Mitsubishi University

L Series Programming (GX Works2)

TRPLC701P

L Series Programming (GX Works2) Training Manual

MEAU Manual Number R72-106-SLSASG-001

Revision History

Rev.	Date	Revision Notes
*	06/02/2010 C. Kellock	Internal, created in part from existing Q Series Programming (GX Works2) and iQ Works Structured Programming training manuals
A	06/29/2010 C. Kellock	First public release version

Disclaimer

Table of Contents

Class Introduction...1

Course Objectives 1
Prerequisites.. 1
Course Duration 1
Course Description 2
List of Relevant Manuals 3

LESSON 1 – Hardware Review..5

Lesson Objectives 5
1.1 Power Supplies.. 5
1.2 CPUs 6
1.3 Memory Drives.. 9
1.4 Discrete I/O 10
1.5 Intelligent Function Modules 12
1.6 Accessories.. 14
1.7 Overall System Configuration 15

LESSON 2 – Controller Basics...17

Lesson Objectives 17
2.1 Memory Areas... 17
2.1.1 Discrete I/O (X/Y)... 17
2.1.2 Memory Bits (M/L) 17
2.1.3 Data Registers (D) 17
2.1.4 Timers and Counters (T/ST/C)... 18
2.1.5 Fault Annunciators (F) 18
2.1.6 Step Relays (S).. 19
2.1.7 File Registers (R).. 19
2.1.8 Index Registers (Z) 20
2.2 Constants.. 20
2.3 System Addressing 21
2.4 EXERCISE – Addressing... 22
2.5 Ladder Review 23
2.6 Timers... 24
2.7 Timer Examples.. 27
2.8 Counters 29
2.9 Counter Examples 30

LESSON 3 – GX Works2 Introduction 31

Lesson Objectives 31
3.1 GX Works2 New and Improved Features 31
3.2 Backward Compatibility 32
3.3 File Format 33
3.4 Compress/Unpack 34
3.5 Launching GX Works2 35
3.6 Docking Windows 36
3.7 Toolbars 39
3.8 Customizing the Keyboard 40
3.9 Status Bar 41
3.10 Software Options 42

LESSON 4 – Creating a Project 43

Lesson Objectives 43
4.1 Simple vs. Structured Project 43
4.2 Creating a New Project 44
4.3 Editing Ladder Logic 45
4.4 EXERCISE – Basic Ladder Logic 46
4.5 Connection Setup 47
4.6 Transferring to the PLC 49
4.7 Reading from the PLC 52
4.8 Verify with PLC 53

LESSON 5 – Online Operations 55

Lesson Objectives 55
5.1 Ladder Logic Monitor 55
5.2 Device / Buffer Memory Batch Monitor 57
5.3 Watch Windows 59
5.4 Intelligent Module Monitor 60
5.5 Online Edits 61
5.6 Modify Value 63
5.7 Forced I/O Registration 64

LESSON 6 – GX Works2 Utilities 67

Lesson Objectives 67
6.1 Find/Replace Menu 67
6.2 Cross Reference 76
6.3 Device List 77
6.4 PLC Diagnostics 78
6.5 System Monitor 82
6.5.1 Module Detailed Information 83
6.5.2 Product Information List 86
6.6 Sampling Trace 87
6.7 EXERCISE – Sampling Trace 92

LESSON 7 – Special Addresses 93

Lesson Objectives 93
7.1 Special Memory Bits 93
7.2 Special Registers 95
7.3 Troubleshooting Examples 97
7.3.1 Battery Low Warning Indicator 97
7.3.2 PLC Error Code Backup 97
7.4 Real Time Clock 98

LESSON 8 – Intelligent Modules 99

Lesson Objectives 99
8.1 Intelligent Module Introduction 99
8.2 TO/FROM Instructions 101
8.3 U\G Addresses 104
8.4 Intelligent Function Utility 105
8.5 EXERCISE – Intelligent Module Access 110

LESSON 9 – PLC Parameters 111

Lesson Objectives 111
9.1 PLC Parameters 111
9.2 PLC Name 112
9.3 PLC System 113
9.4 PLC File 115
9.5 PLC RAS 116
9.6 Boot file 117
9.7 Program 118
9.8 SFC 118
9.9 Device 119
9.10 I/O Assignment 121
9.11 Built-In Ethernet 124
9.12 Built-In I/O Functions 125
9.13 Network Parameters 126

LESSON 10 – Label Programming 127

Lesson Objectives 127
10.1 What are Labels? 127
10.2 Registering Global Labels 129
10.3 Registering Local Labels 131
10.4 Automatic Assignment 132
10.5 Using Labels 133
10.6 EXERCISE – Global Labels 134

LESSON 11 – Structured Projects 135

Lesson Objectives 135
11.1 IEC 61131-3 135
11.2 IEC Addresses 137
11.3 Structured Project 140
11.4 Program Organization Unit 141
11.5 Tasks 143
11.6 Programs 145
11.7 Compiling the Program 146
11.8 Symbolic Information 147
11.9 EXERCISE – Structured Project 149

LESSON 12 – Structured Ladder 151

Lesson Objectives 151
12.1 Introduction 151
12.2 Editor Basics 152
12.3 Editing Modes 154
12.4 Ladder Symbols 155
12.5 Connecting Lines 157
12.6 Functions and Function Blocks 160
12.7 Constants 163
12.8 Program Documentation 164
12.9 Monitoring 165
12.10 EXERCISE – Structured Ladder 166
12.11 Timers and Counters 167
12.12 EXERCISE – Timers and Counters 171

LESSON 13 – Structured Text 173

Lesson Objectives 173
13.1 Introduction 173
13.2 Editor Basics 174
13.3 Operators 177
13.4 Syntaxes 177
13.5 Functions and Function Blocks 183
13.6 Monitoring 186
13.7 EXERCISE – Structured Text 187
13.8 Inline Structured Text Box 188
13.9 EXERCISE – Inline Structured Text Box 190

APPENDIX 191

APPENDIX 1 – L Series USB Driver Installation 193

Class Introduction

Welcome to the L Series Programming with GX Works2 training course. This course is intended for designers and control engineers, responsible for developing application programs using the L Series programmable logic controllers. This class will use the GX Works2 programming software.

Course Objectives

By the end of this training course, the student should be able to:

- Identify the models of the L Series family.
- Understand the system design and addressing of the L Series.
- Write, download, monitor, and debug programs.
- Troubleshoot and debug systems utilizing the L Series.

Prerequisites

Before attending this class, it is strongly advised that the student should attend the PLC Basics (TRPLC003B) training class. If not attending the PLC Basics class, experience working with PLCs from any manufacturer would be required. This class does NOT cover the basics of PLCs.

Course Duration

This course is designed for a 3 day class length.

Course Description

LESSON 1 – Hardware Review

This lesson will introduce the various hardware components used to design an L Series programmable controller.

LESSON 2 – Controller Basics

This lesson discusses memory areas, address allocation, and basics of ladder logic, including timers and counters.

LESSON 3 – GX Works2 Introduction

This lesson introduces the GX Works2 programming software.

LESSON 4 – Creating a Project

This lesson discusses creating a new project and communication with the PLC.

LESSON 5 – Online Operations

This lesson will explain tools for monitoring and editing programs online.

LESSON 6 – GX Works2 Utilities

This lesson explains some of the troubleshooting tools built into GX Works2.

LESSON 7 – Special Addresses

This lesson will review the special bits and words in the PLC memory.

LESSON 8 – Intelligent Modules

This lesson explains the methods for communication with intelligent modules.

LESSON 9 – PLC Parameters

This lesson discusses the settings made in the PLC parameters.

LESSON 10 – Label Programming

This lesson discusses the use of labels instead of addresses when programming.

LESSON 11 – Structured Projects

This lesson introduces the concepts of structured programming.

LESSON 12 – Structured Ladder

This lesson demonstrates the structured ladder programming language.

LESSON 13 – Structured Text

This lesson introduces the structured text programming language.

List of Relevant Manuals

Hardware Manuals

SH(NA)080888	L I/O Module User’s Manual
SH(NA)080889	LCPU User's Manual (Function Explanation, Program Fundamentals)
SH(NA)080890	LCPU User's Manual (Hardware, Design, Maintenance, Inspection)
SH(NA)080891	LCPU User’s Manual (Built-In Ethernet Function)
SH(NA)080892	LCPU User’s Manual (Built-In I/O Function)
SH(NA)080893	LCPU User’s Manual (Data Logging Function)

Programming Manuals

SH(NA)080782	Q/L/FX Structured Programming Manual (Fundamentals)
SH(NA)080783	Q/L Structured Programming Manual (Common Instructions)
SH(NA)080784	Q/L Structured Programming Manual (Application Functions)
SH(NA)080785	Q/L Structured Programming Manual (Special Instructions)
SH(NA)080809	Q/L Programming Manual (Common Instructions)

GX Works2 Manuals

SH(NA)080779	GX Works2 Version 1 Operating Manual (Common)
SH(NA)080780	GX Works2 Version 1 Operating Manual (Simple Project)
SH(NA)080781	GX Works2 Version 1 Operating Manual (Structured Project)
SH(NA)080787	GX Works2 Version 1 Beginner’s Manual (Simple Project)
SH(NA)080788	GX Works2 Version 1 Beginner’s Manual (Structured Project)
SH(NA)080921	GX Works2 Operating Manual (Intelligent Function Module)

LESSON 1 – Hardware Review

This lesson discusses the hardware structure of the L Series programmable controller. This includes a review of the different CPU types, input/output modules, and intelligent modules.

Lesson Objectives

At the conclusion of this lesson, you will be able to…

- List the modules required for an L Series control system.
- List some factors to be considered when specifying hardware.

The L Series controller is a modular PLC which does not require a backplane for modules to communicate with the CPU. The backplane is built into the modules, and the modules all have slide lock levers used to connect to the module before it on the PLC. The power supply is the left-most module in the system, followed by the CPU. On the right side of the CPU, I/O modules, network modules, and intelligent modules can be connected. The bus is terminated with an end cover, which is included with the CPU or CC-Link IE Field head station.

1.1 Power Supplies

The power supply module provides +5VDC power to all of the modules that are connected to it. Each module installed has a required current draw specification. The total current draw of the modules should not exceed the capability of the supply.

There are 2 basic power supplies available:

Part Number	Supply Voltage	Bus (5VDC)
L61P	120/240VAC	5A
L63P	24 VDC	5A

Notes

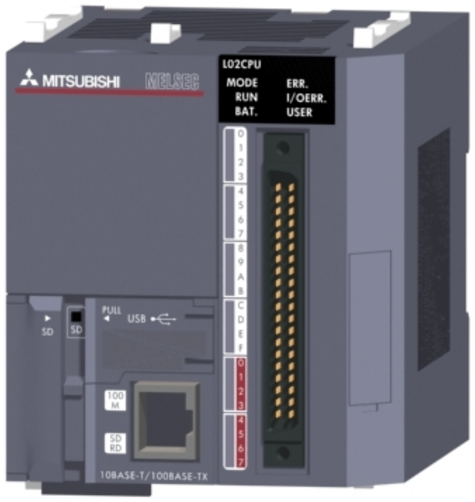
1.2 CPUs

The CPU module provides the processing power to the L Series controllers.

All L Series CPUs have some key built-in features. The list below details some of the important features.

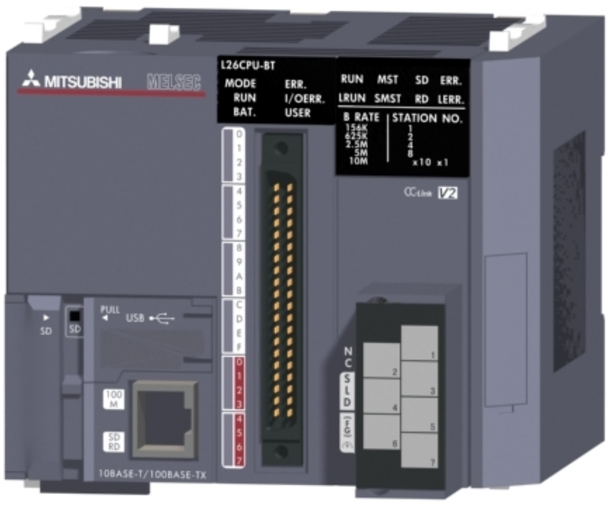
- USB programming port
- Ethernet communications
- SD/SDHC memory card slot
- 16 built-in inputs, offering functions such as general input, interrupt input, high speed counter input, pulse catch input
- 8 built-in outputs, offering functions such as general output or high speed pulse positioning outputs
- Built-in high speed data logging function
- Location for mounting optional display module
- Bus terminating end cover included with CPU module

The first processor available is the L02CPU. The L02CPU offers control of up to 1024 I/O points and 20K of program memory, with a minimum instruction speed of 40ns.



Notes

The high end processor is the L26CPU-BT, which offers up to 4096 I/O points and 260k of program memory, with a minimum instruction speed of 9.5ns. The L26CPU-BT also includes built-in CC-Link version 2 functionality.



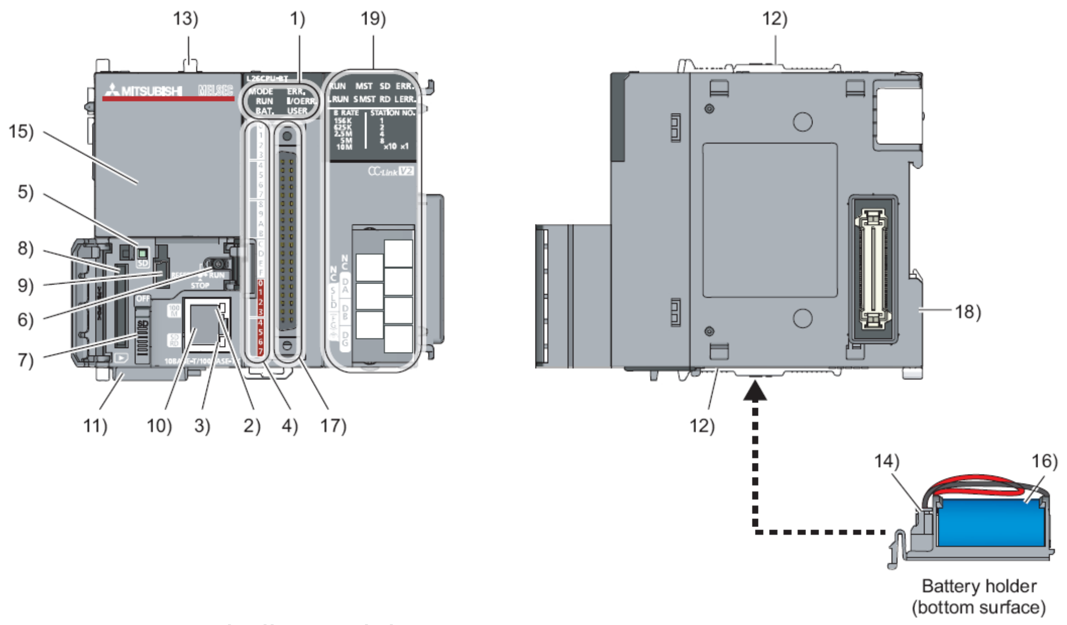
The table below is a basic comparison of the CPU performance specifications.

