Number invalid, Pout_Number outputs 0)	0: Q1.1 outputs PTO 1: Q1.1 outputs PWM 2: Q1.1 and Q1.0 complementary outputs PTO 3: Q1.1 and Q1.0 complementary outputs PWM
Start_End_Fre	DWORD	Start & end frequency(Hz)	See table 5-24-1 for details.
Run_Fre	DWORD	Running frequency(Hz)	See table 5-24-1 for details.
Accel_Fre	DWORD	accelerated frequency (Hz, positive)	
Accel_Swich	BOOL	frequency change switch	0: not changed 1: changed
Duty	BYTE	duty	0-100(always 50 in PTO mode)
P_Number	DWORD	number of pulse	0-4294967295
Output	Data Type	Function Description	Value
State	BOOL	Indicate the pulse transmission	0: transmission stopped 1: in transmission
Pout_Number	DWORD	number of pulse transmitted (valid in mode 0 and 2)	0-4294967295
Accel_Number	DWORD	number of pulse transmitted in accelerated stage	0-4294967295

Module Type	PTO	PWM
LM3106	3-20000	3-20000
LM3106A	3-50000	3-100000
LM3108	3-20000	3-20000

Table 5-24-1 Specifications of Start_End_Fre and Run_Fre parameters

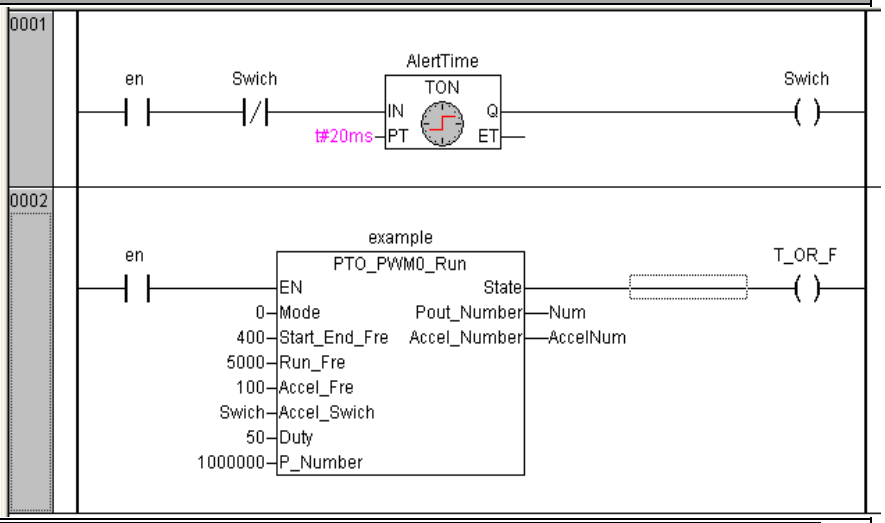
ASSOCIATED INSTRUCTION CONFLICT			
Used Instruction A	Associated Instruction B	Associated Hardware	Availability
PTO_PWM0_Run	PTO_PWM0	T12	Can not use instruction B.

APPLICATION EXAMPLE (LD and ST)

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT | RETAIN

	Name	Address	Type	Initial value	Comment
0001	en		BOOL		
0002	T_OR_F		BOOL		
0003	AlertTime		TON		
0004	Num		DWORD		
0005	Swich		BOOL		
0006	example		PTO_PWM0_Run		
0007	AccelNum		DWORD		

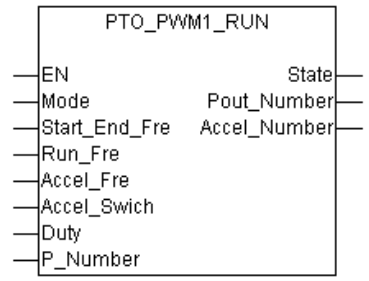
Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	0001 AlertTime(in:=en AND NOT swich,pt:=t#20ms); 0002 SWICH:=AlertTime.Q; 0003 0004 example(en:=en,mode:=0,start_End_Fre:=400,Run_Fre:=5000,Accel_Fre:=100, 0005 Accel_Swich:=Swich,duty:=50,P_Number:=1000000); 0006 T_OR_F:=example.State; 0007 NUM:=example.Pout_Number; 0008 AccelNum:=example.Accel_Number;

Remarks on the Program:

- When EN is set and retained, Q1.1 starts to transmit pulses at a pulse frequency of 400Hz, Num is the number of pulses transmitted, AccelNum is 0. After this, the pulse frequency will increase by 100Hz each 20ms until pulse frequency reaches 5000Hz, AccelNum is the number of pulse transmitted in the accelerated stage (affected by scanning period); and then run with constant speed in 5000Hz, because Mode=0, it will decelerate when the rest pulses are equal to the pulses transmitted in accelerated stage until the transmission of 1000000 pulses is finished, stop transmitting, when the next scanning period arrives, AccelNum is reset to 0, Numretains until EN reaches rising edge again.
- When EN is reset, Q1.1 stop transmitting, Q equals 0.
- When Mode=0, PTO will be outputted with 50 Duty (Duty will always be 50, no matter what is the input value of Duty).

5.24.2 PTO_PWM1_RUN—PTO_PWM Accelerated and Decelerated Pulse Output

Example of Ladder Diagram Language



Function Description

- If Start_End_Fre>Run_Fre, decelerated running and in any mode run as decelerated---constant speed.
- If Start_End_Fre<Run_Fre, accelerated running, and in PTO mode run as accelerated—constant speed—decelerated (accelerated and decelerated are symmetrical), in PWM mode run as accelerated—constant speed.
- If Start_End_Fre=Run_Fre, constant speed, and Accel_Fre, Accel_Swich are invalid at the time.

PARAMETER SPECIFICATIONS			
Channel Specification			
Q0.3	high speed pulse output (Mode=0, 1)		
Input	Data Type	Function Description	Value
EN	BOOL	enable	0: disabled 1: rising edge enabled
Mode	BYTE	Select the mode of pulse output (when Mode=1, 3, P_Number invalid, Pout_Number outputs 0)	0: Q0.3 outputs PTO 1: Q0.3 outputs PWM
Start_End_Fre	DWORD	Start & end frequency (Hz)	See table 5-24-2 for details.
Run_Fre	DWORD	Running frequency (Hz)	See table 5-24-2 for details.
Accel_Fre	DWORD	accelerated frequency (Hz, positive)	
Accel_Swich	BOOL	frequency change switch	0: not changed 1: changed
Duty	BYTE	duty	0-100(always 50 in PTO mode)
P_Number	DWORD	number of pulse	0-4294967295
Output	Data Type	Function Description	Value
State	BOOL	indicate the pulse transmission	0: transmission stopped 1: in transmission
Pout_Number	DWORD	number of pulse transmitted (valid in mode 0 and 2)	0-4294967295
Accel_Number	DWORD	number of pulse transmitted in accelerated stage	0-4294967295

Module Type	PTO	PWM
LM3104	1-20000	1-20000
LM3106	1-20000	1-20000
LM3106A	1-50000	1-100000
LM3108	1-20000	1-20000

Table 5-24-2 Specification of Start_End_Fre and Run_Fre parameters

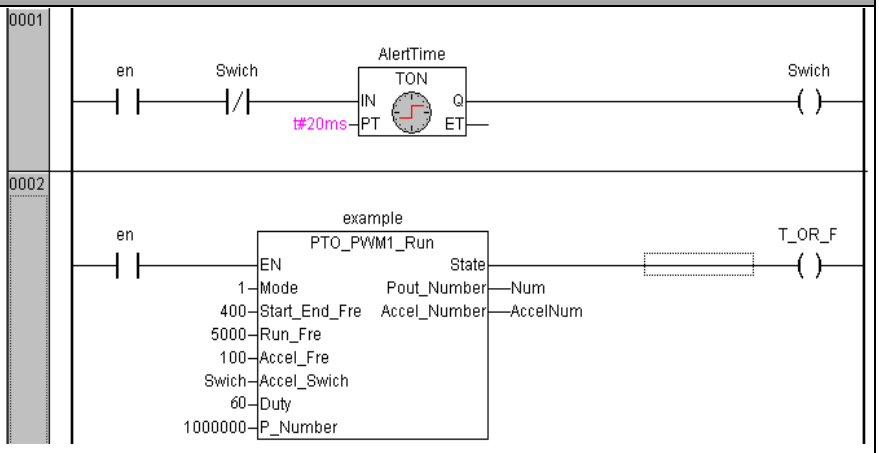
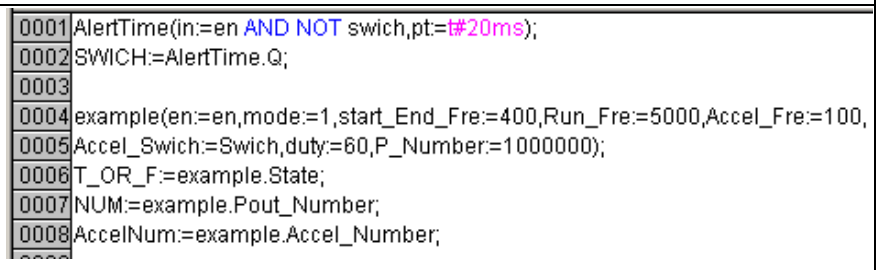
ASSOCIATED INSTRUCTION CONFLICT			
Used Instruction A	Associated Instruction B	Associated Hardware	Availability
PTO_PWM1_Run	PTO_PWM1	T8, T13	Can not use instruction B
	PTOCtrl_1	T13	

APPLICATION EXAMPLE (LD and ST)

Variable Declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT | RETAIN

	Name	Address	Type	Initial value	Comment
0001	en		BOOL		
0002	T_OR_F		BOOL		
0003	AlertTime		TON		
0004	Num		DWORD		
0005	Swich		BOOL		
0006	example		PTO_PWM1_Run		
0007	AccelNum		DWORD		

Programming Language	Program
Ladder Diagram (LD)	
Structured Text (ST)	


```
0001 AlertTime(in:=en AND NOT swich,pt:=t#20ms);
0002 SWICH:=AlertTime.Q;
0003
0004 example(en:=en,mode:=1,start_End_Fre:=400,Run_Fre:=5000,Accel_Fre:=100,
0005 Accel_Swich:=Swich,duty:=60,P_Number:=1000000);
0006 T_OR_F:=example.State;
0007 NUM:=example.Pout_Number;
0008 AccelNum:=example.Accel_Number;
```

Remarks on the Program:

- When EN is set and retained (on the rising edge), Q0.3 starts to transmit pulses at the pulse frequency of 400Hz, Num is 0, AccelNum is 0. After that, pulse frequency increases by 100Hz each 20ms until pulse frequency reaches 5000Hz, AccelNum is the number of pulse transmitted in accelerated stage (affected by scanning period), and then run at 5000Hz.
- When EN is reset, Q0.3 stop transmitting, Q equals 0.
- When Mode=1, P_Number=1000000 is invalid, PWM will be outputted with its Duty set to 60, Q0.3 will transmit until EN is reset to 0.

Method of controlling the acceleration and deceleration Curve of Stepper Motor

Stepper motor can control position and speed by open loop control without feedbacks. However, because there is no feedback of load positions, stepper motor must respond to the change of excitation accurately. If the frequency of excitation is not correctly selected, motor cannot move to the new position, then there will be a permanent error between actual load position and expected position, out-of step or over-swing will occur. Hence, out-of step or over-swing shall be avoided in the open loop control system of stepper motor.

Out-of step or over-swing occurs respectively at the startups and stops of the motor. Generally the limit startup frequency is low and the required running frequency is high. If motor starts up directly at the required running frequency which is higher than the limit startup frequency, motor cannot start up normally, out-of step may occur, or more seriously, the motor will not be able to start. After the system is running, if the sending of pulses stops immediately at the termination, the motor will pass balanced position for the system. If the load inertia is too large, motor will stop at the next balanced position near termination.

To avoid the out-of step and over-swing, the acceleration and deceleration control shall be executed at the starts and stops of stepper motors, as shown in figure 5-24-1, where “ls” indicates startup frequency, “hs” indicates running frequency.

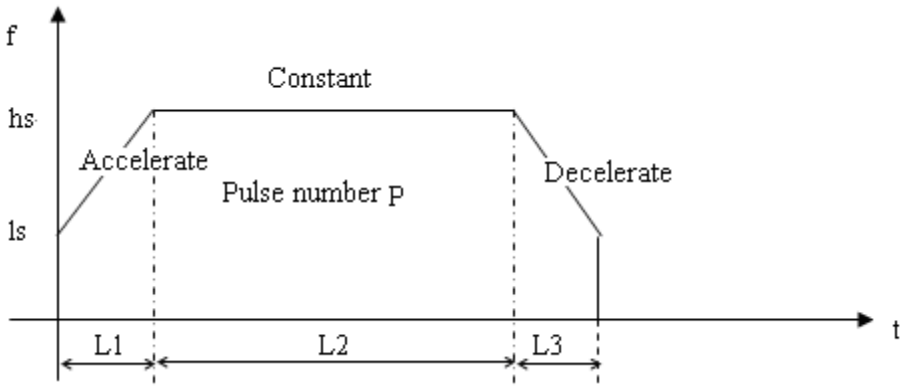


Figure 5-24-1 Acceleration and Deceleration Curve of Stepper Motor

L2 is running, L1 is acceleration, L3 is deceleration. According to the definition of “out-of step”, motor will out-of step if frequency change at L1, L3 is larger than the change of responding frequency and motor may stop and influence system. So, the correct control of acceleration and deceleration must be executed in stepper motors.