Feature	L02CPU	L26CPU-BT
Program Memory	20k	260k
Minimum Execution Speed	40ns	9.5ns
Max I/O Points	1024	4096
Built-In I/O Functions	Yes	Yes
Built-In Ethernet	Yes	Yes
Built-In CC-Link	No	Yes
Max CC-Link Networks	2	4

The CPUs are also sold as a set, which includes the L61P power supply, selected CPU, and L6DSPU display unit. These can be ordered by adding **–SET** to the end of the CPU part number.

Notes

An overview of the CPU hardware is shown below.



1. CPU indicator lights
2. 100M Ethernet speed light
3. Ethernet send/receive indicator light
4. Input and output indicator lights
5. SD memory card status light
6. RUN/STOP/RESET switch
7. SD memory card lock switch
8. SD memory card slot
9. USB connector
10. Ethernet port
11. Serial number display
12. Module locking levers
13. Display module locking lever
14. Battery connector
15. Display unit cover
16. Battery
17. I/O connector
18. DIN rail hook
19. Built-in CC-Link (L26CPU-BT only)

Notes

1.3 Memory Drives

All L series processors have 4 memory areas referred to as drives. Each drive can store certain components of the processor’s configuration. Drive numbers are indicated in the table below. Drives 0, 3, and 4 are built into the CPU module. Drive 2 is the SD memory card, if installed.

Each drive can store certain sections of the processor’s data, as shown below.

○ : Storable, ×: Not storable

File type	Program Memory	Standard RAM	Standard ROM	SD memory card	File name and extension (any given name for ***)	Remarks
	Drive 0	Drive 3	Drive 4	Drive 2		
Parameter	○	×	○	○	PARAM.QPA	One file per drive
Intelligent function module parameters[*1]	○	×	○	○	IPARAM.QPA	One data per drive
Program	○	×	○	○	***.QPG	—
Device comment	○	×	○	○	***.QCD	—
Initial device value	○	×	○	○	***.QDI	—
File register	×	○[*2]	×	×	***.QDR	—
Local device	×	○[*2]	×	×	***.QDL	One file per module
Sampling trace	×	○[*2]	×	×	***.QTD	—
PLC user data	×	×	○	○	***.CSV/BIN	—
Source information (simple project) [*4]	○	×	○	○	SRCINFOM.C32	—
Source information (structured project) [*4]	○	×	○	○	SRCINFOI.C32	—
Drive heading	○	×	○	○	QN.DAT	—
Device data storage file	×	×	○	×	DEVSTORE.QST	—
Module error collection file	×	○[*2]	×	×	IERRLOG.QIE	—
Data logging setting file	×	×	○	○	LOGCOM.QLG, LOG01 to 10.QLG	—
Data logging file	×	×	○[*3]	○	***.CSV	—
Menu definition file	×	×	○	○	MENUDEF.QDF	—

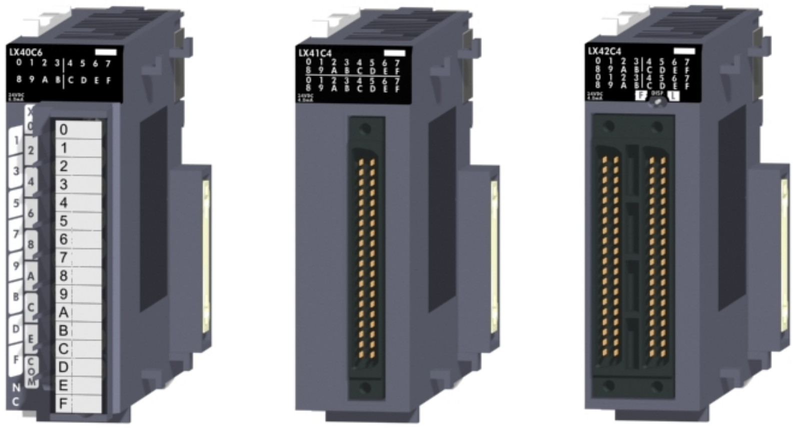
*1 Store parameters (PARAM.QPA) and intelligent function module parameters (IPARAM.QPA) in the same drive. Otherwise the intelligent function module parameters are invalid.
*2 Only one file can be stored.
*3 This drive cannot be selected as a storage file by the data logging function. To write data to this drive, perform Write PLC User Data.
*4 The data in which the information of label program configuration is stored.
GX Works2 Version 1 Operating Manual (Common)

Notes

1.4 Discrete I/O

Each CPU has a total of 24 I/O points built in, using the high density 40-pin connector on the front of the CPU.

Input and output modules in the L Series are available in a number of module types and I/O counts. Input or output modules are available in 16, 32, or 64 points per module. Modules have either inputs or outputs.



Modules with 16 points all come with removable screw-type terminal blocks for I/O wiring. Modules with 32 or 64 points utilize high-density connectors.

Notes

Cables and terminal blocks to connect to the high density connectors are available, or the user can make their own cables with a field installable plug.

All high density modules use the same style 40-pin connector. The connector options for the high density modules are shown below.

- **A6CON1** for solder type
- **A6CON2** for crimp contacts
- **A6CON3** for flat cable

Cables which have this connector already attached at one end, and ferrules on the loose wires at the other end, have been made available as well. Those part numbers are shown below.

- **LCBL40P-xM** where x is length in meters (2, 5, or 10)

Terminal blocks which are compatible with the high density modules are also available. These terminal blocks are listed below. Note these terminal blocks cannot be used with the I/O connector built into the CPU.

Name	Model Name	A6TBXY36	A6TBXY54	A6TBX70
Input module*1	LX41C4	○	○	○
	LX42C4	○	○	○
Output module	LY41NT1P	○	○	×
	LY42NT1P	○	○	×

*1 Applicable only when using the positive common type module.

Cables for these terminal blocks are available in lengths from 0.5 meters to 10 meters.

Another terminal block, **A6TE2-16SRN**, can be used to connect to the DC output modules, and provides individual relays for each output. Each module has 16 outputs, so the cable from the module connects to 2 of these terminal blocks. Cables for this module are available from 0.6 meters to 10 meters.

Notes

1.5 Intelligent Function Modules

All modules which are not discrete inputs or discrete outputs are called intelligent function modules. Some examples of intelligent modules would include analog inputs and outputs, high speed counter, positioning, motion control, serial communications, and networking.

All can be set up easily with built-in tools in GX Works2.

Analog Inputs and Outputs

There are 2 types of analog modules: analog input modules and analog output modules. Both are based on varying current or voltage, usually 4-20 mA, 0-10V, or –10 to + 10 V, as set by the programmer.

With an input module, the actual current or voltage reading is converted into an integer value based on the selected full scale. The analog output modules receive an integer value from the controller, and output a voltage or current signal based on that value.

Model	Channels	Description
L60AD4	4	Voltage or current inputs
L60DA4	4	Voltage or current outputs

High Speed Counter

Ordinary counters in the CPU are dependent on the scan time in 2 ways. The updating of the input that is used as the counting input and the updating of the accumulated value of the counter are part of the CPU's program scan. This may be too slow for high speed counting applications. These modules provide the high speed counting ability, up to 500 KHz for these higher speed applications.

Model	Channels	Description
LD62	2	200KHz sink output
LD62D	2	500KHz differential line driver

Notes

Positioning

These modules provide functions for the control of Servo and Inverters. They can control multiple axes of motion, and provide functions like S-curve, acceleration/ deceleration, linear and circular interpolation, and a variety of origin point return methods.

Different modules offer different types of servo compatibility, and motion control capabilities.

Model	Description
LD75P4	Open collector pulse output motion modules
LD75D4	Differential line driver pulse output motion modules
LD75MH4	Motion modules for use with MR-J3-B servos over SSCNETIII fiber optic cable

Serial Communications

These modules provide RS-232 and RS-422 communication ports. These modules provide connections for computers, barcode readers and other external devices. They can be also be used to provide another programming connection to the controller.

Model	Channels	Description
LJ71C24N	2	(1) RS232 and (1) RS422/485
LJ71C24N-R2	2	(2) RS232

Networking

Network modules provide the controller with a variety of communications capabilities. Networks can be used to make multiple controllers communicate with each other, to connect remote racks of modules to the controller, or to provide remote I/O points on the network. There are various network topologies and communication formats available.

A quick list of network modules is shown below.

Module	Network
LJ61BT11	CC-Link Master/Local Module
LJ72GF15-T2	CC-Link IE Field Head Module

Notes

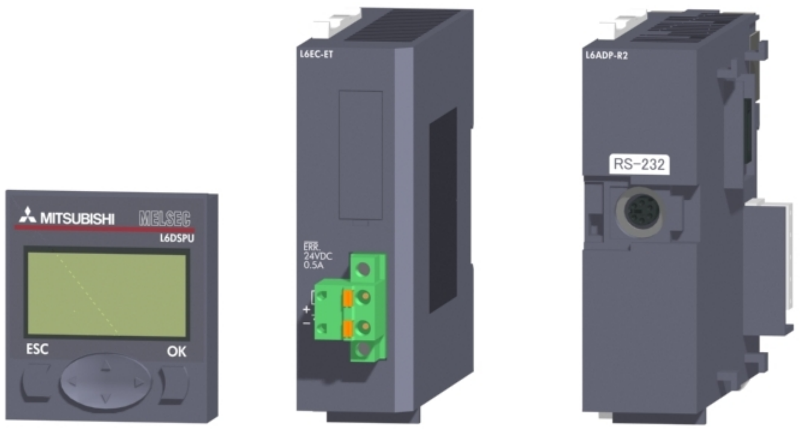
1.6 Accessories

A variety of accessories exist for use with the L Series controllers. Some are detailed below.

L6DSPU Display Unit
The L6DSPU display unit can be installed into the front of the CPU. Data in the PLC can be monitored or adjusted via the display unit. PLC error information will be displayed on the display module when an error is active. Using the UMSG instruction in the PLC program, the programmer can cause text messages of their choice to be displayed on the display module.

L6EC-ET End Cover with Error Terminal
All Q Series controllers offer an alarm output relay on the power supply module. Since this is not built into the L Series power supply, this optional replacement end cover includes a relay contact closure to indicate a CPU error. This end cover installs at the right end of the PLC in place of the cover which comes with the CPU

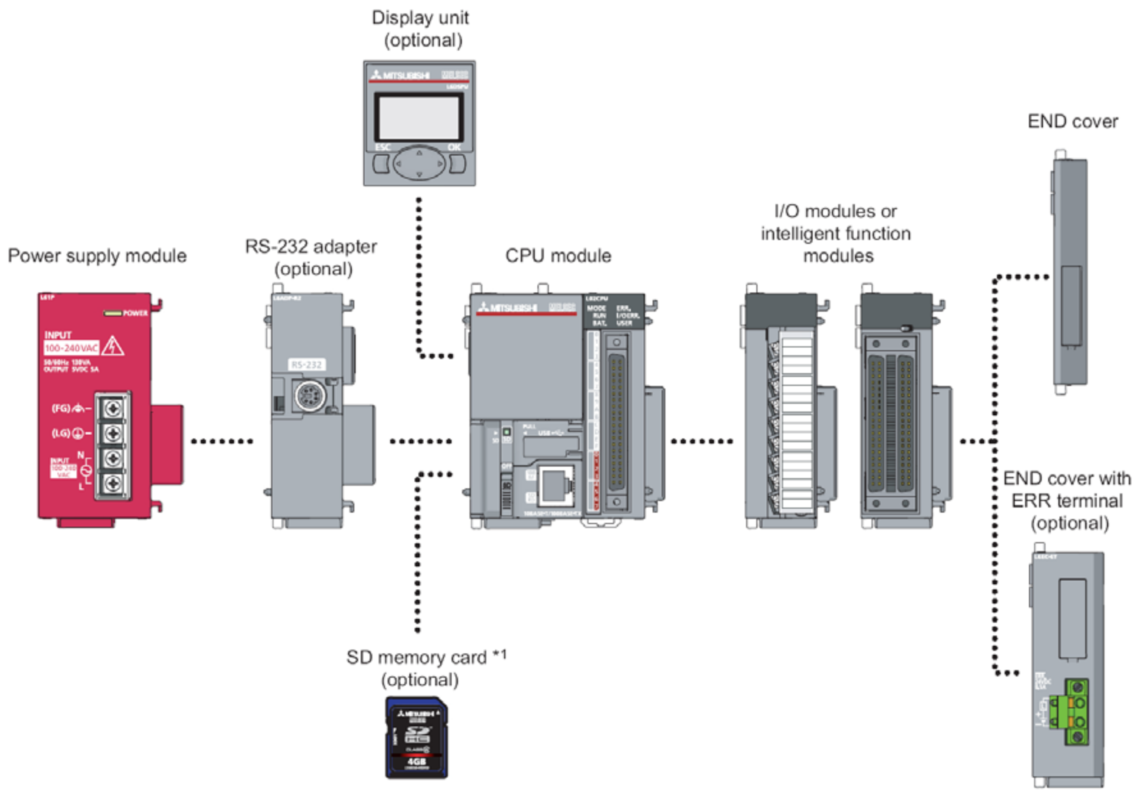
L6ADP-R2 RS232 Connection
The L6ADP-R2 is designed to provide the 6-pin round RS232 port, as found on some of the Q Series processors. It is used for serial connection of a GOT to the processor. This module installs on the left of the CPU between the processor and the power supply.



Notes

1.7 Overall System Configuration

The diagram below shows the typical system configuration of an L02CPU based control system.


Display unit (optional)
END cover
I/O modules or intelligent function modules
Power supply module
RS-232 adapter (optional)
CPU module
END cover with ERR terminal (optional)
SD memory card *1 (optional)

Notes

Notes

LESSON 2 – Controller Basics

This lesson will cover the various areas of the PLC's memory, addressing, and ladder basics.

Lesson Objectives

At the conclusion of this lesson, you will be able to…

- Identify the different areas of the PLC's memory.
- Explain I/O addresses for an L Series control system.
- Understand basic ladder logic programming.

2.1 Memory Areas

The internal device memory in the PLC is broken into many different areas based on their function. Some of these addresses refer to single bits, while some refer to 16-bit registers. This section is a review of the memory areas covered in the PLC Basics class, and includes some additional memory areas.

2.1.1 Discrete I/O (X/Y)

- ✓ X represents a physical input.
- ✓ Y represents a physical output.
- ✓ Each address refers to a single bit.
- ✓ In the L Series, inputs and outputs are addressed in hexadecimal.

2.1.2 Memory Bits (M/L)

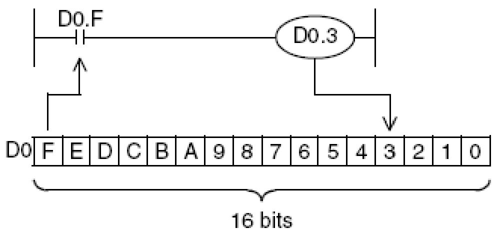
- ✓ M represents a memory bit.
- ✓ L represents a memory bit which is latched by the PLC battery.
- ✓ Memory bits are addresses in decimal.
- ✓ The number of each address type is adjustable.

2.1.3 Data Registers (D)

- ✓ D represents a data register.
- ✓ Data registers are addresses in decimal.
- ✓ The number of data registers is adjustable.

Notes

In L Series, the PLC has the ability to directly access bits within a word. By placing a decimal point between the word address and bit address, you can reference the status of a single bit within a word. An example would be D100.0 which references the least significant bit in word D100.



2.1.4 Timers and Counters (T/ST/C)

- ✓ T addresses are used to represent a timer.
- ✓ ST addresses are used to represent retentive timers.
- ✓ C addresses are used for counters.
- ✓ Timers and counters are addressed in decimal.
- ✓ The number of each address type is adjustable

Timers and counters will be covered later in this lesson.

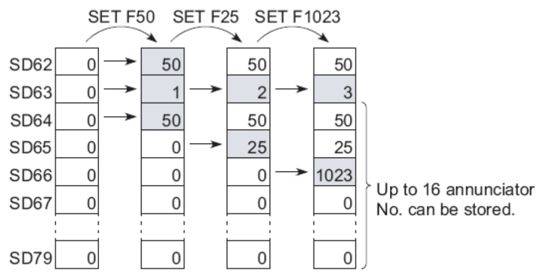
2.1.5 Fault Annunciators (F)

An F indicates a fault annunciator address, which can be used by the user to create fault indications in the controller.

When an F bit is turned on, the number of that bit as well as the number of active annunciators is stored into special memory addresses. The USER LED on the front of the CPU is also on when any F bits are on.

Data stored at special registers (SD62 to 79)

- Nos. of annunciators which switched ON are stored in order at SD64 to 79.
- The annunciator No. which was stored at SD64 is stored at SD62.
- "1" is added to the SD63 value.



Notes

2.1.6 Step Relays (S)

Step relays are used for programming in the sequential function chart (SFC) programming language. SFC is similar to a flowchart, and each block of the chart is assigned a number.

Each S address refers to a single bit. This bit is used to indicate which step in the flowchart is currently active. As program execution transfers from one step to the next, step bits will be set and reset.

If not using SFC programming, the S addresses can NOT be used as internal bits. This will cause an SFC error and the PLC will be stopped.

2.1.7 File Registers (R)

File registers are additional numeric data storage locations in the PLC. These registers operate in similar fashion to the D registers. They can be used for 16-bit numbers or 32-bit numbers the same way standard data registers are, in all of the same commands.

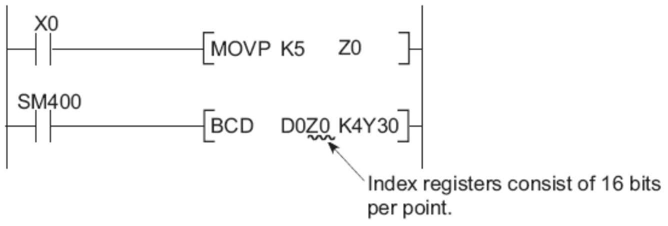
File registers are configured by default in the L Series, but are created in data registers (D). The amount and allocation can be adjusted in PLC Parameters if required. The number of file registers and their storage location is adjustable.

Notes

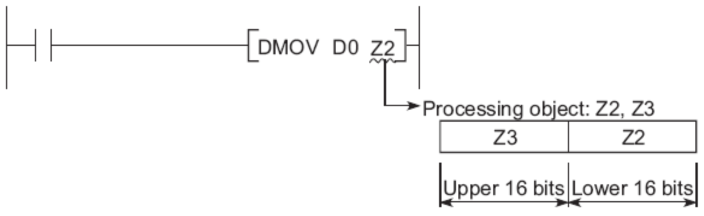
2.1.8 Index Registers (Z)

Index registers are used for offset addressing. They are indicated by Z addresses. A number is stored in a Z register, and that number is used as an offset for an address used in a PLC program. There are 20 index registers available in the L Series processors.

The index registers can be used to modify most addresses in the PLC, as well as numeric constants in decimal and hexadecimal.



Index registers can also be used for 32-bit offsets, in which case 2 consecutive index registers are used to store the offset value.



2.2 Constants

To use numeric data in PLC commands, it must be prefixed with a letter to indicate the type of numeric data. Numbers can be put in decimal, hexadecimal, or real number format depending on the commands used and the prefix assigned to the number.

- ✓ K indicates a decimal numeric value (16 or 32-bit).
- ✓ H indicates a hexadecimal numeric value (16 or 32-bit).
- ✓ E indicates a floating point numeric value.

Text can also be entered in the ASCII format by enclosing it in double quotes.

- ✓ “Mitsubishi’ is an ASCII text string.

Notes

2.3 System Addressing

This section will quickly review the rules of I/O allocation for the L Series. A more thorough review is part of the PLC Basics class.

- All I/O addressing can be done automatically or set in PLC Parameters. The CPU will read the connected modules at power-up. Manual configuration is not required.
- L Series uses a free addressing system. Addresses are allocated based on the density of modules.
- Addresses are allocated in blocks of 16. All I/O addressing is in hexadecimal. If an 8-point module is used, it uses addresses 0-7, and addresses 8-F are lost. Each module's starting address always ends in 0.
- Intelligent modules are referred to by their head address, which is the first I/O address assigned to the intelligent module. Intelligent modules can have both X and Y addresses.
- L Series CPUs have 16 inputs and 8 outputs built in. These will occupy X0-XF and Y0-Y7 in the default configuration.
- L26CPU-BT has a CC-Link module built in, which is a 32 point intelligent module. It will use addresses 10-2F in the default configuration.

Notes

2.4 EXERCISE – Addressing

Determine the correct I/O addressing of the L Series controller on the trainer unit.

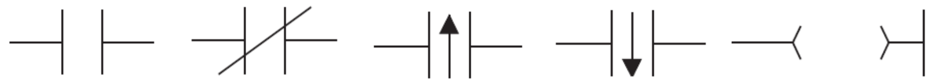
Notes

2.5 Ladder Review

This section will review the basics of ladder logic. This material is covered in detail in the PLC Basics training class.

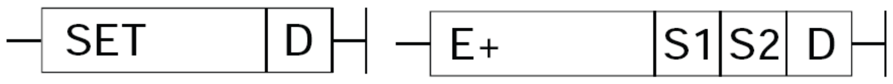
Ladder logic is composed of many symbols. The ladder logic language is designed to mimic an electrical power flow diagram, promoting ease of use with electricians.

The basic symbols of ladder logic are contacts and coils. Inputs are represented as contacts. Contacts with arrows are pulsed, either on the rising or falling edge of the input signal. Outputs are represented as a coil.



AND conditions in logic are created by drawing contacts in series. OR conditions are created by drawing contacts in parallel.

Other instructions are all represented as function blocks. The number of parameters a function block requires depends on the instruction.



Some basic ladder logic commands include:

- SET – latches a bit on
- RST – resets a latched bit (or a register)
- BKRST – block reset a range of addresses
- PLS – rising edge pulse
- PLF – falling edge pulse
- FF – alternates the state of a bit

In order for a rung of code to be considered valid, it must have a minimum of one input condition and one output condition. Input conditions are contacts and output conditions are coils or instructions.

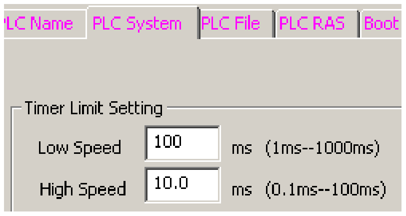
Notes

2.6 Timers

There are two basic types of timers available in the Q Series or L Series processors. These two types are low speed and high speed timers. The difference between low speed and high speed timers is the time base intervals which can be timed. The actual preset time base for each type is adjustable in PLC parameters, and applies to all timers of that type in the entire project.

- Low speed timers have presets in increments of 1 to 1000ms
- High speed timers have presets in increments of 0.1ms to 100ms.

To adjust the time bases, a setting must be made in the PLC Parameters. On the PLC system tab, the time base settings are in the top left as shown below.



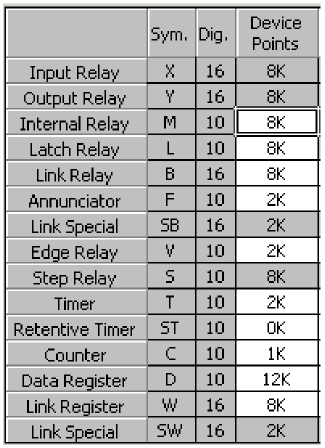
Timer Limit Setting

Low Speed 100 ms (1ms--1000ms)

High Speed 10.0 ms (0.1ms--100ms)

Either type of timer is available in a non-retentive or a retentive form. The non-retentive timers will reset when deactivated. Retentive timers will retain their current value when deactivated, and will continue from that value when reactivated.

Notes

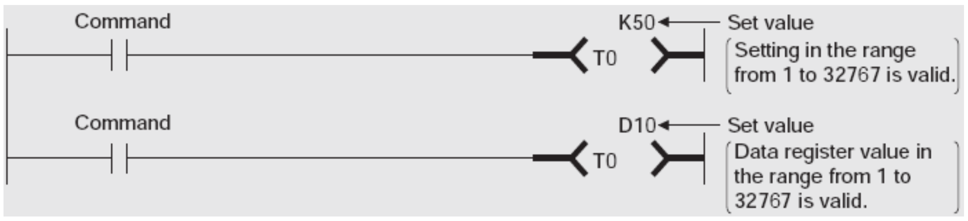


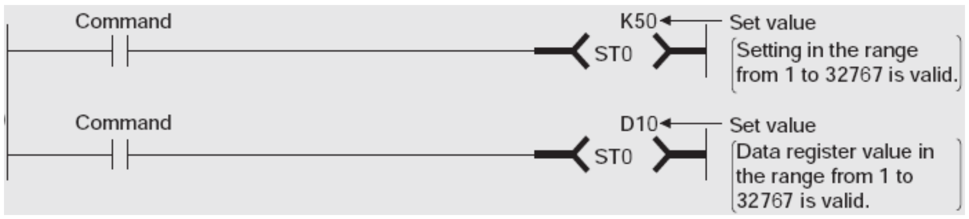
	Sym.	Dig.	Device Points
Input Relay	X	16	8K
Output Relay	Y	16	8K
Internal Relay	M	10	8K
Latch Relay	L	10	8K
Link Relay	B	16	8K
Annunciator	F	10	2K
Link Special	SB	16	2K
Edge Relay	V	10	2K
Step Relay	S	10	8K
Timer	T	10	2K
Retentive Timer	ST	10	0K
Counter	C	10	1K
Data Register	D	10	12K
Link Register	W	16	8K
Link Special	SW	16	2K

Retentive timers are not allocated by default. In order to use retentive timer addresses, memory must be reserved for their storage on the Device tab in PLC parameters. To do this, enter a number in the 'Device Point' column next to Retentive timer (ST). This number must be a multiple of 16, or input in K (1024 addresses per K). It may be necessary to reduce another address range to make room for these new timer addresses.

When coding a timer in a ladder program, the output coil symbol is used. A timer address is referenced for the location to store this timer's current value. A preset is also given to tell the timer how long it is to run. This preset can be a fixed numeric value or the value of a value in a data register.

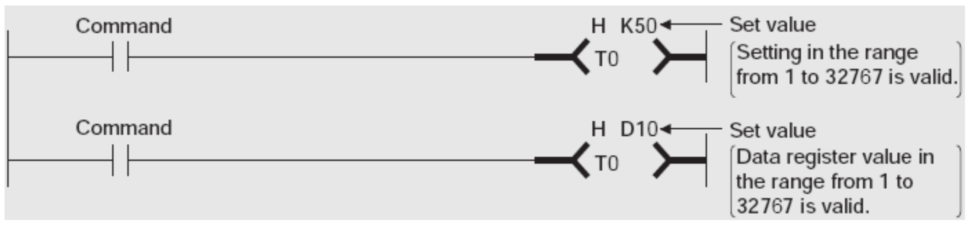
To enter a low speed timer, draw an output coil, and enter the timer number, followed by a space, and then the preset value.

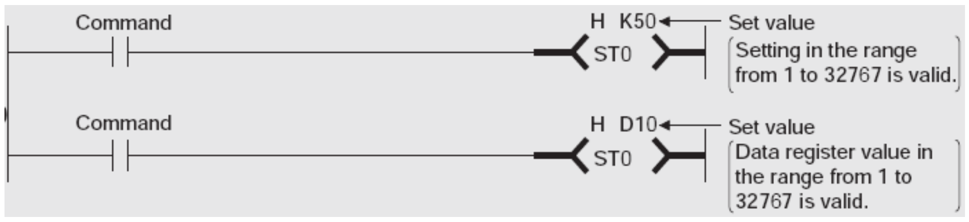

Command
K50
Set value
T0
Setting in the range from 1 to 32767 is valid.
Command
D10
Set value
T0
Data register value in the range from 1 to 32767 is valid.


Command
K50
Set value
ST0
Setting in the range from 1 to 32767 is valid.
Command
D10
Set value
ST0
Data register value in the range from 1 to 32767 is valid.

Notes

To enter a high speed timer, draw an output coil, enter an ‘H’ followed by a space, then the timer address, and then a space, and then the preset value.


Command
H K50
Set value
T0
Setting in the range from 1 to 32767 is valid.
Command
H D10
Set value
T0
Data register value in the range from 1 to 32767 is valid.


Command
H K50
Set value
ST0
Setting in the range from 1 to 32767 is valid.
Command
H D10
Set value
ST0
Data register value in the range from 1 to 32767 is valid.

To check for the completion of the timer in the logic, a contact is coded on the same address as the timer. The H for high speed is not required on the contact.

Timers have some basic limitations in their operation:

- Timers time up from zero to their preset, so negative presets are not allowed.
- All timers are 16-bit, so largest preset value is 32,767.
- Timers do NOT count past their preset value.