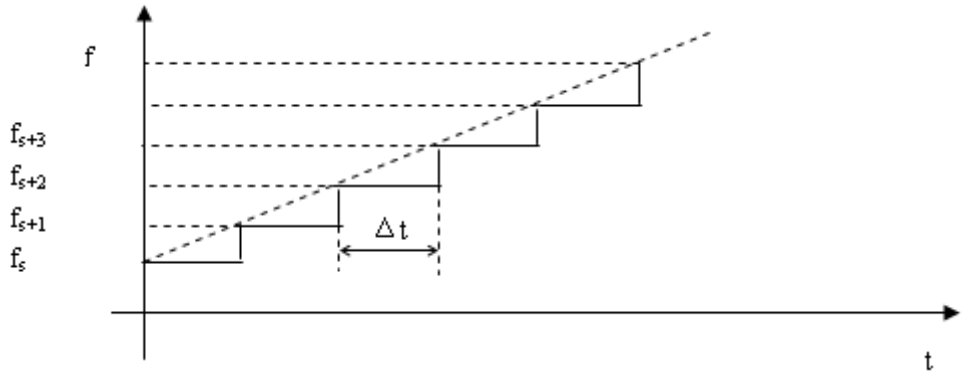


Figure 5-24-2 Discretization Process of Speed

In PLC acceleration and deceleration controlling processes, discrete method is generally used to approach the ideal curve. In acceleration or deceleration, the speed does not change continuously, but change at different stages. The Δt at each stage shall be determined to approach the required slope of acceleration and deceleration, as shown in figure 5-24-2. The faster the acceleration, the less the Δt . Δt can be determined theoretically or experimentally according to the principle of high speed without out-of step.

Appendix

A

Appendix A

A.1 LM INSTRUCTION FAST-CHECK LIST

Basic Instructions

Instruction Type	Instruction Specification	Instruction Name	Library
Arithmetic Operators	Addition	ADD (FUN)	none
	Multiplication	MUL (FUN)	
	Subtraction	SUB (FUN)	
	Division	DIV (FUN)	
	Modulo Division	MOD (FUN)	
Assignment Operator	Assignment	MOVE (FUN)	none
Logical Operators	Bitwise AND	AND (FUN)	none
	Bitwise OR	OR (FUN)	
	Bitwise XOR	XOR (FUN)	
	Bitwise NOT	NOT (FUN)	
Bit-Shift Operators	Left-shift	SHL (FUN)	none
	Right-shift	SHR (FUN)	
	Rotation to the left	ROL (FUN)	
	Rotation to the right	ROR (FUN)	
Selection Operators	Binary selection	SEL (FUN)	none
	Returns the greater of 2 values	MAX (FUN)	
	Returns the lesser of 2 values	MIN (FUN)	
	Limits the value range	LIMIT (FUN)	
	Select a value out of a group of values	MUX (FUN)	
Comparison Operators	Greater than	GT (FUN)	none
	Less than	LT (FUN)	
	Greater or equal	GE (FUN)	
	Lesser or equal	LE (FUN)	
	Equal	EQ (FUN)	
	Not equal	NE (FUN)	

Instruction Type	Instruction Specification	Instruction Name	Library
Type Conversion Instructions	BOOL_TO conversions	BOOL_TO_<TYPE>(FUN)	None
	BYTE_TO conversions	BYTE_TO_<TYPE>(FUN)	
	DATE_TO conversions	DATE_TO_<TYPE>(FUN)	
	DINT_TO conversions	DINT_TO_<TYPE>(FUN)	
	DT_TO conversions	DT_TO_<TYPE>(FUN)	
	DWORD_TO conversions	DWORD_TO_<TYPE>(FUN)	
	INT_TO conversions	INT_TO_<TYPE>(FUN)	
Type Conversion Functions	WORD_TO conversions	WORD_TO_<TYPE>(FUN)	none
	REAL_TO conversions	REAL_TO_<TYPE>(FUN)	
	SINT_TO conversions	SINT_TO_<TYPE>(FUN)	
	STRING_TO conversions	STRING_TO_<TYPE>(FUN)	
	TIME_TO conversions	TIME_TO_<TYPE>(FUN)	
	TOD_TO conversions	TOD_TO_<TYPE>(FUN)	
	UDINT_TO conversions	UDINT_TO_<TYPE>(FUN)	
	UINT_TO conversions	UINT_TO_<TYPE>(FUN)	
	USINT_TO conversions	USINT_TO_<TYPE>(FUN)	
	Conversion from REAL to INT	TRUNC(FUN)	
Elementary Mathematical Functions	Returns the absolute value	ABS(FUN)	none
	Returns the square root	SQRT(FUN)	
	Returns the natural logarithm of a number	LN(FUN)	
	Returns the logarithm of a number in base 10	LOG(FUN)	
	Returns the exponentiation function	EXP(FUN)	
	Returns the sine	SIN(FUN)	
	Returns the cosine	COS(FUN)	
	Returns the tangent	TAN(FUN)	
	Returns the arc sine	ASIN(FUN)	
	Returns the arc cosine	ACOS(FUN)	
	Returns the arc tangent	ATAN(FUN)	
	Exponentiation	EXPT(FUN)	
Address Operators	Address of the operand	ADR(FUN)	none
	Content operator	^(FUN)	
	Returns the bit offset	BITADR(FUN)	
	Internal index for a POU	INDEXOF (FUN)	
	Number of bytes	SIZEOF (FUN)	
Calling Operators	Calling a function block or a program	CAL(FUN)	none
Initialization Operator	Initialize retain variables	INI(FUN)	none

Instruction Type	Instruction Specification	Instruction Name	Library
Bit-string Operators	Concatenation of two strings	CONCAT(FUN)	Standard.lib
	Delete string	DELETE(FUN)	
	Find string	FIND(FUN)	
	Insert string	INSERT(FUN)	
	Left initial string of given size of string	LEFT(FUN)	
	String length	LEN(FUN)	
	Returns a partial string from within a string	MID(FUN)	
	Replace a partial string from a larger string with a third string	REPLACE(FUN)	
	Right initial string of given size of string	RIGHT(FUN)	
Library version check	Get library version	Version_Util(FUN)	Util.lib
Software version check	Get Software version message	SyslibGetVersion2300(FUN)	SysLibC16x.lib
Callback function	Register a definite callback	SysCallbackRegister(FUN)	SysLibCallback.lib
	Un-register a definite callback	SysCallbackUnregistrer(FUN)	
Check instruction	Check for out-of-range errors	CheckBounds(FUN)	Check.lib
	Check whether the divisor is 0 in BYTE division	CheckDivByte(FUN)	
	Check whether the divisor is 0 in WORD division	CheckDivWord(FUN)	
	Check whether the divisor is 0 in DWORD division	CheckDivDword(FUN)	
	Check whether the divisor is 0 in REAL division	CheckDivReal(FUN)	
	Check whether subrange variables of INT are over range	CheckRangeSigned(FUN)	
	Check whether subrange variables of Unsigned are over range	CheckRangeUnsigned(FUN)	
BCD Conversion	Convert a byte in BCD format into an INT value	BCD_TO_INT(FUN)	Util.lib
	Convert an INT value into a byte in BCD format	INT_TO_BCD(FUN)	
bit/byte functions	Extract the Nth bit of input	EXTRACT(FUN)	Util.lib
	Pack 8 bits into 1 byte	PACK(FUN)	
	Set Nth bit value	PUTBIT(FUN)	
	Unpack 1 byte into 8 bits	UNPACK(FB)	
Mathematical Functions	Local derivation	DERIVATIVE(FB)	Util.lib
	Integral	INTEGRAL(FB)	
	Min, Max, Average values in INT format	STATISTICS_INT(FB)	
	Min, Max, Average values in REAL format	STATISTICS_REAL(FB)	
	Variance	VARIANCE(FB)	
Controller	Proportion controller	P(FB)	Util.lib
	PD controller	PD(FB)	
	PID controller	PID(FB)	
	PID controller with fixed cycle time	PID_FIXCYCLE(FB)	
Signal generators	Simulate a turn signal	BLINK(FB)	Util.lib
	Generate some periodic functions	GEN(FB)	

Instruction Type	Instruction Specification	Instruction Name	Library
SFC action control	SFC action control	SFCActionControl(FB)	Iecsfc.lib
Function Manipulators	Map an input signal on a characteristic curve	CHARCURVE(FB)	Util.lib
	Limit the slope of a value to a certain value	RAMP_INT(FB)	
	Limit the slope of a value to a certain value	RAMP_REAL(FB)	
Analog Monitors	Hysteresis	HYSTERESIS(FB)	Util.lib
	Monitor whether the input is within a set range	LIMITALARM(FB)	
Bistable Function Block	Bistable function, set dominant	SR(FB)	Standard.lib
	Bistable function, reset dominant	RS(FB)	
Trigger	Rising edge detection	R_TRIG(FB)	Standard.lib
	Falling edge detection	F_TRIG(FB)	
Counter	Counter up	CTU(FB)	Standard.lib
	Counter down	CTD(FB)	
	Counter up down	CTUD(FB)	
Timer	Timer pulse	TP(FB)	Standard.lib
	Timer on delay	TON(FB)	
	Timer off delay	TOF(FB)	
	Real time clock	RTC(FB)	

Expansion Instructions

Instruction Type	Instruction Specification	Instruction Name	Library
Analog modules	Analog input	Analog_IN(FB)	Hollysys_PLC_Analog.lib
	Analog output	Analog_OUT(FB)	
RS232 port communication setting	RS232 free port communication parameter setting	Set_COMM_PRMT(FB)	Hollysys_PLC_Comm.lib
	Send RS232 Free Port Communication Data	COMM_SEND(FB)	
	Receive RS232 Free Port Communication Data	COMM_RECEIVE(FB)	
	Reset RS 232 setting	Reset_COMM_PRMT(FB)	
RS485 port communication setting	RS485 free port communication parameter setting	Set_COMM2_PRMT(FB)	Hollysys_PLC_Comm2.lib
	Send RS485 Free Port Communication Data	COMM2_SEND(FB)	
	Receive RS485 Free Port Communication Data	COMM2_RECEIVE(FB)	
	Reset RS 485 setting	Reset_COMM2_PRMT	
Profibus-DP module call	Profibus-DP slave module call	DP_Slave(FB)	Hollysys_PLC_DPSlave.lib
EtherNet module call	EtherNet module call	EtherNet_TCP(FB)	Hollysys_PLC_EtherNet.lib
Positive and negative action optional PID controller	PID controller	PID2(FB)	Hollysys_PLC_Util.lib
Modbus check instruction	Generate Modbus CRC code	Generate_CRC(FB)	Hollysys_PLC_Modbus_CRC.lib
Hardware real time clock setting	Set RTC (in DT data type)	Set_HD_RTC(FB)	Hollysys_PLC_HdRtc.lib
	Set RTC (in TP data type)	Set_HD_RTC_X(FB)	
	Get RTC date/time/day	Get_HD_RTC(FB)	
HDRTC alarm instruction	Get HDRTC alarm time/day	Get_HDRTC_ALM(FB)	Hollysys_PLC_HdRtcALM.lib
	Set HDRTC alarm time/day	Set_HDRTC_ALM(FB)	
HDRTC alarm of LM3104/5	Get HDRTC alarm time/day	Get_HDRTC_ALM(FB)	Hollysys_PLC_HdRtcALM_N.lib
	Set HDRTC alarm time/day	Set_HDRTC_ALM(FB)	
Multi-segment Pulse Transmission	Channel 1.1 Multi-segment Pulse Transmission	PTOCtrl_0(FB)	Hollysys_PLC_PTOCtrl.lib
	Channel 0.3 Multi-segment Pulse Transmission	PTOCtrl_1(FB)	
Immediate Output	Immediate Output	OutPut_Bit(FB)	Hollysys_PLC_IO.lib
	Set Interruption Immediate Output	Set_INT_OutPut(FB)	
Engineering	Convert EU to Hexadecimal	E_H(FB)	Hollysys_PLC_Cnvt.lib

Instruction Type	Instruction Specification	Instruction Name	Library
Unit Conversion	Convert Hexadecimal to EU	H_E(FB)	
Generate A Random Number	Generate A Random Number	Rand(FB)	Hollysys_PLC_Math.lib
Modbus local address	Set Modbus local communication address	SET_LOCAL_ADDRESS (FB)	Hollysys_PLC_Ex.lib
	Get Modbus local communication address	GET_LOCAL_ADDRESS (FB)	
Analog Potentiometer	Get Analog Potentiometer value	POT(FB)	Hollysys_PLC_Ex.lib
System Watch-Dog Reset	System Watch-Dog Reset	HD_WDT_Reset(FB)	Hollysys_PLC_Ex.lib
Single-phase Counter	T2 High Speed Counter	HD_CTUD_T2(FB)	Hollysys_PLC_Ex_CT.lib
	T3 High Speed Counter	HD_CTUD_T3(FB)	
	T4 normal counter	HD_CTUD_T4(FB)	
	T7 High Speed Counter	HD_CTU_T7(FB)	
Diphase Counter	T2 High Speed Counter	HD_DCTUD_T2(FB)	Hollysys_PLC_Ex_DCT.lib
	T3 High Speed Counter	HD_DCTUD_T3(FB)	
	T4 normal counter	HD_DCTUD_T4(FB)	
32-bit High-Speed Counter	32-bit High-Speed Counter	HD_DCTUD32_T3(FB)	Hollysys_PLC_Ex_DCT32.lib
Interruption Timer	Interruption Timer	HD_TIMER_T7(FB)	Hollysys_PLC_Ex_TIMER.lib
	Clear and Download Timer	HD_CLEAR_T7(FB)	
	Stop Timer	HD_STOP_T7(FB)	
External Interruption	Fast External Interruption (fit for 28K memory space)	Fast_ExINT(FB)	Hollysys_PLC_Ex_ExINT.lib
	Fast External Interruption (fit for 120K memory space)	Fast_ExINT_E(FB)	
PTO/PWM Pulse Transmission Out	PTO/PWM Pulse Transmission Out	PTO_PWM0(FB)	Hollysys_PLC_Ex_PT.lib
		PTO_PWM1(FB)	
Pulse Accelerated and Decelerated Transmission	PTO/PWM Pulse Transmission (Accelerated and Decelerated)	PTO_PWM0_Run(FB)	Hollysys_PLC__EX_PTRun.lib
		PTO_PWM1_Run(FB)	

A.2 IEC STANDARD INSTRUCTION LIST

LM PLC PowerPro is in accordance with the IEC61131-3 standard that specified the standard instructions for PLCs. The standard instructions include conversion instructions, arithmetic operators, bit-shift operators, comparison operators, type conversions, timers, counters, etc. The standard instructions of IEC61131-3 are as follows.