The method to reset a timer depends on whether or not it is a retentive timer.

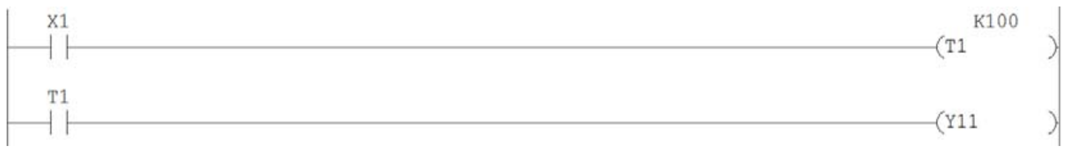
- Non-retentive timers are reset when the logic in front of the timer coil turns false. They can also be reset with the RST instruction.
- Retentive timers require the use of the RST instruction.
- If not marked as latched in the PLC parameters, the timer value will be lost when controller power is cycled.

Notes

2.7 Timer Examples

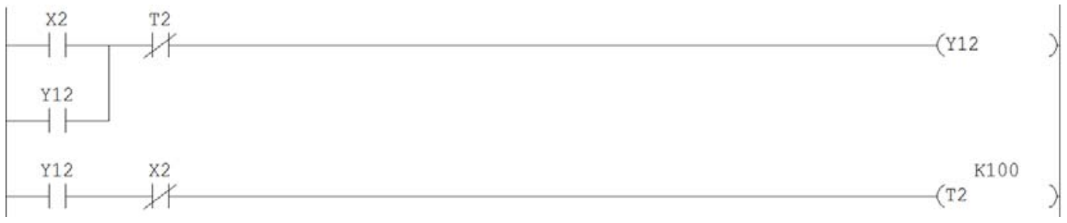
Since all timers time up to their preset values, it is necessary to write code to create various types of timers. Some examples are below.

On Delay Timer



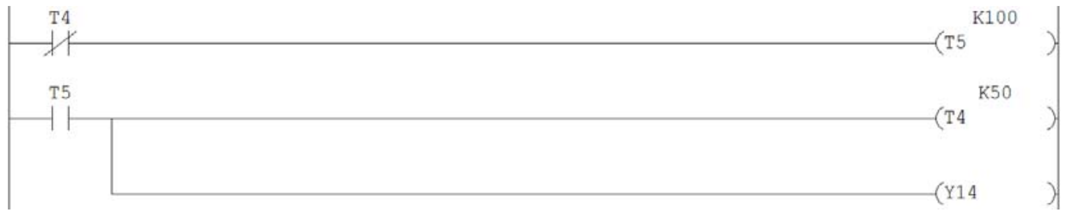
- Output Y11 will turn on 10 seconds after input X1.

Off Delay Timer



- Output Y12 will turn on with X2 and remain on for 10 seconds after X2 turns off.

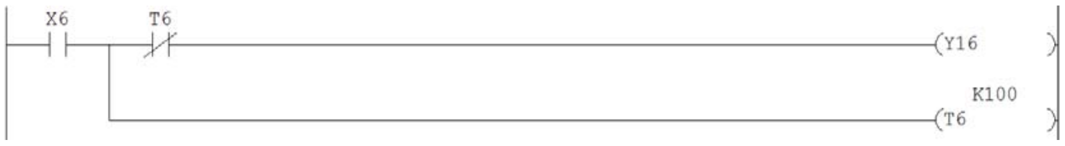
Flip Flop Timer



- Output Y14 will be on for 5 seconds and off for 10 seconds, repeating as long as the CPU is in RUN mode.

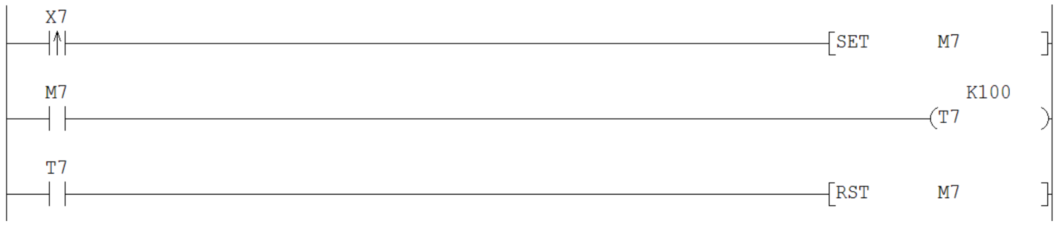
Notes

One Shot Timer



- Output Y16 will turn on with X6 and turn off after 10 second, even if X6 stays on longer than 10 seconds. If X6 turns off in less than 10 seconds, the output on Y16 will be the same length as the input.

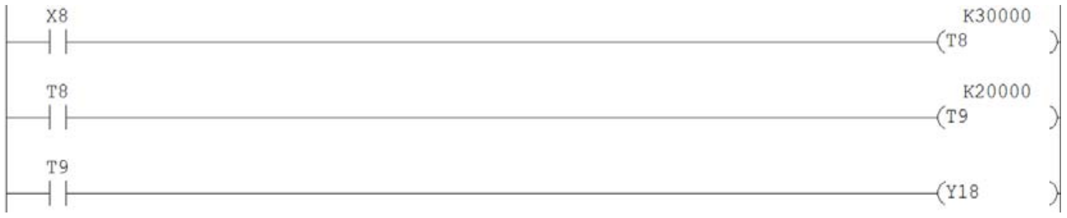
One Shot Fixed Length Timer



- Same as above, but the M7 output will always be 10 seconds long, no matter how short of a time the X7 input signal remains on.

Cascaded Timers

Since the maximum preset is 32,767 and a typical low speed timer is set in 10th of a second, timers cannot exceed 3276.7 seconds. To time for 5000 seconds, two timers can be cascaded as shown below.



- Y18 turns on after X8 has been on for 5000 seconds (preset would be 50,000 (out of range) with a single timer).

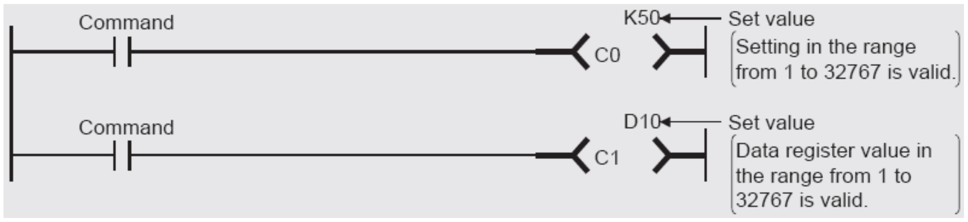
Notes

2.8 Counters

All L Series and Q Series CPUs have 1024 counters configured by default. As with the timers, the allocation of memory can be modified to allocate more or less counter addresses.

Counters operate by incrementing a value each time the logic in front of the counter coil is turned on. This function only counts once, on the rising edge of the input condition, so the input condition does not need to be pulsed.

Counters are coded as an output coil, just as timers. A preset must be defined when the counter is created. The preset is a 16-bit number, and can be a fixed numeric value or a reference to a value in a data register.


Command
C0
K50
Set value
Setting in the range from 1 to 32767 is valid.
Command
C1
D10
Set value
Data register value in the range from 1 to 32767 is valid.

To check for the completion of a counter in ladder logic, code a contact with the counter address.

Counters have some basic limitations in their operation:

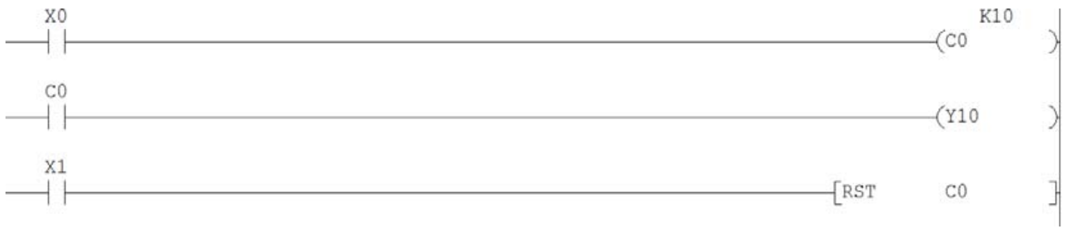
- Counters count up from zero to their preset, so negative presets are not allowed.
- All counters are 16-bit, so largest preset is 32,767.
- Counters do NOT count past their preset value.

Counters are reset using the RST instruction. If not marked as latched in the PLC parameters, the count value will be lost when controller power is cycled.

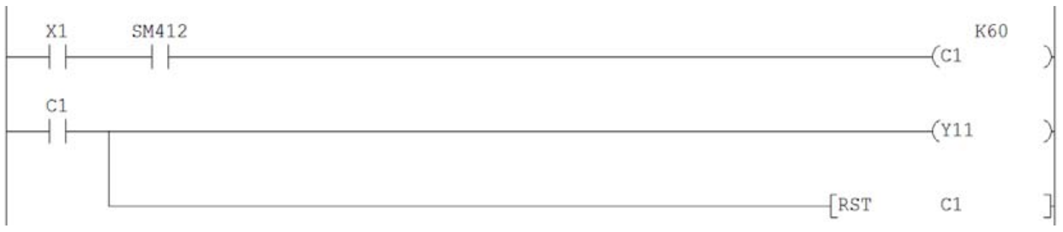
Notes

Since counters only count up, some users will attempt to cause a counter to count down using the DEC (decrement) instruction. This is not advised, since the counter completion signal will not be modified by instructions other than the counter coil. If the counter is completed and a DEC is performed, the counter will lower its count by one, but the completion indicator will NOT turn off. By the same token, if the INC (increment) instruction is used, the counter may go past its preset value, and the completion indicator will not be updated.

2.9 Counter Examples



- Each time X0 turns on, counter C0 increases by one.
- When C0 reaches its preset, output T10 turns on.
- Y10 will remain on, and the counter will not count again, until the counter is reset by X1 turning on.



- When X1 is on, count seconds (using SM412, 1-second clock pulse).
- Output Y11 will turn on for one scan when the counter completes, and the counter will be reset and continue to run.

Notes

LESSON 3 – GX Works2 Introduction

This lesson is intended to introduce the GX Works2 software and allow the user to understand the options for customizing the user interface.

Lesson Objectives

At the conclusion of this lesson, you will be able to…

- Understand the new features of GX Works2.
- Customize screen layout, keyboard settings, and software preferences.

3.1 GX Works2 New and Improved Features

Many improvements were made to the GX Works2 programming software, making it easier to use and more user friendly. Some of them are listed below.

Improved Ease of Use

- GX Works2 operating manuals are installed during the installation process and readily available from the Help menu.
- Help menu now offers helps on all instructions, similar to the contents of the programming manual, available by pressing F1 on an instruction.
- Function block libraries, with drag and drop usage from selection window.
- Some function blocks can be resized in the structured ladder programming mode, allowing more inputs to certain instructions.
- Text of label names and function names supports auto-complete while typing in the editor windows.
- Compress and unpack utilities in the Project menu to package a project for transmission on disk or email.

Improved User Interface

- More use of docking windows in GX Works2, allowing commonly used components to be docked anywhere in the workspace.
- Toolbars automatically change based on the type of open window.
- Many keyboard shortcuts are able to be modified by the end user.
- Screen colors can be customized for user preferences

Notes

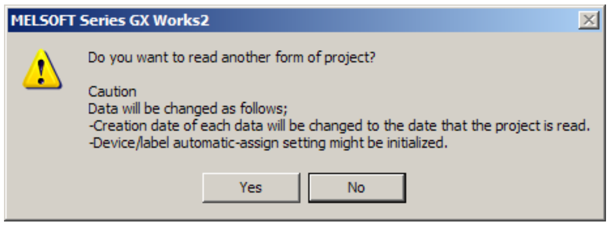
New Features

- Built-in comments for special relays and registers can be imported into project from right click in comment list.
- Built-in comments for special function modules, also available for insertion by right click menu in comment list.
- Complete project revision history function, allowing multiple versions of the program to be registered into the same project, verified against each other, or restored.
- Multi-level project security, allowing the programmer to choose which security levels have access to which portions of a project.

3.2 Backward Compatibility

The GX Works2 software is capable of reading projects which were written in GX Developer for any PLC type available in GX Works2. These projects can be opened using the 'Open Other Data' option in the Project menu.

To open a GX Developer file, navigate to the directory which contained the 'gppw.gpj' file in the GX Developer project directory. This dialog will confirm reading of the GX Developer data type.

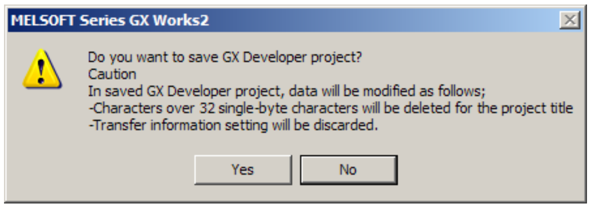


Do you want to read another form of project?

Caution
Data will be changed as follows;
-Creation date of each data will be changed to the date that the project is read.
-Device/label automatic-assign setting might be initialized.

GX Works2 can save simple projects back into a GX Developer compatible file format. This allows programs which were modified in GX Works2 to be opened in older versions of GX Developer.

In the Project menu, select 'Export to GX Developer Format File…' to start the utility and indicate where to save the project. Saving of structured projects in the GX Developer format is not allowed.



Do you want to save GX Developer project?
Caution
In saved GX Developer project, data will be modified as follows;
-Characters over 32 single-byte characters will be deleted for the project title
-Transfer information setting will be discarded.

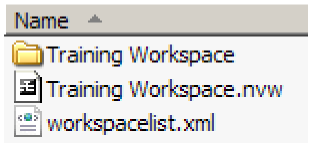
Once 'Yes' is selected, a project will be created in the GX Developer format for use in the GX Developer software.

Notes

3.3 File Format

All iQ Works titles use a new file storage format, which is consistent throughout the packages. Projects are now saved into a workspace. Each workspace can contain projects from GX Works2, GT Works3, and MT Works2.

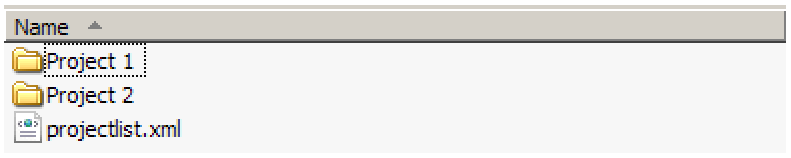
In the directory where the workspace is created, there will be a file called 'workspacelist.xml' which contains a list of the workspaces found in this directory.



Name
Training Workspace
Training Workspace.nvw
workspacelist.xml

Each workspace is stored in a directory with the same name as the workspace. If MELSOFT Navigator was used to create a workspace, there will also be a file with the same name as the workspace with the extension '.nvw' which stores the Navigator configuration.

In each workspace directory, there will be a file called 'projectlist.xml' which contains a list of the projects found in this workspace.



Name
Project 1
Project 2
projectlist.xml

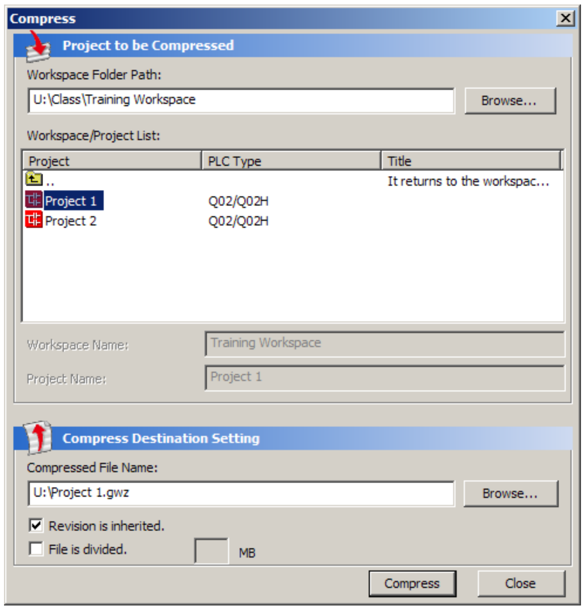
Each project in the workspace will have its own subdirectory in the workspace folder. All files relayed to that project are stored in the subdirectory. Double clicking on the 'Project.gd2' file within that directory will open the project in GX Works2.

When moving files from one drive to another, it is imperative that the 'workspacelist.xml' file and the directory named for the workspace is included with all of its subdirectories. Without the 'workspacelist.xml' file, the projects cannot be opened.

Notes

3.4 Compress/Unpack

Using the Compress option on the Project menu will compact all required files into a single file, making it easier to move or send via email. The compressed file can be set to inherit all registered revisions, and can be divided into smaller pieces automatically.



Workspace Folder Path: U:\Class\Training Workspace

Workspace/Project List:

Project	PLC Type	Title
..		It returns to the workspac...
Project 1	Q02/Q02H	
Project 2	Q02/Q02H	

Workspace Name: Training Workspace

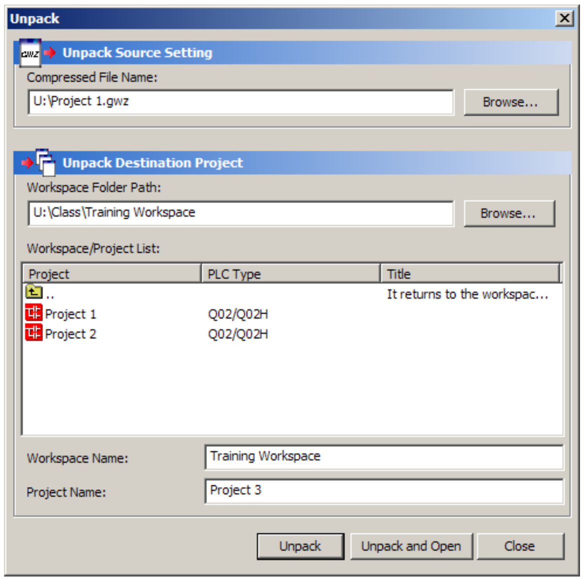
Project Name: Project 1

Compressed File Name: U:\Project 1.gwz

[✓] Revision is inherited.

[ ] File is divided. MB

The Unpack utility allows the compressed file to be expanded back into a project in an existing workspace. It can be extracted with a different project name than it was created with, in case a project of the same name already exists on the destination computer.



Compressed File Name: U:\Project 1.gwz

Workspace Folder Path: U:\Class\Training Workspace

Workspace/Project List:

Project	PLC Type	Title
..		It returns to the workspac...
Project 1	Q02/Q02H	
Project 2	Q02/Q02H	

Workspace Name: Training Workspace

Project Name: Project 3

Notes

3.5 Launching GX Works2

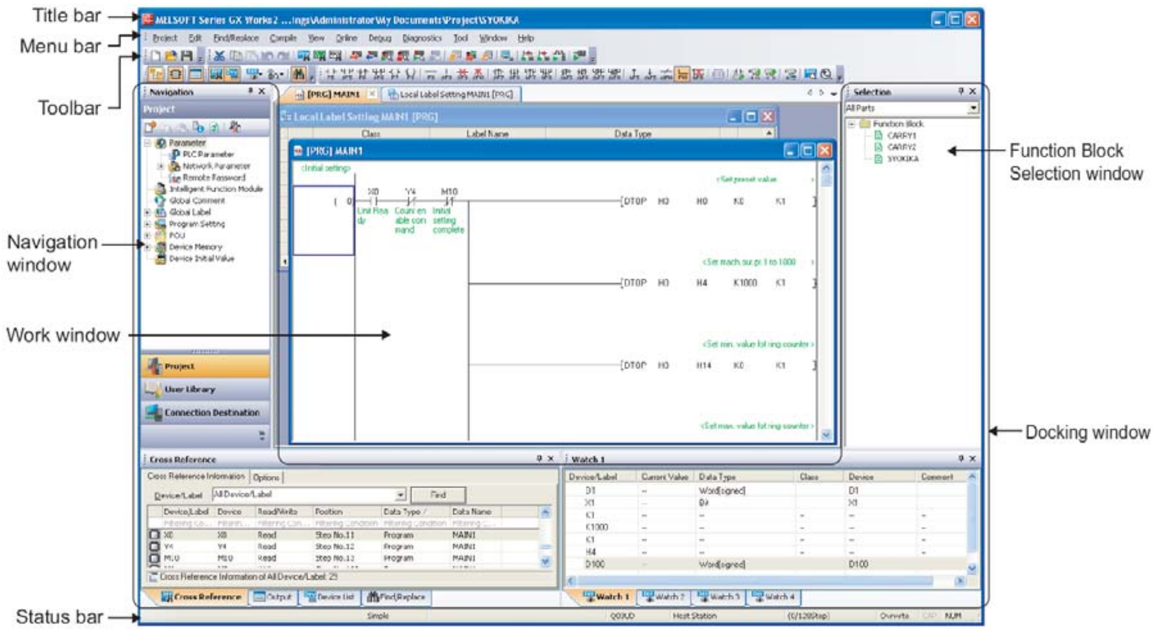
GX Works2 is Windows based programming software. Because it is a Windows package, it can be started from the Start menu.

Start -> All Programs -> MELSOFT Application -> GX Works2

The GX Works2 installation process creates an icon on the desktop when it completes, which can be used to start GX Works2.

GX Works2 can also be started automatically from within MELSOFT Navigator by double clicking on a PLC with a project attached.

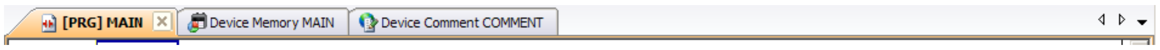
An example of the main GX Works2 screen is shown below.



The GX Works2 screen is broken into many components. Many of these components, such as toolbars and certain windows, can be docked at various locations around the screen, or their display can be turned off.

Notes

The work window on GX Works2 is broken into a series of tabs to indicate open windows. It can also be tiled, cascaded, or arranged as windows.

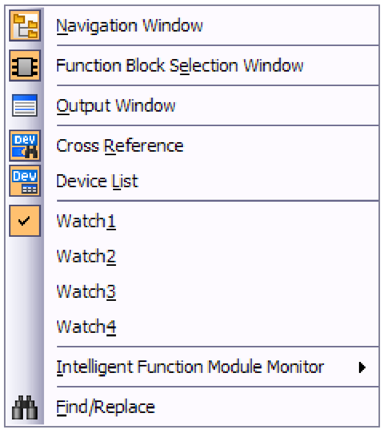


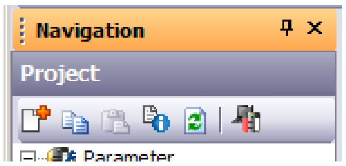
Font size in the ladder window can be increased or decreased with the Text Size option in the View menu. It is also available from the right click menu in the ladder window. Display fonts and colors can also be adjusted in the View menu.

3.6 Docking Windows

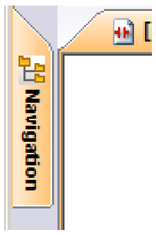
There are a variety of windows which can be docked to the edges of the screen in GX Works2. These windows can be turned on and off using the Docking Window option in the View Menu.

The docking windows which are enabled will have an orange box behind the icon to the left of the window name in the menu.



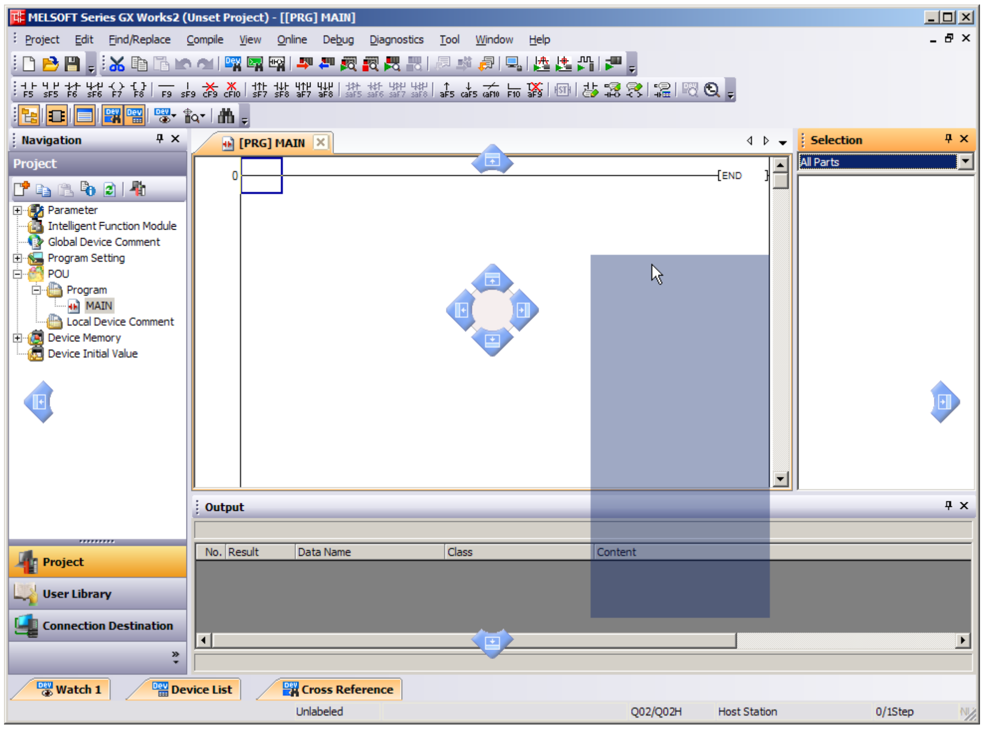


Any of the docking windows can be turned on or off in the View menu or with the X button in the upper right corner. Docking windows will have a pin in the upper right corner, and when clicked, it will turn the window into a tab which slides in from the side of the screen. Docking windows can be moved to any side of the screen.



Notes

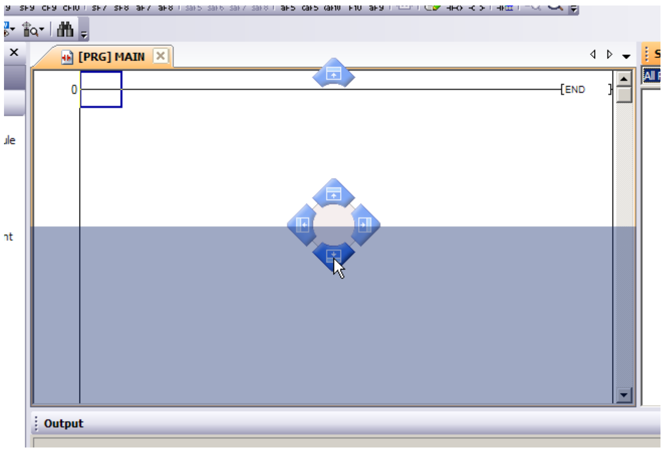
Docking windows can be moved to various locations on the screen. To move a docking window, click and hold the left mouse button on its title bar, and move the mouse. Arrows will appear as shown below.



If the mouse is not over an arrow, the window becomes a floating window. When the mouse moves over an arrow, it will dock to that side of the workspace.

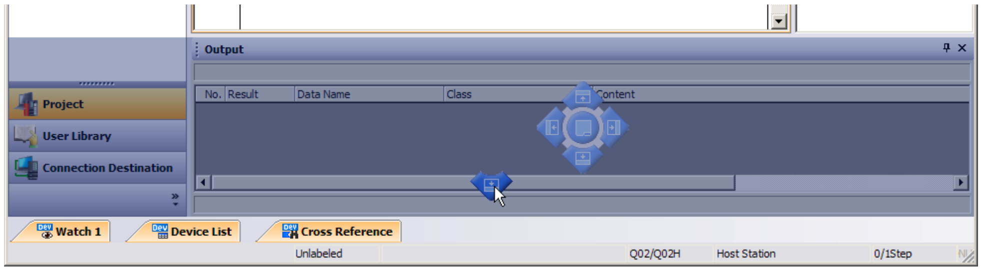
Notes

When the mouse is over one of the arrows in the center, the docking window will be placed on that edge of the current window, as shown by the darkened section on the screen.



If there is a center button in the center arrow block, this will make the window being moved into another tab in that window.

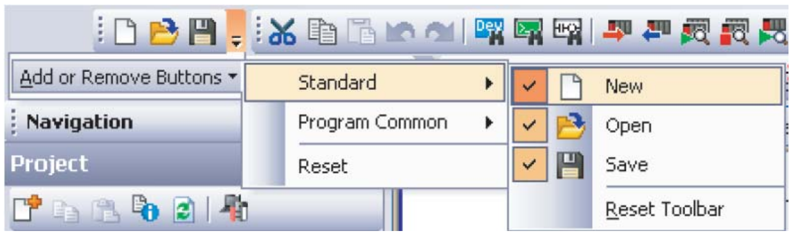
When the mouse is positioned over one of the arrows on the outside edge of the screen, that docking window will be located completely across that side of the screen, as shown below.