IEC standard instructions		
Instruction Type	Instruction Specification	Instruction Name
Arithmetic Operators	Addition	ADD
	Multiplication	MUL
	Subtraction	SUB
	Division	DIV
	Modulo Division	MOD
Logical Operators	Bitwise AND	AND
	Bitwise OR	OR
	Bitwise XOR	XOR
	Bitwise NOT	NOT
Bit-Shift Operators	Left-shift	SHL
	Right-shift	SHR
	Rotation to the left	ROL
	Rotation to the right	ROR
Selection Operators	Binary selection	SEL
	Returns the greater of 2 values	MAX
	Returns the lesser of 2 values	MIN
	Limits the value range	LIMIT
	Select a value out of a group of values	MUX
Comparison Operators	Greater than	GT
	Less than	LT
	Greater or equal	GE
	Lesser or equal	LE
	Equal	EQ
	Not equal	NE

Instruction Type	Instruction Specification	Instruction Name
Type Conversion Functions	BOOL_TO conversions	BOOL_TO_<TYPE>
	BYTE_TO conversions	BYTE_TO_<TYPE>
	DATE_TO conversions	DATE_TO_<TYPE>
	DINT_TO conversions	DINT_TO_<TYPE>
	DT_TO conversions	DT_TO_<TYPE>
	DWORD_TO conversions	DWORD_TO_<TYPE>
	INT_TO conversions	INT_TO_<TYPE>
	WORD_TO conversions	WORD_TO_<TYPE>
	REAL_TO conversions	REAL_TO_<TYPE>
	SINT_TO conversions	SINT_TO_<TYPE>
	STRING_TO conversions	STRING_TO_<TYPE>
	TIME_TO conversions	TIME_TO_<TYPE>
	TOD_TO conversions	TOD_TO_<TYPE>
	UDINT_TO conversions	UDINT_TO_<TYPE>
	UINT_TO conversions	UINT_TO_<TYPE>
	USINT_TO conversions	USINT_TO_<TYPE>
	Conversion from REAL to INT	TRUNC
Elementary Mathematical Functions	Returns the absolute value	ABS
	Returns the square root	SQRT
	Returns the natural logarithm of a number	LN
	Returns the logarithm of a number in base 10	LOG
	Returns the exponentiation function	EXP
	Returns the sine	SIN
	Returns the cosine	COS
	Returns the tangent	TAN
	Returns the arc sine	ASIN
	Returns the arc cosine	ACOS
	Returns the arc tangent	ATAN
	Exponentiation	EXPT
Address Operators	Address of the operand	ADR
	Content operator	^
	Returns the bit offset	BITADR
	Internal index for a POU	INDEXOF
	Number of bytes	SIZEOF
Calling Operators	Calling a function block or a program	CAL
Initialization Operator	Initialize retain variables	INI

Instruction Type	Instruction Specification	Instruction Name
Calling Operators Initialization Operator Bit-string Operators	Calling a function block or a program	CONCAT
	Initialize retain variables	DELETE
	Concatenation of two strings	FIND
	Delete string	INSERT
	Find string	LEFT
	Insert string	LEN
	Left initial string of given size of string	MID
	String length	REPLACE
	Returns a partial string from within a string	RIGHT
BCD Conversion	Convert a byte in BCD format into an INT value	BCD_TO_INT
	Convert an INT value into a byte in BCD format	INT_TO_BCD
Bi-stable Function Block	Bi-stable function, set dominant	SR
	Bi-stable function, reset dominant	RS
Trigger	Rising edge detection	R_TRIG
	Falling edge detection	F_TRIG
Counter	Counter up	CTU
	Counter down	CTD
	Counter up down	CTUD
Timer	Timer pulse	TP
	Timer on delay	TON
	Timer off delay	TOF
	Real time clock	RTC

A.3 FAST-CHECK LIST OF ASSOCIATED INSTRUCTION CONFLICT

Certain instructions cannot be used simultaneously for they use the same internal hardware or the same input/output ports. The following table shows all the associated conflict instruction B in the execution of the instruction A and its availability.

Used Instruction A	Associated Instruction B	Associated Hardware	Availability
HD_CTUD_T2	HD_DCTUD_T2	T2 Internal Counter, I0.0, I0.1 Pulse Input	Can not use instruction B
HD_CTUD_T3	HD_DCTUD_T3	T3 Internal Counter, I0.2, I0.3 Pulse Input	Can not use instruction B
	HD_DCTUD32_T3		
HD_CTUD_T4	HD_DCTUD_T4	T4 internal counter, I0.4, I0.5 pulse input	Can not use instruction B
	HD_DCTUD32_T3		
HD_CTU_T7	HD_DCTUD_T2	I0.6 Pulse Input and I0.6 External Reset Pulse Input	Can not use the External Reset Pulse Input B
	Fast_ExINT	I0.6 Pulse Input and I0.6 Fast External Interruption 3 Pulse Input	Can not use the Fast External Interruption 3 Pulse Input Channel B
	Fast_ExINT_E		
	HD_TIMER_T7	T7	Can not use instruction B
HD_DCTUD_T2	HD_CTUD_T2	T2 Internal Counter, I0.0, I0.1 Pulse Input	Can not use instruction B
	HD_CTU_T7	I0.6 External Reset pulse Input, I0.6 Pulse Input	Cannot use instruction B when using the external reset pulse input A.
	Fast_ExINT	I0.6 External Reset pulse Input, I0.6 Pulse Input of Fast External Interruption 3	Cannot use the Fast External Interruption 3 pulse input channel B when using the external reset pulse input A.
	Fast_ExINT_E		
HD_DCTUD_T3	HD_CTUD_T3	T3 Internal Counter, I0.2, I0.3 Pulse Input	Can not use instruction B
	HD_DCTUD32_T3		
	Fast_ExINT	I0.7 External Reset pulse Input, I0.7 Pulse Input of fast external Interruption 2	Cannot use the Fast External Interruption 2 pulse input channel B when using the external reset pulse input A.
	Fast_ExINT_E		
HD_DCTUD_T4	HD_CTUD_T4	T4 Internal Counter, I0.4, I0.5 Pulse Input	Can not use instruction B
	HD_DCTUD32_T3		
	Fast_ExINT	I1.0 External Reset pulse Input, I1.0 Pulse Input of Fast External Interruption 1	Cannot use the Fast External Interruption 1 pulse input channel B when using the external reset pulse input A.
	Fast_ExINT_E		

Used Instruction A	Associated Instruction B	Associated Hardware	Availability
Fast_ExINT	HD_CTU_T7	I0.6 pulse input of fast external interruption 3 I0.6 impulse input	Can not use instruction B when using the pulse input A of fast external interruption 3
	HD_DCTUD_T2	I0.6 impulse input of fast external interruption 3 I0.6 external reset pulse input	Can not use the external reset pulse input B when using the pulse input of fast external interruption 3
	HD_DCTUD_T3	I0.7 impulse input of fast external interruption 2 I0.7 external reset pulse input	Can not use the external reset pulse input B when using the pulse input of fast external interruption 2
	HD_DCTUD_T4	I1.0 impulse input of fast external interruption 1 I1.0 external reset pulse input	Can not use the external reset pulse input B when using the pulse input of fast external interruption 1
Fast_ExINT_E	HD_CTU_T7	I0.6 impulse input of fast external interruption 3 I0.6 impulse input	Can not use instruction B when using the pulse input A of fast external interruption 3
	HD_DCTUD_T2	I0.6 impulse input of fast external interruption 3 I0.6 external reset pulse input	Can not use the external reset pulse input B when using the pulse input of fast external interruption 3
	HD_DCTUD_T3	I0.7 impulse input of fast external interruption 2 I0.7 external reset pulse input	Can not use the external reset pulse input B when using the pulse input of fast external interruption 2
	HD_DCTUD_T4	I1.0 impulse input of fast external interruption 1 I1.0 external reset pulse input	Can not use the external reset pulse input B when using the pulse input of fast external interruption 1
PTO_PWM0	PTO_PWM0_Run	T12	Can not use instruction B
	PTOCtrl_0		
PTO_PWM1	PTO_PWM1_Run	T8, T13	Can not use instruction B
	PTOCtrl_1	T13	
PTO_PWM0_Run	PTO_PWM0	T12	Can not use instruction B
	PTOCtrl_0		
PTO_PWM1_Run	PTO_PWM1	T8, T13	Can not use instruction B
	PTOCtrl_1	T13	
PTOCtrl_0	PTO_PWM0	T12	Can not use instruction B
	PTO_PWM0_Run		
PTOCtrl_1	PTO_PWM1	T13	Can not use instruction B
	PTO_PWM1_Run		

Used Instruction A	Associated Instruction B	Associated Hardware	Availability
HD_DCTUD32_T3	HD_CTUD_T3	T3 Internal Counter, I0.2, I0.3 Pulse Input	Can not use instruction B
	HD_CTUD_T4	T4 Internal Counter	
	HD_DCTUD_T3	T3 Internal Counter, I0.2, I0.3 Pulse Input	
	HD_DCTUD_T4	T4 Internal Counter	
HD_TIMER_T7	HD_CTU_T7	T7	Can not use instruction B

A.4 HARDWARE MODULE STATUS

A.4.1 Special Register Area

The 10 bytes of %MB0~%MB9 are special register, %MB0, %MB1 are error flags, %MB2, %MB3 are error data, %MB2 are low 8 bits of data and %MB3 are high 8 bits of data.

%MB0:

High bit Low bit

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

- Bit0=1: Expansion address 0 module communication CRC error, error data not defined.
  Bit1=1: Expansion address 1 module communication CRC error, error data not defined.
- Bit2=1: Expansion address 2 module communication CRC error, error data not defined.
- Bit3=1: Expansion address 3 module communication CRC error, error data not defined.
- Bit4=1: Expansion address 4 module communication CRC error, error data not defined.
- Bit5=1: Expansion address 5 module communication CRC error, error data not defined.
- Bit6=1: Expansion address 6 module communication CRC error, error data not defined.

%MB1:

High bit Low bit

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

- Bit0=1: SSC hardware operation error, system fault, error data is current value of SSCCON.
- Bit1=1: ASC hardware operation error, error data is current value of S0CON.
- Bit2=1: ADC hardware operation error, error data is current value of ADEIC.
- Bit3=1: B kind hardware error, error data is current value of TFR.
- Bit4=1: Low overflow error of system stack, error data is current value of TFR.
- Bit5=1: High overflow error of system stack, error data is current value of TFR.
- Bit6=1: System NMI error, error data is current value of TFR.

%MB2, %MB3: error data.

High bit Low bit

%MB3	%MB2

A.4.2 Module Diagnostic Area

When using PowerPro for hardware module configuration, assign a diagnostic area of 10 bytes in M area for each module (including CPU module). The diagnostic area of CPU module starts from %MB20, The diagnostic area of expansion module starts from %MB30. The start address can be gained from the base parameters in PLC configuration, as shown in figure A-4-1.

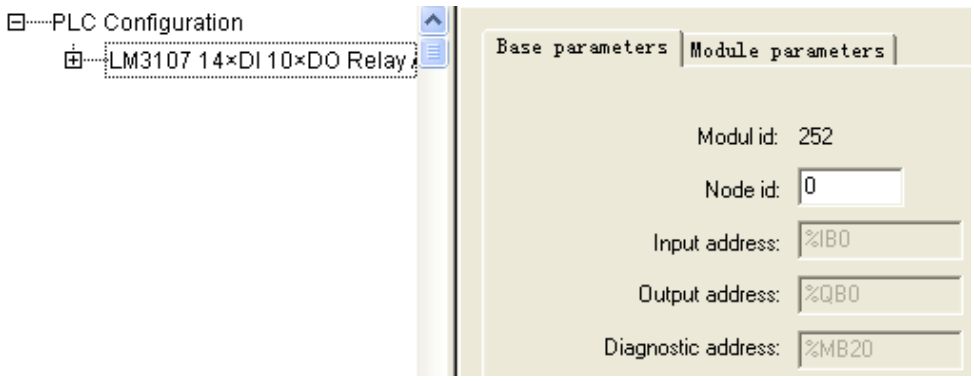


Figure A-4-1

The first byte of module diagnostic bytes stands for the module status, and different modules have different module status; for analog input module, the bytes from the second stand for channel status, each channel occupies 4bit, the low channel occupies low 4bit and the high channel occupies high 4bit.

The detailed module status is as follows:

Diagnostic Area of CPU Module

- CPU module: Only 1 byte for module status.

Bit7							Bit0
×	×	×	×	×	×	×	ID

- Bit0=1, module ID configuration right; Bit0=0, module ID configuration error.
- Bit1~Bit7: hold
- Bytes 2~10 hold

Diagnostic Area of Digital Module

- The first byte:

Bit7							Bit0
×	RD	×	×	×	×	×	ID

- Bit0=1, module ID configuration right; Bit0=0, module ID configuration error.
- Bit6=1, CPU module gets expansion module ID successfully; Bit6=0, CPU module fails to get expansion module ID
- Bit1~Bit5, Bit7: hold
- The second byte is the expansion module ID when bit6 of the first byte is 1, otherwise it is 0.
- Bytes 3~10 hold.

Diagnostic Area of Analog Input Module

The first byte: module status

Bit7	6	5	4	3	2	1	Bit0
CRC	×	×	AD_OK	READY	CHANNEL	PWR	ID

- Bit0=1, module ID configuration right; Bit0=0, module ID configuration error.
- Bit1=1, DC 24V outside power supply normal, Bit1=0, no DC 24V outside power supply.
- Bit2=1, no channel error, Bit2=0, channel error.
- Bit3=1, module ready, Bit3=0, module not ready.
- Bit4=1, AD chip normal, Bit4=0, AD chip abnormal.
- Bit5-Bit6: Hold.
- Bit7=1, expansion module gets wrong CRC, Bit7=0, expansion module gets right CRC.

Bytes from the second indicate the channel status:

- Each 4bit stands for one channel status, and different modules have different channel status.

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

The second channel (high channel) the first channel (low channel)

Channel status of LM3310: 4 channels occupy the second and the third bytes, the others hold.

- 0x1: out of wire (only effective for 4-20mA range).
- 0x2: channel not ready
- 0xF: no channel error
- others: hold

Channel status of LM3311: 4 channels occupy the second and the third bytes, the others hold.

- 0x1: wire-off
- 0x2: channel not ready
- 0xF: no channel error
- others: hold

Channel status of LM3312: 4 channels occupy the second and the third bytes, the others hold.

- 0x0: over range, too small resistance (<39Ω).
- 0x2: channel not ready
- 0xF: no channel error
- others: hold

Channel status of LM3313: 8 channels occupy bytes 2-5 and the others hold.

- 0x2: channel not ready
- 0xF: no channel error
- others: hold

Channel status of LM3314: 8 channels occupy bytes 2-5 and the others hold.

- 0x0: over range, too small resistance (<600Ω).
- 0x1: over range, too large resistance (>100KΩ).
- 0x2: channel not ready
- 0xF: no channel error
- others: hold

Diagnostic Area of Analog Output Module

The first byte: Module status

Bit7	6	5	4	3	2	1	Bit0
CRC	×	×	×	×	×	PWR	ID

- Bit0=1, module ID configuration right; Bit0=0, module ID configuration error.
- Bit1=1, DC 24V outside power supply normal, Bit1=0, no DC 24V outside power supply.
- Bit2~6: hold
- Bit7=1, expansion module gets wrong CRC, Bit7=0, expansion module gets right CRC.
- No channel status for analog output channel, bytes 2~10 hold.

Diagnostic Area of DP Module

The first byte: Module status

Bit7	6	5	4	3	2	1	Bit0
CRC	×	×	SPC3_STATE	OUTDATA_LEN	INDATA_LEN	×	ID

- Bit0=1, module ID configuration right; Bit0=0, module ID configuration error.
- Bit1: Hold.
- Bit2=1, the number of input data bytes for DP module configuration in PowerPro matches the number of output data bytes for DP module configuration in DP-Master；Bit2=0, the number of input data bytes for DP module configuration in PowerPro does not match the number of output data bytes for DP module configuration in DP-Master.
- Bit3=1, the number of output data bytes for DP module configuration in PowerPro matches the number of input data bytes for DP module configuration in DP-Master； Bit3=0, the number of output data bytes for DP module configuration in PowerPro does not match the number of input data bytes for DP module configuration
- Bit4=1, SPC3 communication normal； Bit4=0, SPC3 communication error.
- Bit5-Bit6: hold
- Bit7=1, expansion module gets wrong CRC, Bit7=0, expansion module gets right CRC.
- No channel status for DP module, and bytes2~10 hold.

Diagnostic Area of Ethernet Module

The first byte: Module status

Bit7	6	5	4	3	2	1	Bit0
CRC	×	×	×	×	×	SWP	ID

- Bit0=1, module ID configuration right; Bit0=0, module ID configuration error.
- Bit1=1, have network data exchange, Bit1=0 no network data exchange.
- Bit2-Bit6: hold.
- Bit7=1, expansion module gets wrong CRC, Bit7=0, expansion module gets right CRC.
- No channel status for Ethernet module, and bytes2~10 hold.

Appendix

B

Appendix B

B.1 APPLICATION OF STARTUP AND PULSE-SENDING OF MOTOR

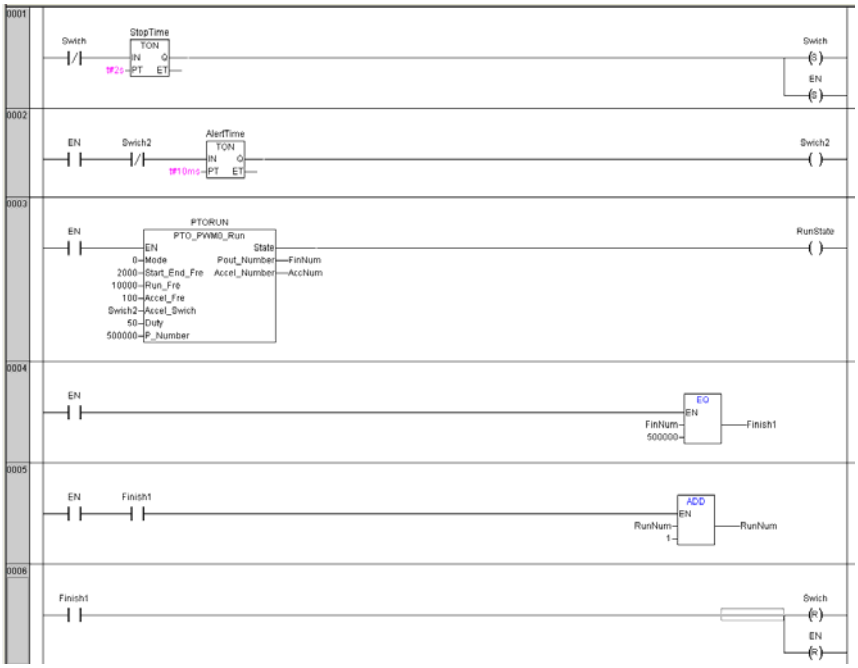
B.1.1 Requirements

Stepper motor starts up at the speed of 2KHz, runs at the speed of 10KHz, sends out 500000 pulses and then stops for 2 seconds, and starts up again at the speed of 2KHz, runs at the speed of 10KHz, sends out 500000 pulses and repeats the operation.

B.1.2 Variable Declaration

```
PROGRAM PLC_PRG
VAR
    Swich: BOOL;
    Swich2: BOOL;
    EN: BOOL;
    Finish1: BOOL;
    RunNum: WORD;
    RunState: BOOL;
    FinNum: DWORD;
    AccNum: DWORD;
    AlertTime: TON;
    PTORUN: PTO_PWM0_Run;
    StopTime: TON;
END_VAR
```

B.1.3 LD Program



B.1.4 Specification

Two seconds after the program starts to run, the motor starts up at the speed of 2KHz, and then increases by 100Hz each 10ms, runs with constant speed when reaches 100Hz. When the number of the rest pulses equals the number of pulses sent out at speed-up stage, the motor starts to decelerate , and the acceleration curve matches the deceleration curve. When 500000 pulses are sent out, the motor stops for 2 seconds and then repeats the operation.

B.2 APPLICATION EXAMPLE OF PROFIBUS-DP MODULE

B.2.1 Requirements

PLC sends out data of 8 bytes to DP master by DP slave and receives data of 8 bytes from DP master at the same time.

B.2.2 Variable Declaration

```
PROGRAM PLC_PRG
VAR
    EN: BOOL;
    Example: DP_Slave;
    T_OR_F: BOOL;
    SendDataA: WORD; PLC sends data A to DP master
    SendDataB: WORD; PLC sends data B to DP master
    SendDataC: WORD; PLC sends data C to DP master
    SendDataD: WORD; PLC sends data D to DP master
    RecDataA: WORD;   DP master sends data A to PLC
    RecDataB: WORD;   DP master sends data B to PLC
    RecDataC: WORD;   DP master sends data C to PLC
    RecDataD: WORD;   DP master sends data D to PLC
END_VAR
```

B.2.3 Software Configuration

The configuration of 3401DP slave is shown at the cursor position in figure B-2-1.

- InputDataLen_Byte is the length of data sent to PLC by DP master, and input the bytes 8.
- OutputDataLen_Byte is the length of data sent to DP master by PLC, and input the bytes 8.

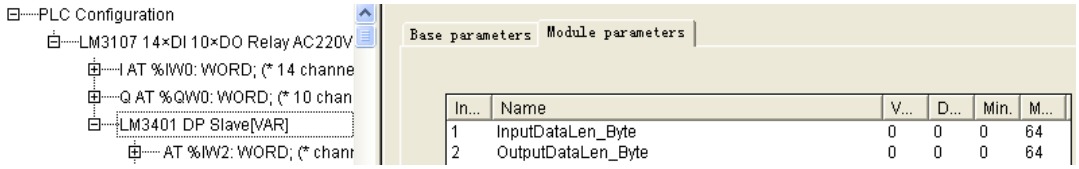


Figure B-2-1

The Address of DP_Slave should be consistent with the node id in figure B-2-2, the data A, B, C, D sent to PLC by DP master are stored in %IW2, %IW4, %IW6, %IW8, as shown in figure B-2-3.

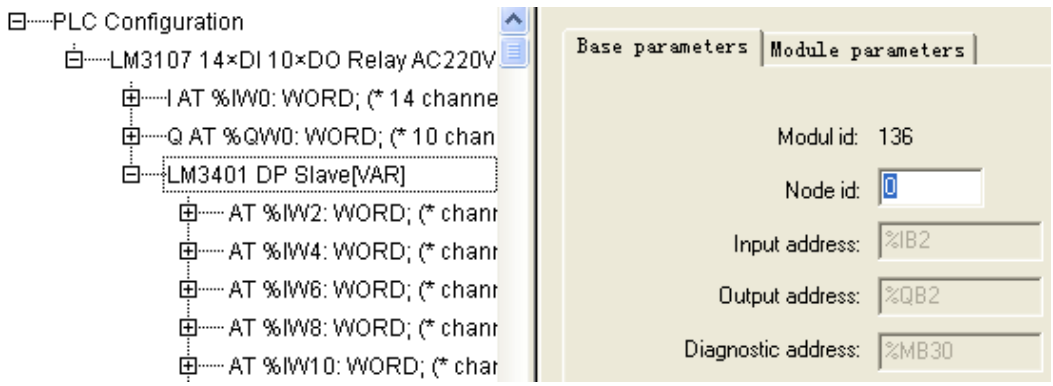


Figure B-2-2

The data A, B, C, D sent to DP master by PLC is stored in %QW2, %QW4, %QW6, %QW8, as shown in figure B-2-3.

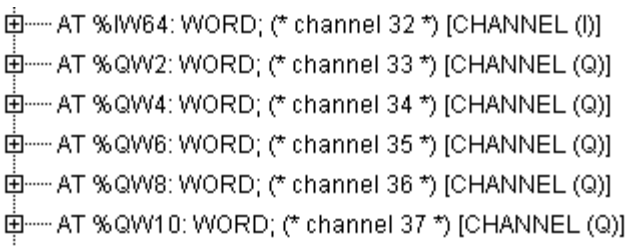


Figure B-2-3

Configure the receive area and send area of DP master so that the data in QW area of DP slave will send to the receive area of DP master and the data of DP master will send to IW area of DP slave automatically.