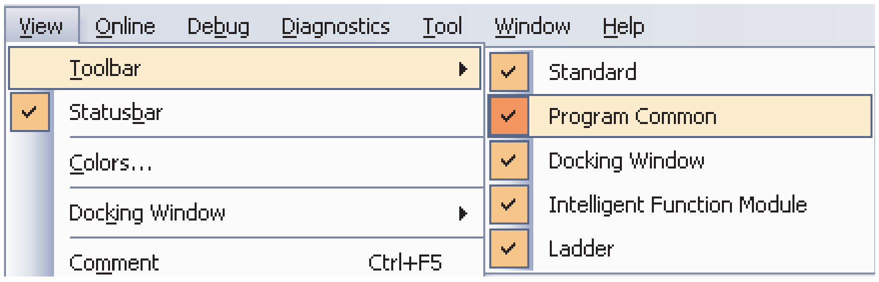
Notes

3.7 Toolbars

Toolbars in GX Works2 can be customized with the small arrow at the end of each toolbar. Check or un-check a tool to update the toolbar contents.



Complete toolbars can be turned on or off from the View menu. Toolbars can be repositioned or docked by dragging them around the screen.



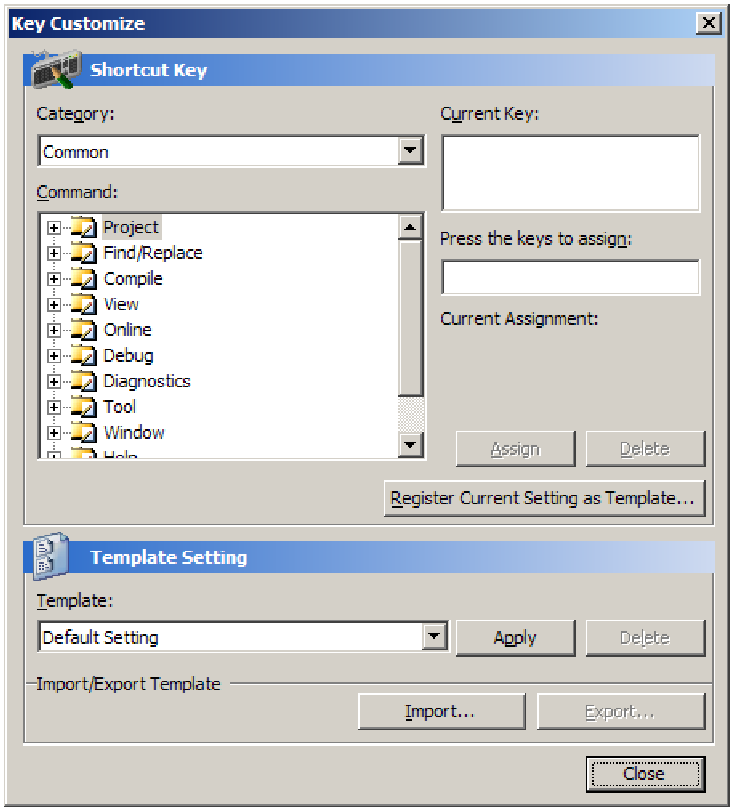
The toolbars being displayed will change automatically as different windows are open, such as the ladder toolbar in the ladder windows, or the device memory toolbar in a device memory window.

A complete list of all toolbars, buttons, and shortcut keys can be found in Appendix 1 of the GX Works2 Operating Manual (Common).

Notes

3.8 Customizing the Keyboard

Keyboard shortcuts used within GX Works2 can be customized using the Key Customize option in the Tool menu.



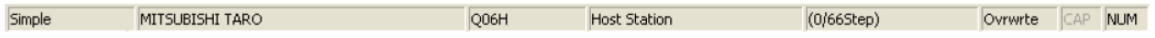
On the left side of the window, select the command from the selection list. The current keyboard shortcut (if any) will be displayed in the top right window. To set a new key, click in the window under 'Press a key to assign' and enter the new keystroke.

Changes can be saved as a template, and the default template can be restored from this window as well. Templates can be imported and exported for use on other computers.

Notes

3.9 Status Bar

The status bar on the bottom of the GX Works2 window shows the status of various settings within the project for quick reference. The status bar is broken up as indicated below.



Simple	MITSUBISHI TARO	Q06H	Host Station	(0/66Step)	Ovrwrte	CAP	NUM

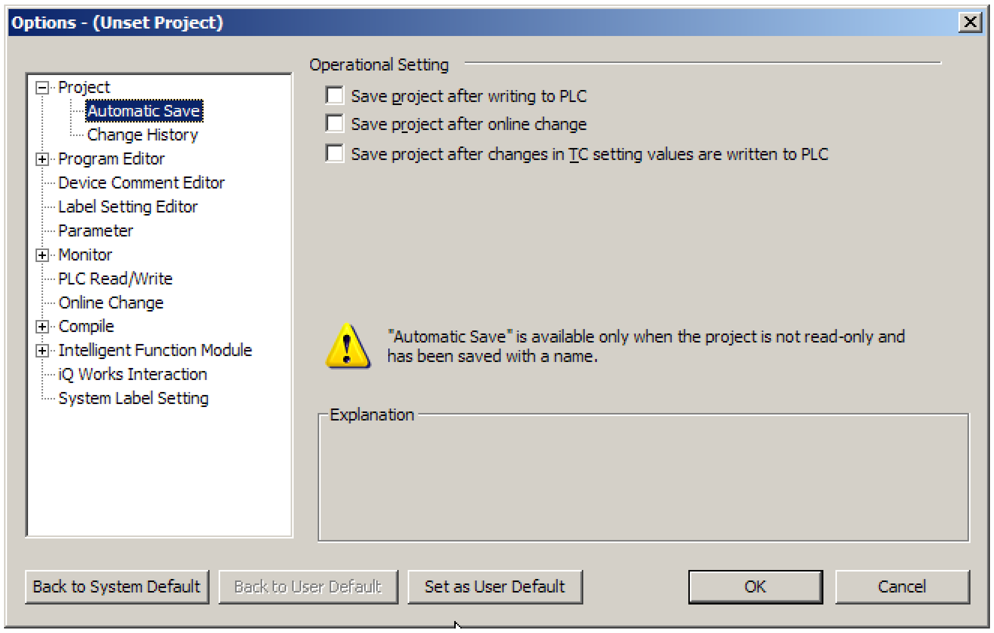
Project type	Security information	Programmable controller type	Connection destination	Cursor position	Insert/ Overwrite	Caps Lock	Num Lock

- The project type will be one of 3 settings
    - ‘Unlabeled’ for a simple project without labels
    - ‘Simple’ for a simple project with labels
    - ‘Structured’ for a structured project
- Security information will display the name of the currently logged in user then a project with security settings active is opened
- Programmable controller type will display the model number of the CPU which this project is configured for
- Connection destination will display the route being used in Connection Settings to connect to the CPU
- Cursor position will display the cursor position in the open window
- The next box will display ‘Insert’ or ‘Overwrite’ depending on the edit mode in the software, which can be changed by the INS key on the keyboard
- The last two boxes show the status of the CAPS LOCK and NUM LOCK keys on the keyboard

Notes

3.10 Software Options

The basic software options in GX Works2 are found in the Tool menu under Options. On the left side of this window is a tree structure, which divides the option settings into groups based on their function.



Operational Setting

- [ ] Save project after writing to PLC
- [ ] Save project after online change
- [ ] Save project after changes in TC setting values are written to PLC

"Automatic Save" is available only when the project is not read-only and has been saved with a name.

At the bottom of this window is a button to set the current options as a user default. Another button will allow the restoration of that user default if changes have been made. There is also a button to return to the factory defaults.

Several of the option settings will be discussed in more detail in other sections of this training manual.

Notes

LESSON 4 – Creating a Project

This lesson will explain creating a project, editing basic ladder, and writing to the CPU in GX Works2.

Lesson Objectives

At the conclusion of this lesson, you will be able to…

- Create a new project in GX Works2.
- Enter ladder logic into the project.
- Configure communication settings for connection to the PLC.
- Read and write projects to/from the CPU.

4.1 Simple vs. Structured Project

There are two program types which can be created in GX Works2. These are called simple or structured projects.

Simple projects allow the use of the standard Mitsubishi programmable controller instruction set. Simple projects offer the same functions which were available in GX Developer.

Simple projects can be written in:

- Ladder
- Sequential Function Chart (SFC)
- Structured Text (ST)

Structured projects offer more functionality by offering IEC 61131-3 compliant programming in multiple languages. In addition to standard Mitsubishi instruction set, the IEC standard libraries are available in a structured project. This is similar to the older GX IEC Developer software.

Notes

Structured programs can be written in:

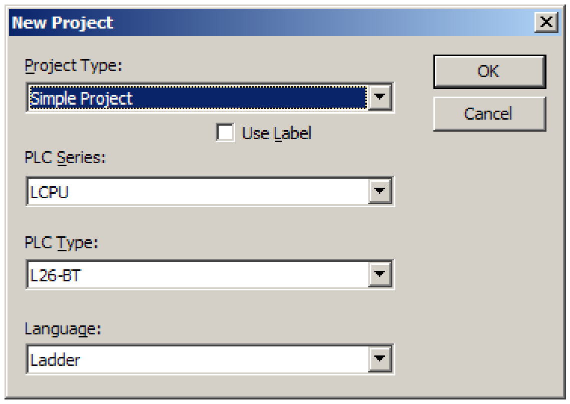
- Ladder
- Structured Ladder
- Sequential Function Chart (SFC)
- Structured Text (ST)

More detail on the various programming languages will be provided in later lessons in this class.

4.2 Creating a New Project

There are 3 ways to accomplish most tasks in GX Works2.

- Shortcut key on keyboard
- Button on toolbar
- Pull down menu item



Project Type: Simple Project
Use Label
PLC Series: LCPU
PLC Type: L26-BT
Language: Ladder

The first button on the Standard toolbar is a white sheet of paper. The shortcut key is Ctrl-N. New Project can be selected from the Project menu. This will present the New Project dialog box.

On this dialog box, the project type is selected.

If label based programming is desired, please check the Use Label box. Label programming allows names to be assigned to PLC addresses to facilitate coding.

Select the PLC series and PLC type for your controller. PLC Series should be L Series. Select the PLC Type based on the provided training hardware.

Label programming and structured programs are covered later in this class. For now, select ‘Simple Project’, and do not select to use labels.

Notes

4.3 Editing Ladder Logic

Ladder logic can be entered in 3 ways.

- Buttons on the toolbars can be pressed to draw a symbol where the cursor is positioned.
- Double clicking inside the cursor box will allow the symbol to be selected from a drop down list.
- A keyboard shortcut exists for each ladder logic symbol.

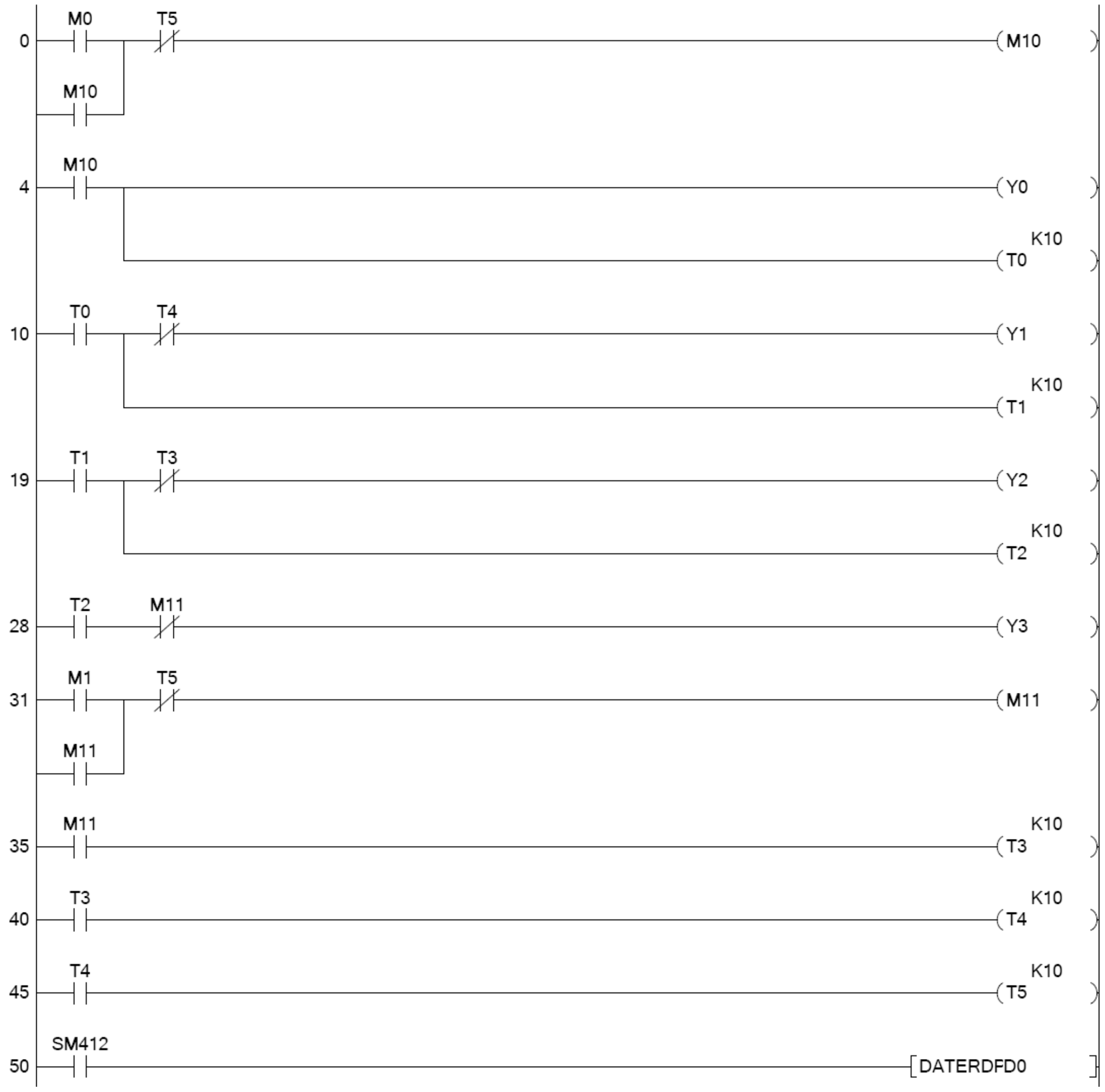
After creating or modifying ladder logic, the code is highlighted in gray. The gray portion of the code has not been compiled. Compiling is the process GX Works2 uses to review the logic changes and verify that the structure of the ladder logic is valid. The logic can be compiled with the F4 shortcut key, or by selecting Build from the 'Compile' menu.

If there are errors, such as no input condition or unconnected lines, the compile will fail with an error message. The software will not allow a project to be downloaded or saved until the code is compiled without errors.