B.2.4 LD Program

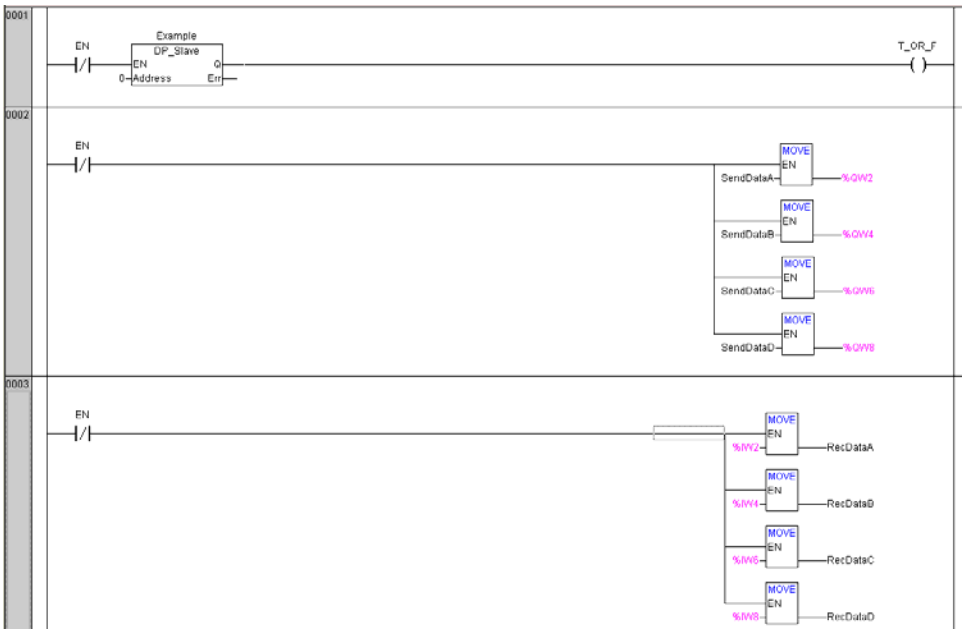


Figure B-2-4

B.2.5 Specification

After power-on, program will copy the data of SendDataA, SendDataB, SendDataC, SendDataD to %QW2, %QW4, %QW6, %QW8 continuously, and the data in %QW2, %QW4, %QW6, %QW8 will be sent to the receive area of DP master automatically.

The data sent by DP master will be stored automatically in %IW2, %IW4, %IW6, %IW8, and the program will copy the data in %IW2, %IW4, %IW6, %IW8 continuously to RecDataA, RecDataB, RecDataC, RecDataD.

B.3 APPLICATION EXAMPLE OF ETHERNET INSTRUCTION

HMI (MODBUS/TCP master) sends data of 2 bytes to the input area of PLC by Ethernet module, and receives data of 2 bytes from the output area of PLC. At the same time PC sends 1bit to the input area of PLC and receives 1bit from output area of PLC. HMI configuration is not introduced here.

B.3.1 PowerPro Program

Variable Declaration

```
PROGRAM PLC_PRG
VAR
      EN: BOOL;
      Example: EtherNet_TCP;
T_OR_F: BOOL;
SendDataA: WORD; (*data A sent to MODBUS/TCP master by PLC *)
SendDataB: WORD; (*data B sent to MODBUS/TCP master by PLC *)
SendBitC: BOOL;   (*data C sent to MODBUS/TCP master by PLC *)
RecDataA: WORD;   (*data A sent to PLC by MODBUS/TCP master*)
RecDataB: WORD;   (*data B sent to PLC by MODBUS/TCP master *)
RecBitC: BOOL;    (*data C sent to PLC by MODBUS/TCP master *)
END_VAR
```

Software Configuration

The configuration of LM3403 Ethernet module is shown in figure B-3-1.

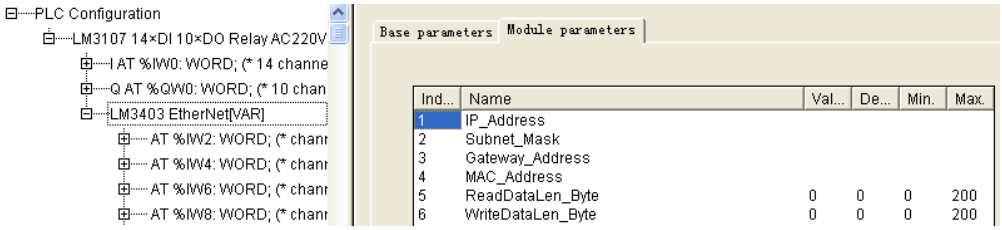


Figure B-3-1

- IP_Address is the IP of Ethernet module (must be in the same network segment with PC and no conflict with other IPs).
- Subnet_Mask is the subnet mask that is consistent with the subnet mask of PC.
- GateWay_Addres is the address of gateway.
- MAC_Address null.
- ReadDataLen_Byte is the length of data sent to PLC by PC, and the received bytes is 8 (which must be larger than the actual length and the maximum is 200).
- WriteDataLen_Byte is the length of data sent to PC by PLC, and the sent bytes is 8 (which must be larger than the actual length and the maximum is 200).
- The Address in Ethernet module should be consistent with the node id in figure B-3-2, the data A, B sent to PLC by PC are stored in %IW4, %IW6 in figure B-3-2, and data C is stored in %IX8.0.

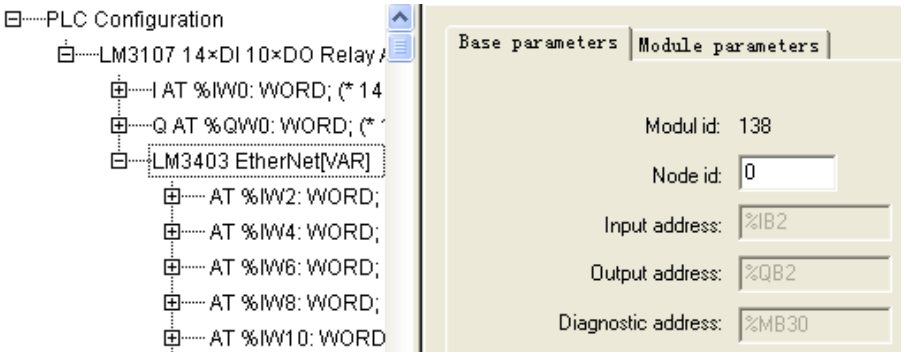


Figure B-3-2

The data A, B sent to PC by PLC are stored in %QW2, %QW4, as shown in figure B-3-3, and data C is stored in %QX6.0.

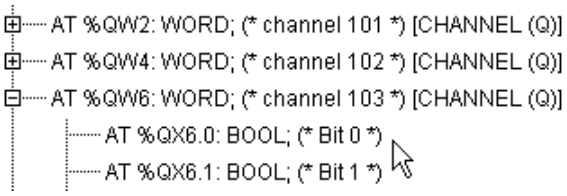


Figure B-3-3

LD Program

Figure B-3-4

B.4 APPLICATION EXAMPLE OF INTERRUPT ASSOCIATED EVENTS

B.4.1 PRI of Interrupt Events

Interrupt Events	PRI
Fast External 0 interrupt	
Fast External 1 interrupt	low
Fast External 2 interrupt	
Fast External 3 interrupt	
HD_TC7 interrupt	
HD_TC2 interrupt	
HD_TC3 interrupt	
HD_TC4 interrupt	high
HD_RTC_ALM 0 interrupt	

B.4.2 Requirements

Responding to the pulse, the PLC generates an interrupt whenever I0.6 reaches a rising edge without the effects of PLC scanning periods, and the value in %MW100 increases by 10. When I0.7 reaches a rising edge, the PLC will respond to the pulse and generate a interrupt, the value in %MW102 increases by 15.

B.4.3 Program Analysis

According to the requirements, LM3106 CPU module can be used as the hardware, and the following instructions shall be called in the program:

Fast_ExINT (fast external interrupt)

The program includes three parts:

- Main Program——define the fast external interrupt mode of 3106
- INT3PRO——I0.6 pulse to interrupt program——the value in %MW100 increases by 10
- INT2PRO——I0.7 pulse to interrupt program——the value in %MW102 increases by 15

B.4.4 Program

Main program：

(1) First add Hollysys_PLC_Ex_ExINT.lib to library manager, see section 1.5 for details.

(2) According to the requirements, I0.6, I0.7 generate an interrupt when reach a rising edge and I1.0 is not used, so for Fast_ExINT, the Mode=16#50, the variable declaration in main program and ladder diagram are shown in figure B-4-1.

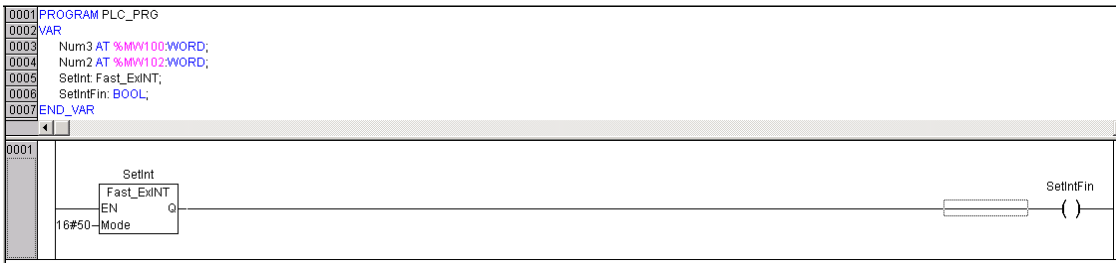


Figure B-4-1

(3) Because fast external interrupt 3 (I0.6) and fast external interrupt 2 (I0.7) are used, “√” shall be filled in the cursor position in figure B-4-2.

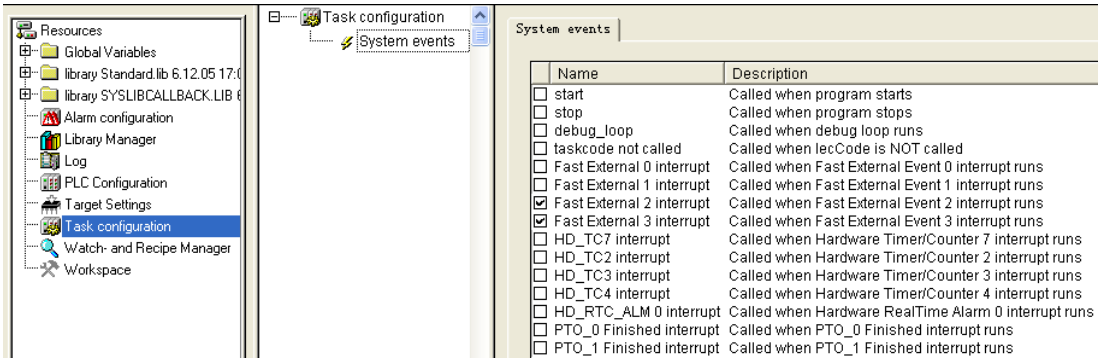


Figure B-4-2

(4)Click Create POU to build sub-programs INT2Pro, INT3Pro after Fast External interrupt 2 and Fast External interrupt 3, as shown in figure B-4-3.

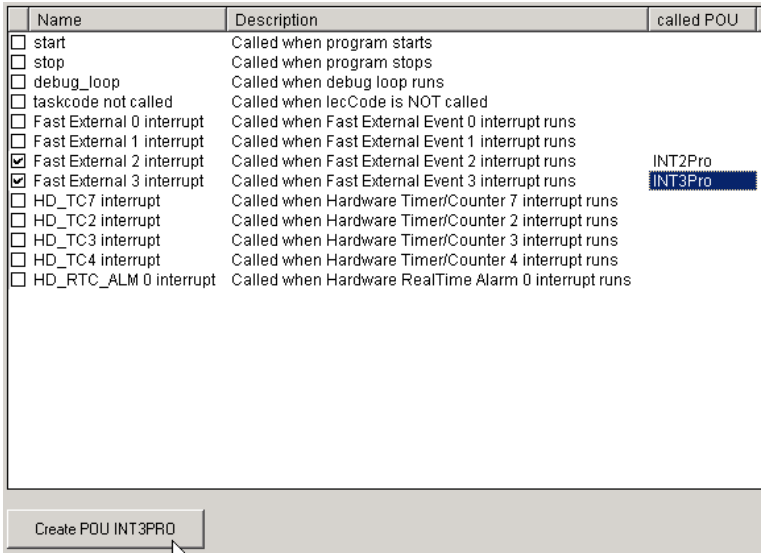


Figure B-4-3

(5)The default language is ST for the new interrupt program, and it can be converted to LD after compiling, as shown in figure B-4-4.

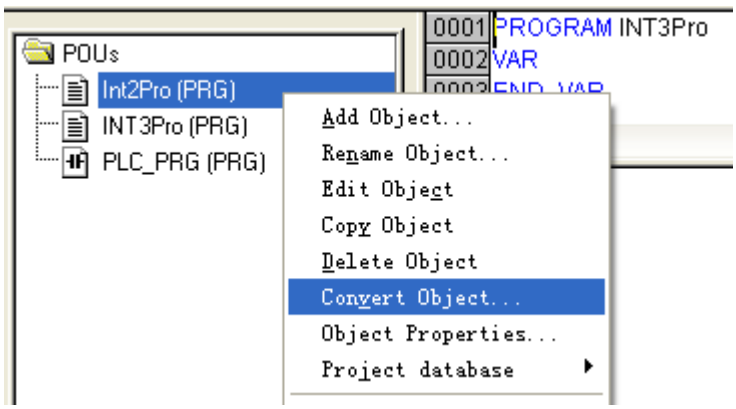


Figure B-4-4

(6)LD of INT3Pro interrupt program is shown in figure B-4-5.

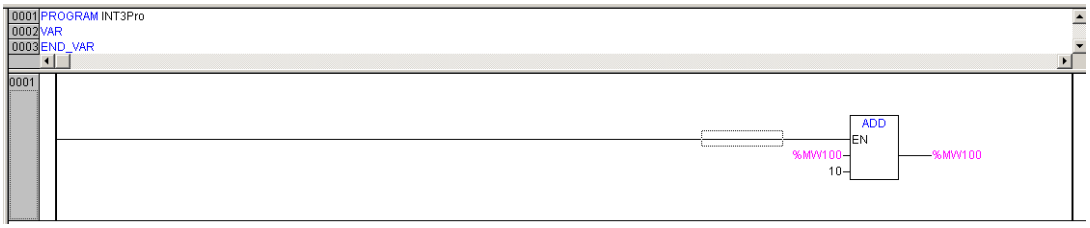


Figure B-4-5

(7)LD of INT2Pro interrupt program is shown in figure B-4-6.

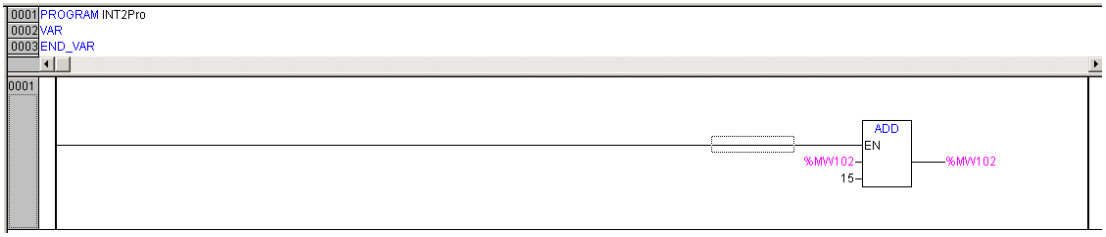


Figure B-4-6

B.5 APPLICATION EXAMPLE OF FREE PORT COMMUNICATION

B.5.1 Requirements

When a measure instrument and LM3108-CDT module communicate through 485 free port, the protocol format is as follows:

Free Protocol Format (18 bytes in all)																		
STX	A	B	C	X	X	X	X	X	X	Y	Y	Y	Y	Y	Y	CR	CKS
Start Symbol (02H)	Status Word ABC			Weight						Tare						Enter 0DH	Check

Status Word A			
Bit2	Bit1	Bit0	Position of Decimal Point
0	0	0	XXXX00
0	0	1	XXXXX0
0	1	0	XXXXXX
0	1	1	XXXXX.X
1	0	0	XXXX.XX
1	0	1	XXX.XXX
1	1	0	XX.XXXX
1	1	1	X.XXXXX

What LM3108-CDT should analyze is the bit 0, bit 1, bit 2 of the status word A, the position of decimal point and the weight value. LM3108 analyzes the value between 0～10 which is sent by instrument and then outputs analog signal between 0～10V.

Communication parameters: 9600Bps, 8bits, no parity check.

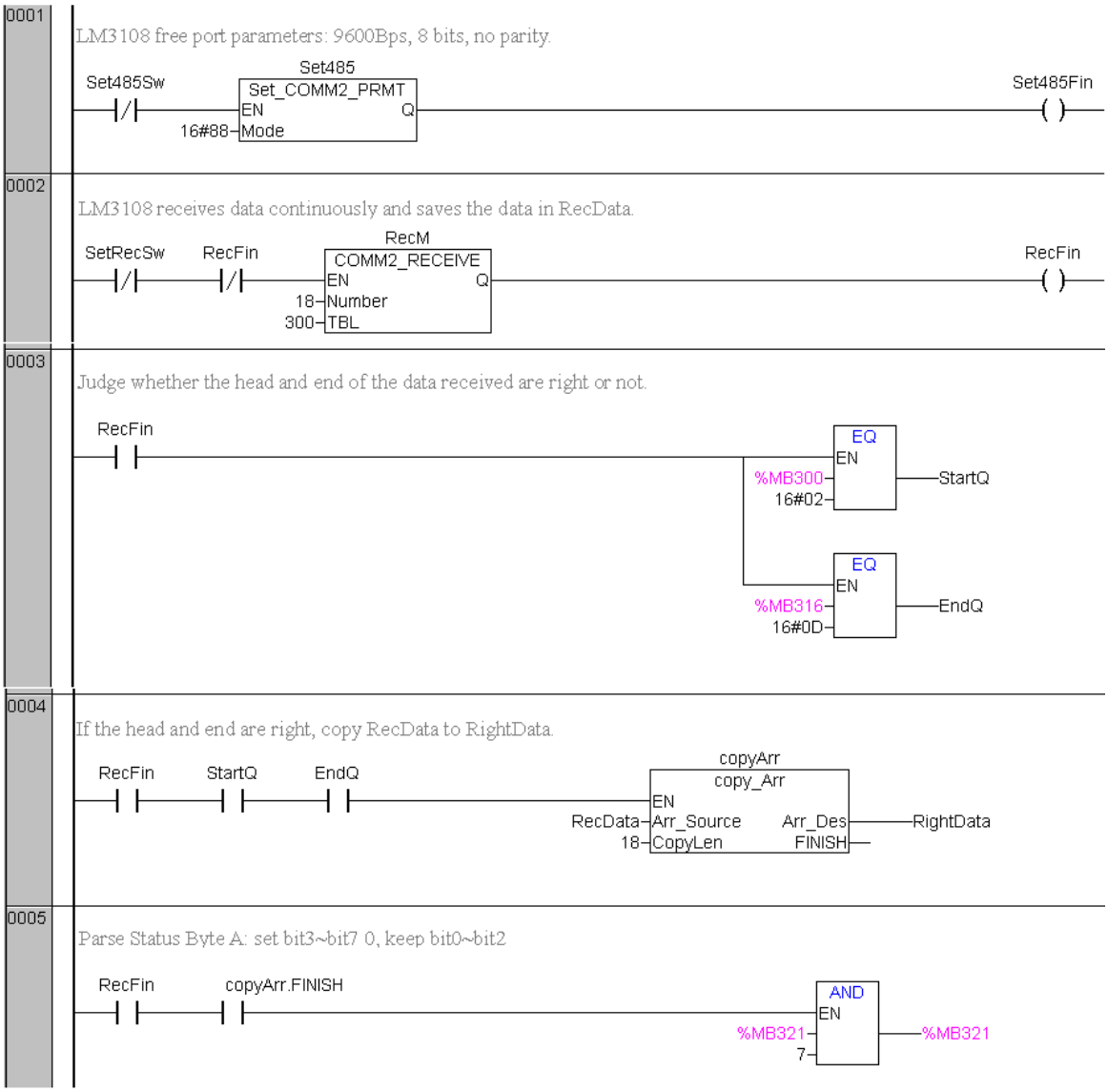
B.5.2 Program

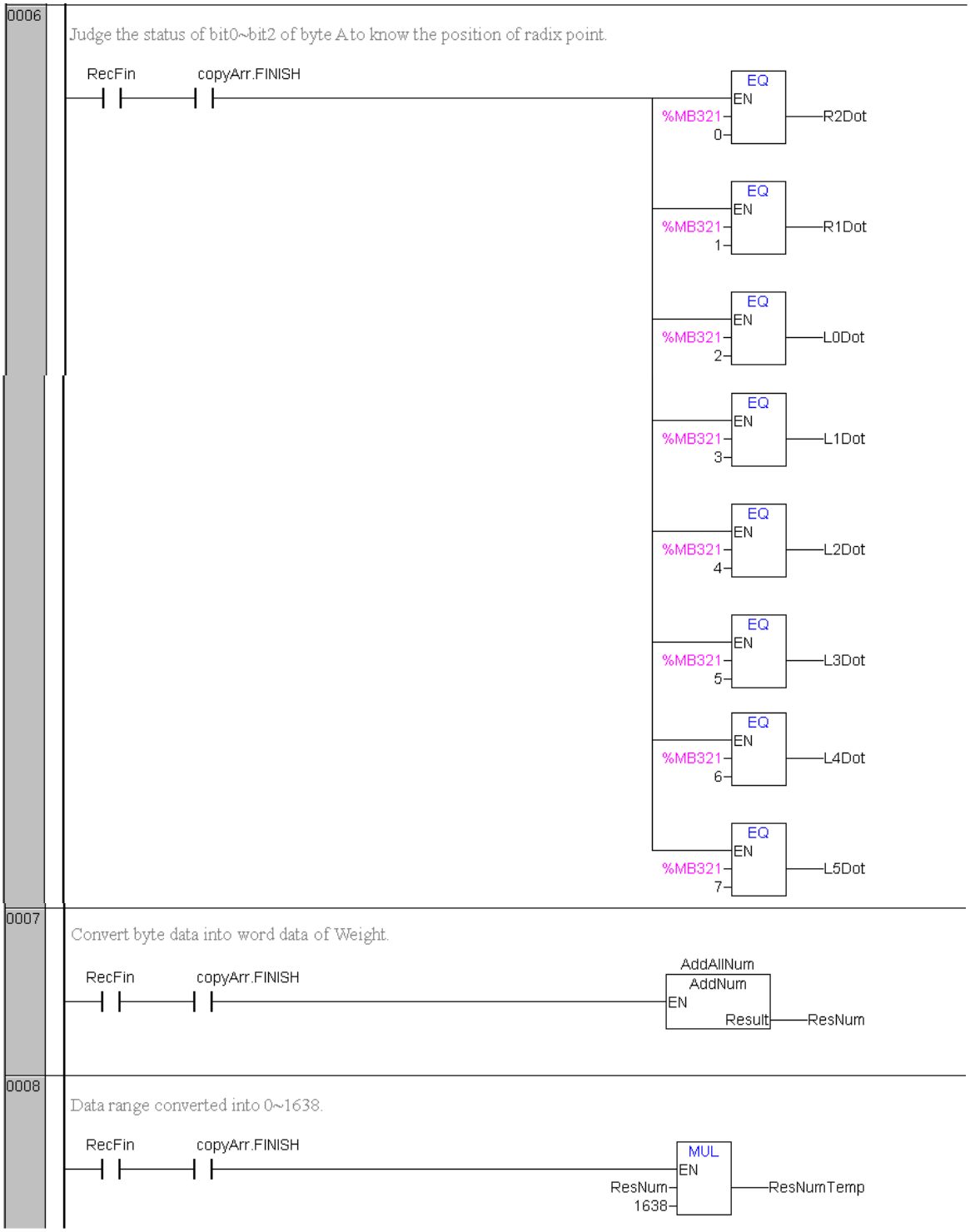
Variable declaration

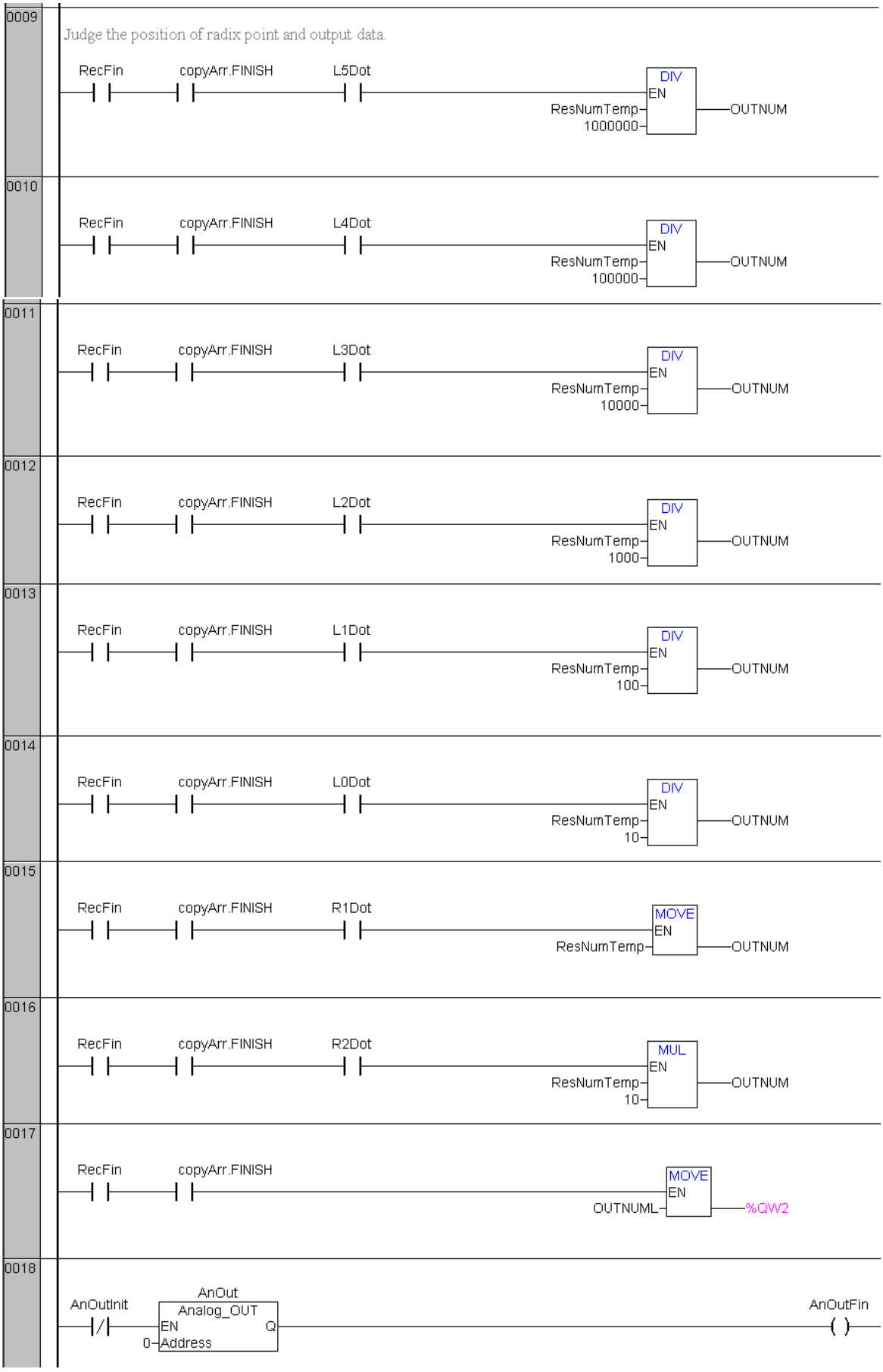
```
PROGRAM PLC_PRG
VAR
    RecFin: BOOL;
    Set485: Set_COMM2_PRMT;
    RecData AT %MB300 : ARRAY [1..18] OF BYTE;
    RightData AT %MB320 : ARRAY [1..18] OF BYTE;
    Set485Sw: BOOL;
    Set485Fin: BOOL;
    SetRecSw: BOOL;
    RecM: COMM2_RECEIVE;
    StartQ: BOOL;
    EndQ: BOOL;
    copyArr: copy_Arr;
    L5Dot: BOOL;
    L4Dot: BOOL;
    L3Dot: BOOL;
    L2Dot: BOOL;
    L1Dot: BOOL;
    L0Dot: BOOL;
    R1Dot: BOOL;
    R2Dot: BOOL;
    ResNum: DWORD;
```

```
        AddAllNum: AddNum;
        ResNumTemp: DWORD;
        DIVNUM: DWORD;
        OUTNUMH AT %MW202 : WORD;
        OUTNUML AT %MW200 : WORD;
        OUTNUM AT %MD200 : DWORD;
        AnOutInit: BOOL;
        AnOut: Analog_OUT;
        AnOutFin: BOOL;
END_VAR
```