Notes

4.4 EXERCISE – Basic Ladder Logic

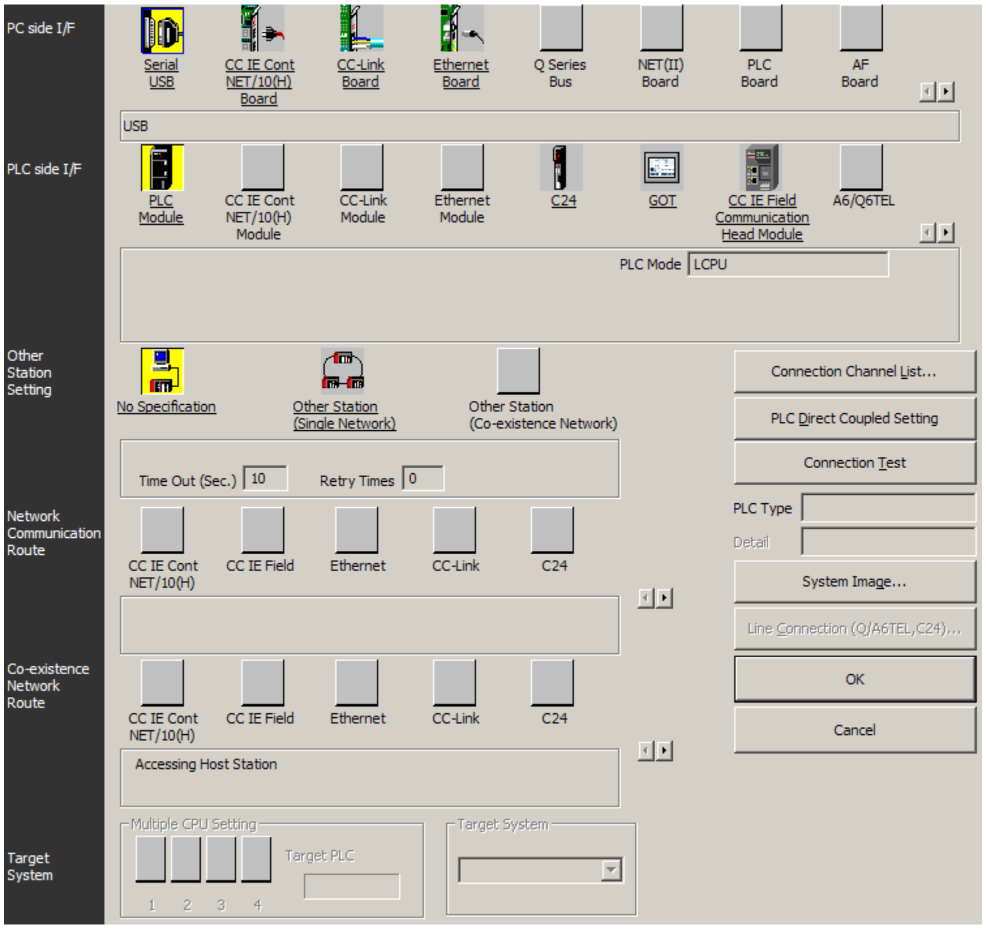
Enter the program shown here. Do not download to the PLC yet.



Notes

4.5 Connection Setup

The first step in transferring the program to the PLC is the configuration of the connection options. In the navigation window is a button called Connection Destination. There is a configuration called Connection1 here by default. Multiple configurations can be defined here. When one is double clicked, a screen similar to the one below is shown.



Notes

The first row of options on the top of the screen is the PC side interface. This is where the connection on the PC end is configured. The options available depend upon the PLC type selected.

The second row of options is the PLC side interface. This row is used to configure the connection at the PLC end. GOT Transparent Mode now has its own setting in this row.

The yellow boxes indicate the currently selected method. Double clicking on many of these buttons will bring up more detailed configuration data.

The ‘Connection test’ button will allow the configuration to be tested to the connected PLC. If all settings are correct, a dialog box indicating successful connection will be displayed.

The ‘Connection Channel List’ button will allow the selection of configuration by picture. It shows all available connection methods. When one is selected and the Update button is pressed, the settings to accomplish this will be made in the Transfer Setup screen.

Pressing the X button in the upper right hand corner or clicking the ‘Close’ button will result in settings not being saved. To save the changes, the Transfer Setup screen must be closed by pressing the ‘OK’ button.

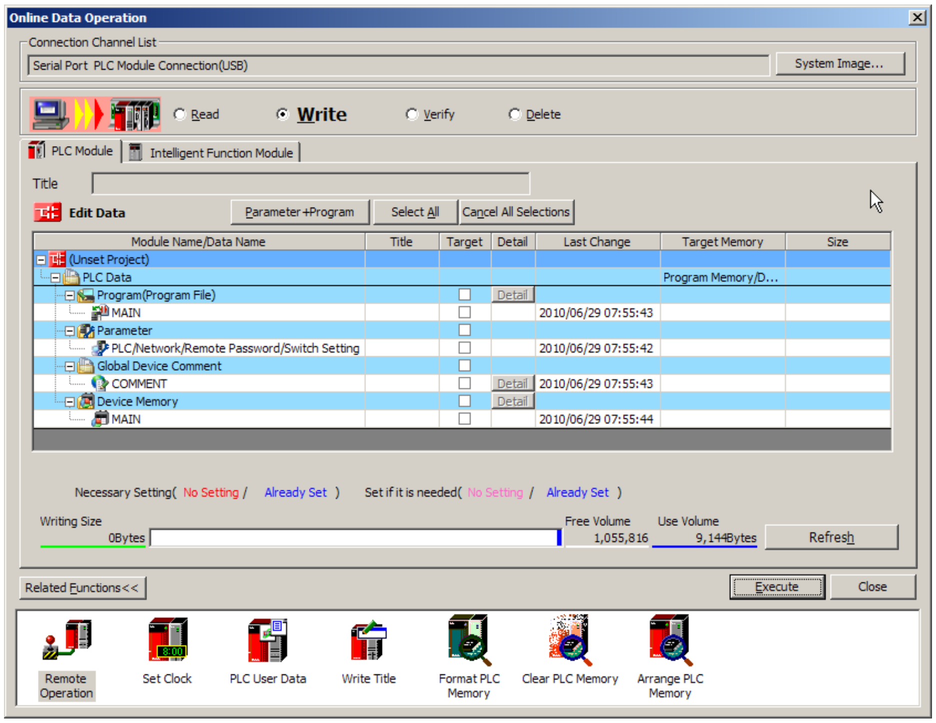
Set the appropriate settings for the connection method being used on the trainer unit.

With GX Works2, multiple connection configurations can be defined. By right clicking on a connection in the Connection Destination list, that connection can be copied, renamed, deleted, or set as the default connection. This allows multiple communication paths to be configured in a single project and easily switched.

Notes

4.6 Transferring to the PLC

Once the connection has been configured, the program can be written to the PLC using the ‘Write to PLC’ option in the ‘Online’ menu.



Online Data Operation

Connection Channel List

Serial Port PLC Module Connection(USB)

System Image...

Read [✓ Write] Verify Delete

PLC Module | Intelligent Function Module

Title

Edit Data

Parameter+Program | Select All | Cancel All Selections

Module Name/Data Name	Title	Target	Detail	Last Change	Target Memory	Size
(Unset Project)						
PLC Data					Program Memory/D...	
Program(Program File)		☐	Detail			
MAIN		☐		2010/06/29 07:55:43		
Parameter		☐				
PLC/Network/Remote Password/Switch Setting		☐		2010/06/29 07:55:42		
Global Device Comment		☐				
COMMENT		☐	Detail	2010/06/29 07:55:43		
Device Memory		☐	Detail			
MAIN		☐		2010/06/29 07:55:44		

Necessary Setting(No Setting / Already Set) Set if it is needed(No Setting / Already Set)

Writing Size 0Bytes

Free Volume 1,055,816

Use Volume 9,144Bytes

Refresh

Related Functions<<

Execute

Close

Across the top of the window are radio buttons to select read from PLC, write to PLC, verify with PLC, and delete PLC data.

There is a tab labeled Intelligent Function Module, which is used to download data directly to an intelligent module, such as writing to Flash ROM in QD75MH modules. It is not required for L Series intelligent module data.

The components of the program will be shown in the middle window. Each component can be checked to include it in the download. Checking only the components which need to be sent can improve communication speed. There are quick select buttons for parameters and programs only, select all objects, or unselect all objects above the window.

Notes

Some items in the transfer list will show a ‘Detail’ button next to them. The detail settings can make changes to the download information.

- For the PLC programs, the detail button allows modification of the number of write during run steps of memory which are reserved automatically in each program
- For the comments, the detail button allows the setting of ranges of comments which will be sent to the processor.
- For device memory, the detail button determines the ranges of the device data to be written to the CPU.

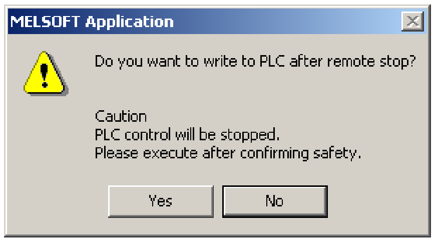
Across the bottom of the transfer window are quick buttons for other online PLC functions. These functions include:

- ➢ Start/Stop the PLC
- ➢ Set the PLC Clock
- ➢ PLC User Data (read and write memory card data)
- ➢ Write Title
- ➢ Format PLC memory (erase program memory)
- ➢ Clear PLC memory (erase data registers and/or file registers)
- ➢ Arrange PLC memory (defragment and arrange program memory)

These functions can be hidden by clicking on the ‘Related Functions’ button at the bottom of the transfer window.

Select to download the parameters and programs. Ensure that the target memory is ‘Program Memory’ and click ‘Execute’ to write the selected objects to the PLC.

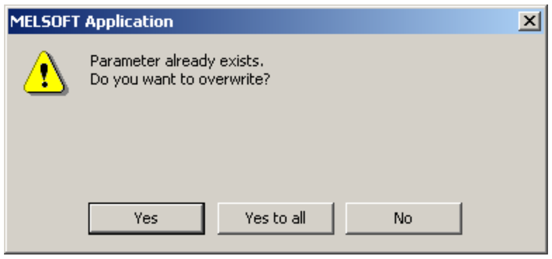
If the controller is running when the download starts, a warning will be presented asking if you wish to stop the processor. It is necessary to stop the PLC in order to perform a download. This will stop the controller and return the outputs to the off state. Be sure the controller is not performing a critical task or running operating equipment before clicking ‘Yes’.


MELSOFT Application

Do you want to write to PLC after remote stop?

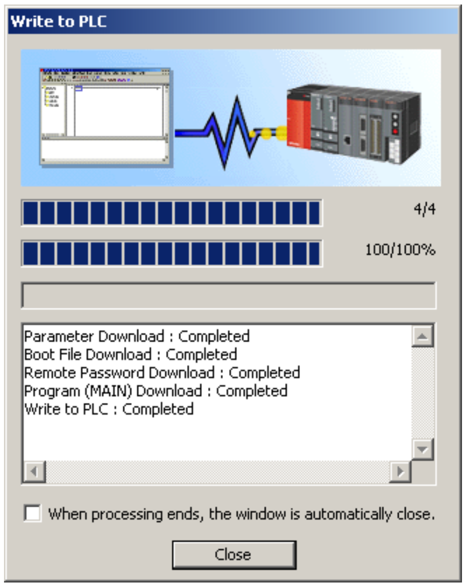
Caution
PLC control will be stopped.
Please execute after confirming safety.

Notes


MELSOFT Application

Parameter already exists.
Do you want to overwrite?

The software will also issue warnings if parameter or program data in the controller will be overwritten during the download process. Click 'Yes' to overwrite the existing information and be prompted for each additional duplicate item, or click 'Yes to all' to overwrite all data without further prompts. To keep data in the controller, click 'No'.


Write to PLC

4/4

100/100%

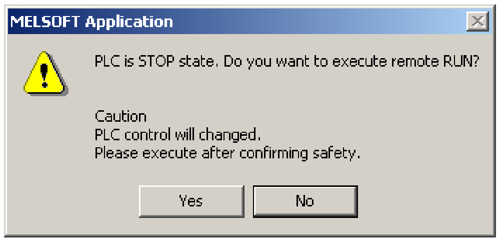
Parameter Download : Completed
Boot File Download : Completed
Remote Password Download : Completed
Program (MAIN) Download : Completed
Write to PLC : Completed

☐ When processing ends, the window is automatically close.

When the download completes, a screen will be displayed which shows the processes completed, in the order in which they were completed.

This screen can be suppressed in the future by checking the box at the bottom labeled 'When processing ends, the window is automatically closed.' Otherwise, click the 'Close' button at the bottom of the window to complete the download.

Once the download completed screen is closed, there may be a prompt to switch the PLC back into the RUN mode.


MELSOFT Application

PLC is STOP state. Do you want to execute remote RUN?

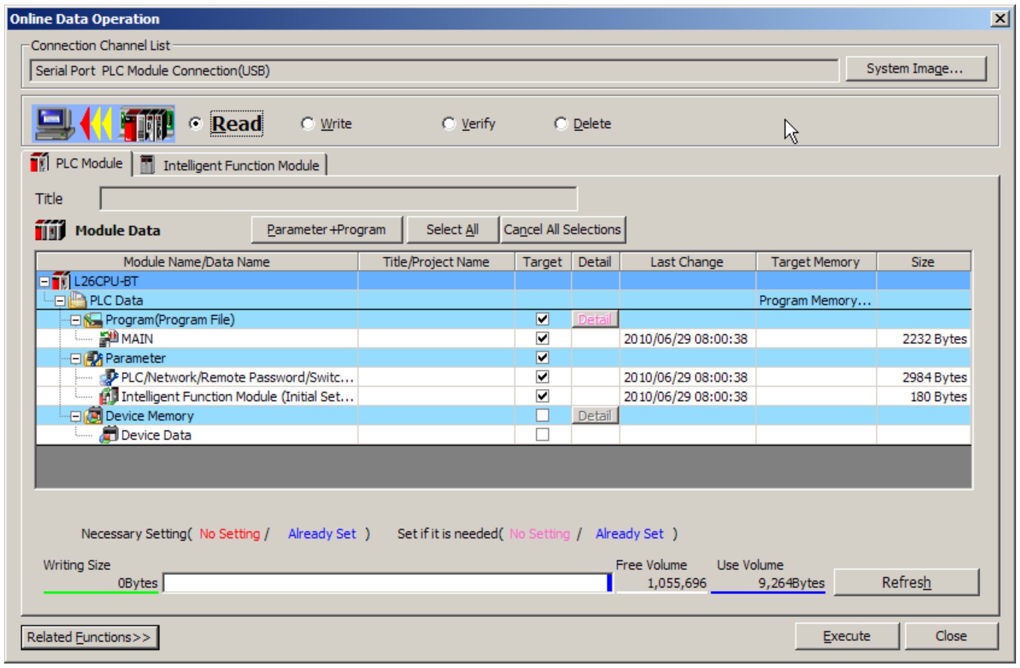
Caution
PLC control will changed.
Please execute after confirming safety.