B.5.3 LD of main program




0006
Judge the status of bit0~bit2 of byte A to know the position of radix point.
RecFin
copyArr.FINISH
EQ
EN
%MB321
0
R2Dot
%MB321
1
R1Dot
%MB321
2
L0Dot
%MB321
3
L1Dot
%MB321
4
L2Dot
%MB321
5
L3Dot
%MB321
6
L4Dot
%MB321
7
L5Dot
0007
Convert byte data into word data of Weight.
RecFin
copyArr.FINISH
AddAllNum
AddNum
EN
Result
ResNum
0008
Data range converted into 0~1638.
RecFin
copyArr.FINISH
MUL
EN
ResNum
1638
ResNumTemp


0009
Judge the position of radix point and output data.
RecFin
copyArr.FINISH
L5Dot
DIV
EN
ResNumTemp
1000000
OUTNUM
0010
RecFin
copyArr.FINISH
L4Dot
DIV
EN
ResNumTemp
100000
OUTNUM
0011
RecFin
copyArr.FINISH
L3Dot
DIV
EN
ResNumTemp
10000
OUTNUM
0012
RecFin
copyArr.FINISH
L2Dot
DIV
EN
ResNumTemp
1000
OUTNUM
0013
RecFin
copyArr.FINISH
L1Dot
DIV
EN
ResNumTemp
100
OUTNUM
0014
RecFin
copyArr.FINISH
L0Dot
DIV
EN
ResNumTemp
10
OUTNUM
0015
RecFin
copyArr.FINISH
R1Dot
MOVE
EN
ResNumTemp
OUTNUM
0016
RecFin
copyArr.FINISH
R2Dot
MUL
EN
ResNumTemp
10
OUTNUM
0017
RecFin
copyArr.FINISH
MOVE
EN
OUTNUML
%QW2
0018
AnOutInit
AnOut
Analog_OUT
EN
Q
0
Address
AnOutFin

B.5.4 **AddNum** - Program of self-defined Function Block

```
0001 FUNCTION_BLOCK AddNum
0002 VAR_INPUT
0003 END_VAR
0004 VAR_OUTPUT
0005     Result: DWORD;
0006 END_VAR
0007 VAR
0008 END_VAR
```

```
0001
0002 Result := %MB324*100000+%MB325*10000+%MB326*1000+%MB327*100+%MB328*10+%MB329;
```

B.5.5 **copy_Arr** - Program of self-defined Function Block

```
0001 FUNCTION_BLOCK copy_Arr
0002 VAR
0003     i: WORD;
0004 END_VAR
0005 VAR_INPUT
0006     Arr_Source: ARRAY [1..18] OF BYTE;
0007     CopyLen: WORD;
0008 END_VAR
0009 VAR_OUTPUT
0010     Arr_Des: ARRAY [1..18] OF BYTE;
0011     FINISH: BOOL := FALSE;
0012 END_VAR
```

```
0001 FOR i:=1 TO CopyLen BY 1 DO
0002     Arr_Des[i]:=Arr_Source[i];
0003 END_FOR;
0004 FINISH:=TRUE;
```

B.6 APPLICATION EXAMPLE OF PID CONTROLLER

B.6.1 Requirements

By generating signal operations to the signal generator, PLC simulates the actual value of PID controller and after PID regulation and E_H conversion outputs the value to the channel of output module.

B.6.2 Variable Declaration

```
PROGRAM PLC_PRG
VAR
    GEN1: GEN;              signal generator to simulate actual signal
    VAR1: INT;              the output value of signal generator
    KP1: REAL := 0.5;       proportion coefficient, the initial value is 0.5
    TV1: DWORD := 4;        derivative time, the initial value is 4
    Tn1: DWORD := 100;      integral time, the initial value is 100
    pid1: PID;              PID controller
    en: BOOL;
    cyc_num: INT := 10000;      number of signal, the initial value is 10000
    perio_num: TIME := T#4000ms;  signal period, 4 second
    VAR_ACTUAL: REAL;       actual value of PID
    E_H1: E_H;              E_H conversion, output to analog output port
    VAR_Y: REAL;            output value of PID controller
    ANA_IN: Analog_IN;      analog input
    ANA_OUT: Analog_OUT;    analog output
END_VAR
```

B.6.3 PLC Configuration

As shown in figure B-6-1, a CPU module LM3107 is configured with an analog input module LM3310 and an analog output module LM3320. The PID controller needs an actual value for the controlled variable and the output value of the PID controller has to be converted to the action value of plant. These two values correspond to the analog input and the analog output modules respectively. In this example, for purpose of testing sine wave is generated by signal generator to simulate the actual value of the controlled variable.

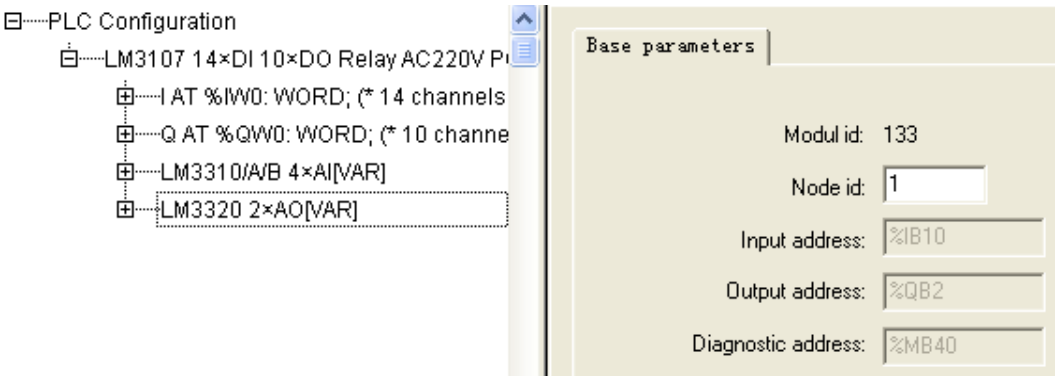


Figure B-6-1

B.6.4 Visualization Program

Parameter optimization of PID controllers is the key of PID control. Since each system has its inherently different attributes, it is needed to determine the parameters by experiments and it is suggested to use visualization function to determine the derivative time, integral time, and proportional constant. In this example, we build a visual interface using the visualization function of the software (see software manual for help) as shown in figure B-6-

2. In this figure, on the top there is a curve of the actual value of a PID controller, at the bottom there is an output curve of the PID controller, on the right there is a parameter interface of derivative time, integral time, proportional constant and you can optimize them by changing the parameter values.

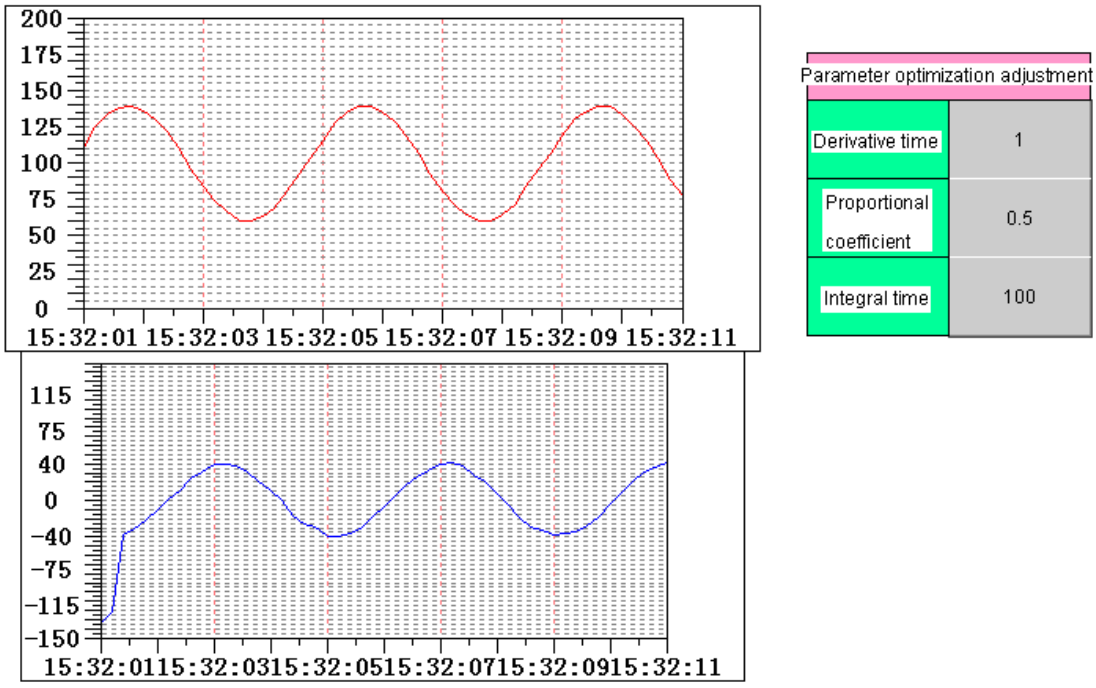
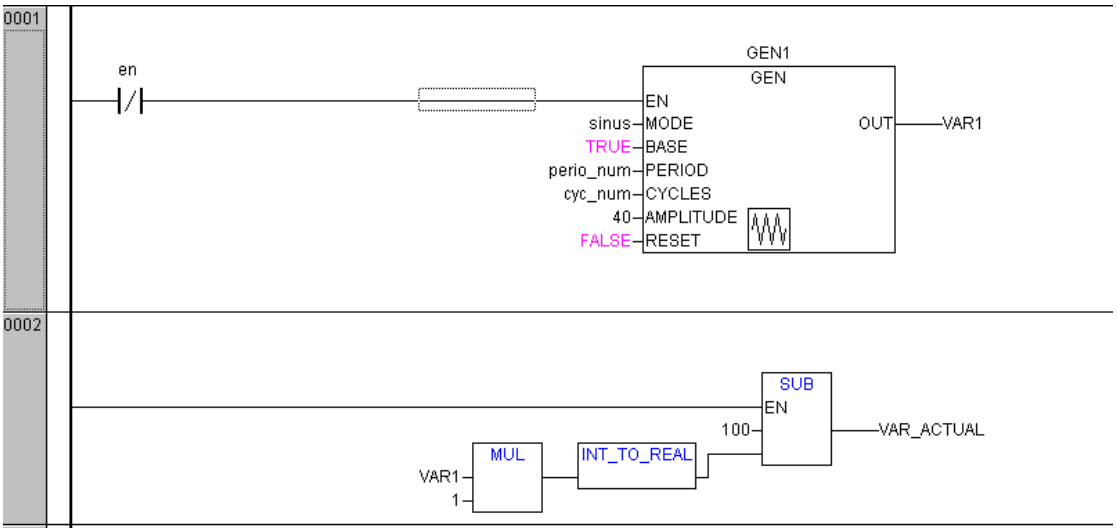


Figure B-6-2

B.6.5 LD Program



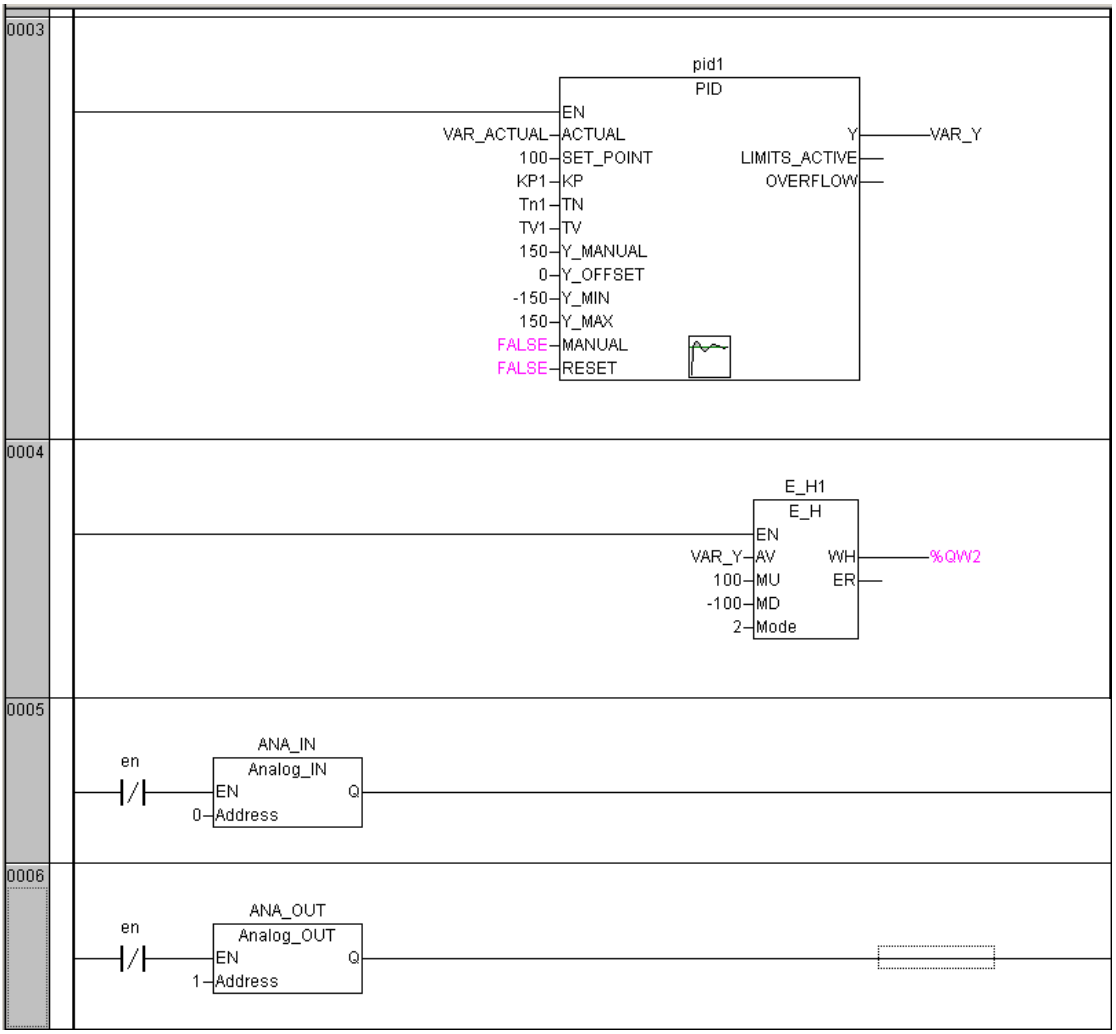


Figure B-6-2

B.6.6 Specification

A sine signal generated by GEN1 is assigned to VAR1 as an analog input; the processed VAR1 is converted as the measured value of a PID controller, and assign output value of PID controller through the E_H conversion to the analog output channel.

B.7 APPLICATION EXAMPLE OF HIGH-SPEED COUNTER

B.7.1 Requirements

A function of LM PLC High-Speed Counter (HD_CTUD_T2) is explained by this example. The High-Speed Counter processes the high-speed pulse signal from PLC and the PTO_PWM0 instruction provides the high-speed pulse output signal.

B.7.2 Wiring

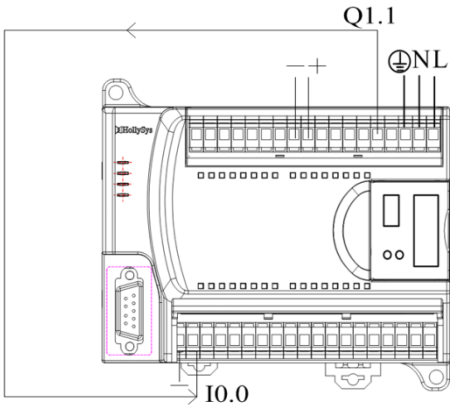


Figure B-7-1 Wiring (LM3106, output pulse to terminal I0.0 from Q1.1)

B.7.3 Variable Declaration

```
PROGRAM PLC_PRG
VAR
    EN: BOOL;
    PTO_INS: PTO_PWM0;          pulse send
    PT_OVER: BOOL;
    CTUD_INS: HD_CTUD_T2;       high-speed counter
    P_NUM: DWORD;               number of pulse sent out
    CV_NUM: UINT;               number of counted
    CTUD_OVER: BOOL;
END_VAR
```

B.7.4 PLC Configuration

LM3106-CDT is configured as the CPU module in this example.

B.7.5 LD Program

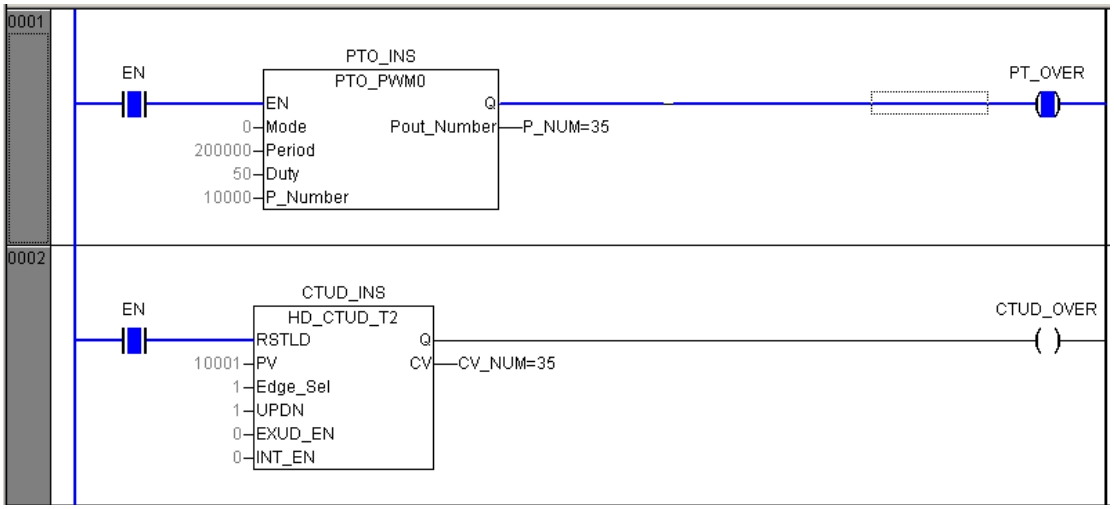


Figure B-7-2

B.7.6 Description

When EN is enabled, PTO_PWM0 outputs pulse through Q1.1 in PTO mode, the number is 10000, period is 200ms and P_NUM is the number of pulse sent out. HD_CTUD_T2 receives pulse through I0.0, and CV_NUM is the current value and increases by 1 at each rising edge.

B.8 APPLICATION EXAMPLE OF ANALOG POTENTIOMETER / PRESET

B.8.1 Requirements

The CPU module of HollySys LM Micro PLC have two analog potentiometers, in this example one is used to simulate the temperature signal and the other is used to set the delay time of timer, when the temperature is larger than the set point and delay time is exceeded, a output channel indicate light will be on to give temperature alarm.

B.8.2 Wiring of Potentiometer

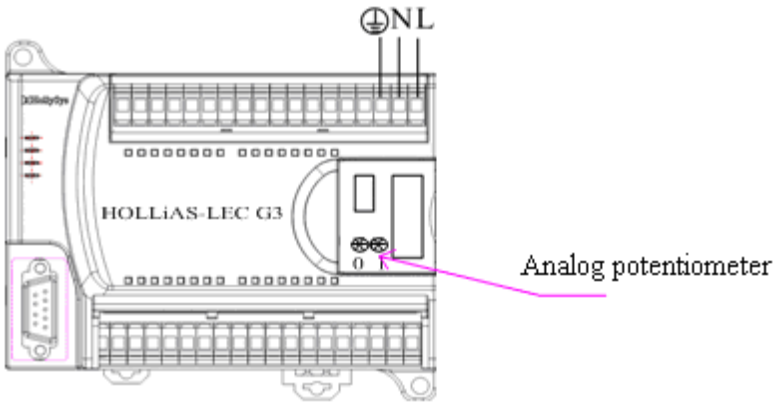


Figure B-8-1 Wiring (LM3106)

B.8.3 Variable Declaration

```
PROGRAM PLC_PRG
VAR
    EN: BOOL;
    POT1: POT;              get the value of channel 1
    TEMPERATURE: BYTE;      temperature value, the value of channel 0
    POT0: POT;              get the value of channel 0
    POT0_TERM: BOOL;
    TIME_DELAY: BYTE;       delay time, the value of channel 1
    POT1_TERM: BOOL;
    M1: BOOL;               middle variable, set point is 1
    TON1: TON;              timer
    TIM_ET: TIME;           time passed
END_VAR
```

B.8.4 PLC Configuration

Any type of CPU module of LM PLC can be used for the practical tests, and LM3106 is used in this example.

B.8.5 LD Program

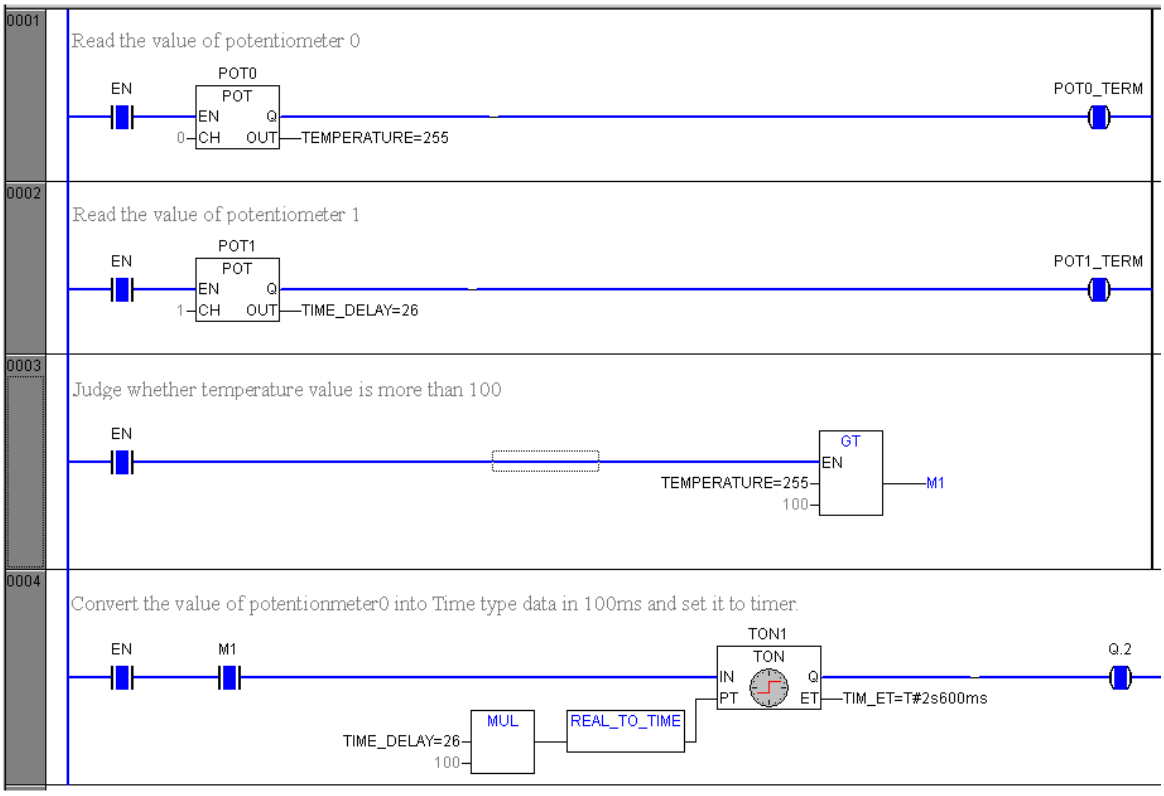


Figure B-8-2

B.8.6 Description

When EN is enabled, POT0 acquires the value of channel 0 and assigns it to TEMPERATURE; POT1 acquires the value of channel 1 and assigns it to TIME_DELAY. Because TEMPERATURE is larger than 100, M1 is set. Since PT must be of TIME type, REAL_TO_TIME is used to convert the data and channel Q0.2 is connected when the delay time is reached.