If the controller was in the RUN mode when the download was started, GX Works2 will ask if it should restart the processor. If the controller was in the STOP mode, it will not ask this question. Click 'Yes' to restart the processor.

Notes

4.7 Reading from the PLC

To read the program already in the PLC, use the ‘Read from PLC’ option in the ‘Online’ menu. Any programs, parameters, comments, or data from the PLC memory can be uploaded from this screen.



Online Data Operation

Connection Channel List: Serial Port PLC Module Connection(USB)

System Image...

[✓ Read] Write Verify Delete

PLC Module | Intelligent Function Module

Title

Module Data — Parameter+Program | Select All | Cancel All Selections

Module Name/Data Name	Title/Project Name	Target	Detail	Last Change	Target Memory	Size
L26CPU-BT						
PLC Data					Program Memory...	
Program(Program File)		✓	Detail			
MAIN		✓		2010/06/29 08:00:38		2232 Bytes
Parameter		✓				
PLC/Network/Remote Password/Switc...		✓		2010/06/29 08:00:38		2984 Bytes
Intelligent Function Module (Initial Set...		✓		2010/06/29 08:00:38		180 Bytes
Device Memory		☐	Detail			
Device Data		☐				

Necessary Setting(No Setting / Already Set) Set if it is needed(No Setting / Already Set)

Writing Size 0Bytes

Free Volume 1,055,696 Use Volume 9,264Bytes

Refresh

Related Functions>> Execute Close

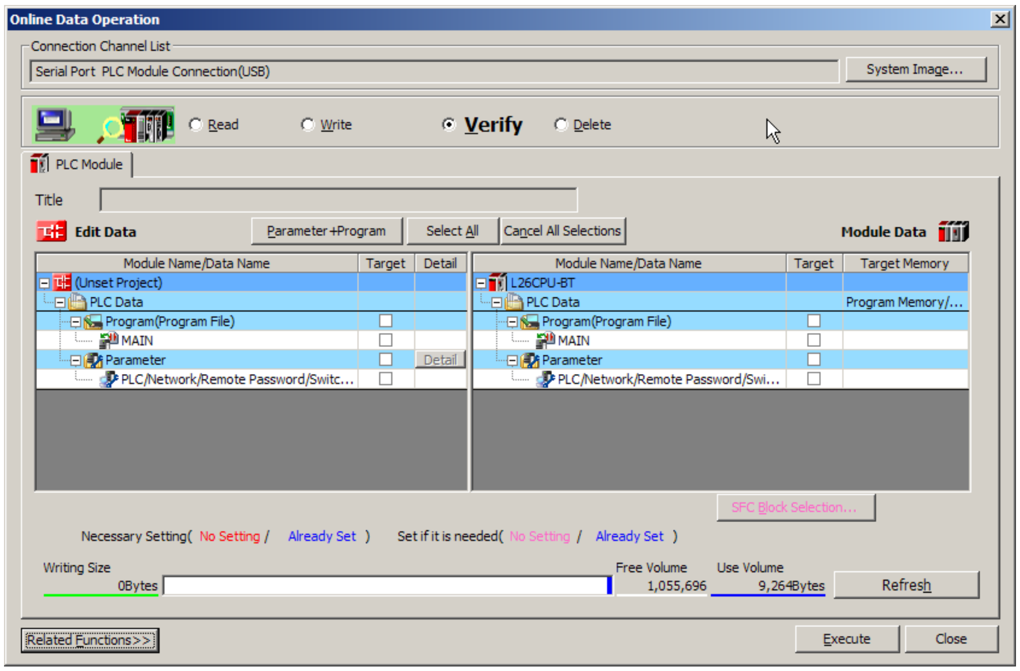
Only the objects which actually exist in the PLC will be shown, so if comments were not downloaded to the PLC, the Comments box will not appear in the ‘Read from PLC’ dialog box.

The ‘Device Data’ option will upload the values stored in the PLC’s data registers.

Notes

4.8 Verify with PLC

To compare the PLC program or parameters in the PLC with the one in the open GX Works2 project, use the ‘Verify with PLC’ option in the ‘Online’ menu.



Online Data Operation

Connection Channel List: Serial Port PLC Module Connection(USB)

Read / Write / [✓ Verify] / Delete

PLC Module

Title:

Module Name/Data Name	Target	Detail
(Unset Project)		
PLC Data		
Program(Program File)	☐	
MAIN	☐	
Parameter	☐	Detail
PLC/Network/Remote Password/Switc...	☐	

Module Name/Data Name	Target	Target Memory
L26CPU-BT		
PLC Data		Program Memory/...
Program(Program File)	☐	
MAIN	☐	
Parameter	☐	
PLC/Network/Remote Password/Swi...	☐	

Necessary Setting(No Setting / Already Set) Set if it is needed(No Setting / Already Set)

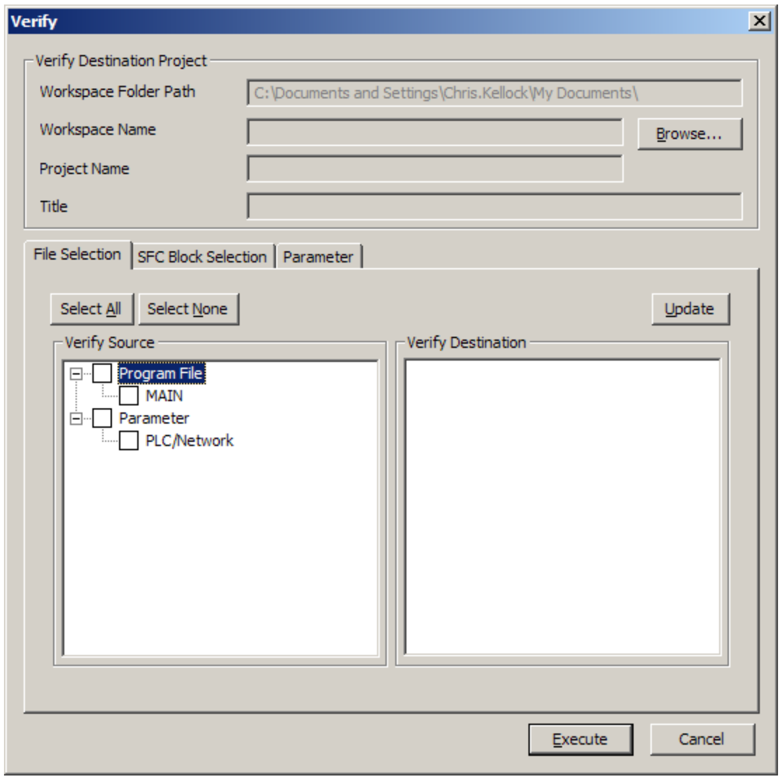
Writing Size 0Bytes

Free Volume 1,055,696 Use Volume 9,264Bytes

Check the items to verify in both columns, and then click ‘Execute’. A separate window will appear with the results of each verify selected.

Notes

There is a second verify option, used to compare 2 GX Works2 projects instead of one project and one PLC. This option is found in the 'Project' menu. The 'Verify' option will compare the open project with another one specified on the verify screen.



Workspace Folder Path: C:\Documents and Settings\Chris.Kellock\My Documents\
Workspace Name:
Project Name:
Title:

Notes

LESSON 5 – Online Operations

This lesson will cover the online monitoring and modification tools in GX Works2.

Lesson Objectives

At the conclusion of this lesson, you will be able to…

- Monitor the operation of the program in the CPU.
- Perform online edits to a running PLC program.
- Force bits and change register values.

There are numerous tools in GX Works2 for monitoring the status of a PLC. Some of these options are:

- Online monitoring of ladder logic
- Device / Buffer Memory batch monitor
- Watch windows

The monitoring tools are found in the 'Online' menu under the 'Monitor' submenu.

5.1 Ladder Logic Monitor

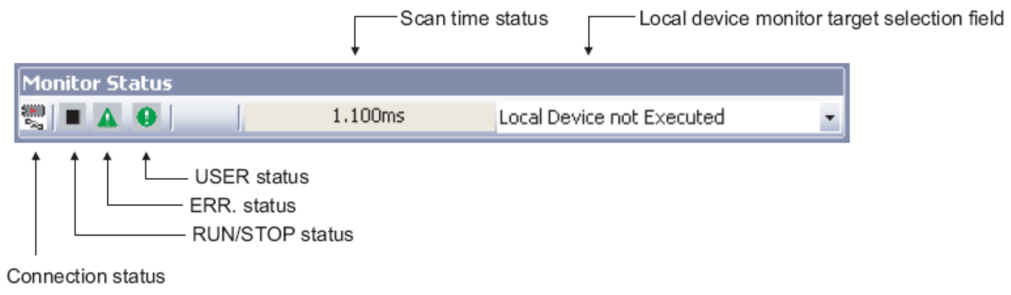
The status of ladder logic can be displayed while online with the PLC. This will highlight the inputs and outputs which are on, and display numeric data for data registers.

To start the ladder logic monitor, select the 'Online' menu; select 'Monitor' and then select either 'Start Monitoring' or 'Start Monitoring (All Windows)'. The shortcut key to start monitor in a single window is F3. Monitoring can be stopped for a window with Alt-F3.

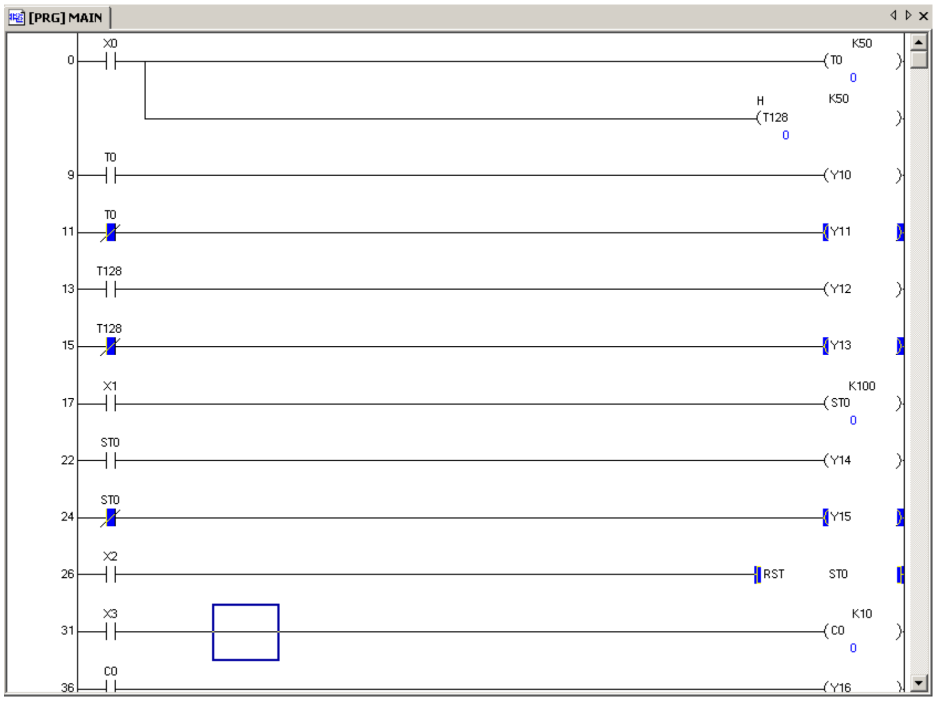
There are also buttons on the toolbars to start and stop monitoring. They are found next to the buttons for write to PLC and read from PLC on the Program Common toolbar.

Notes

When monitoring is active, the monitor toolbar will be shown. This toolbar shows the connection status, PLC mode, PLC and USER error status, maximum scan time since connected, and local device monitor status for L Series controllers.



The contacts or coils which are on will be highlighted. Notice the connection lines are NOT highlighted. Any command which uses numeric values will indicate the values directly below their elements in the ladder diagram.

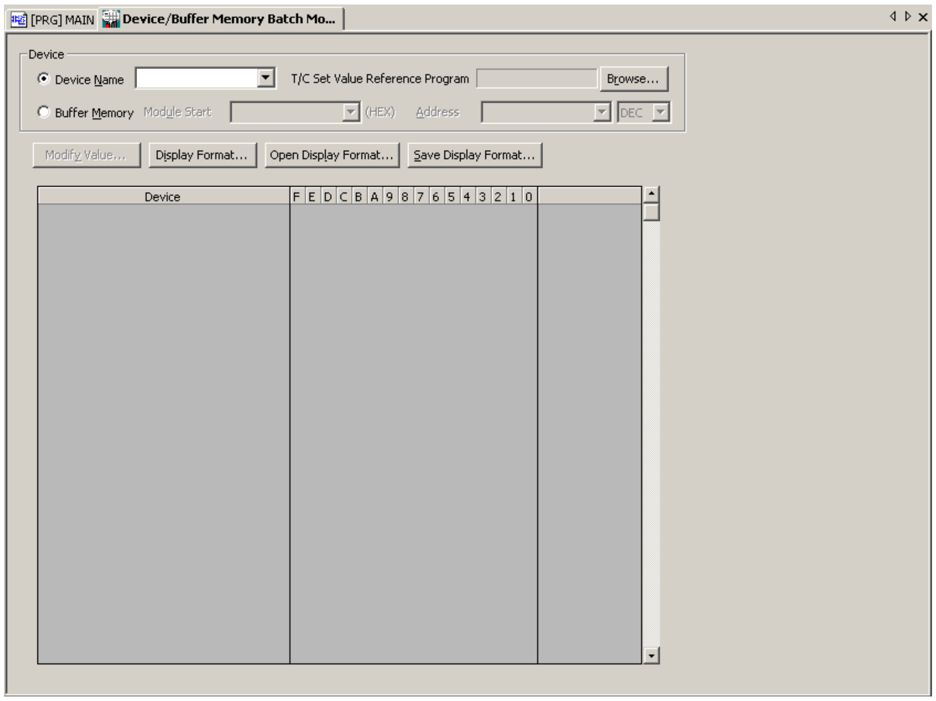


Notes

5.2 Device / Buffer Memory Batch Monitor

The device and buffer memory batch monitor is used to view a number of sequential addresses of PLC data or consecutive buffer memory locations in an intelligent module.

On the top of this window, one address and one display format can be configured, and when the ‘Start Monitor’ button in the toolbar or Enter is pressed, the screen is populated from that address forward. Individual display formats or non-sequential addresses cannot be monitored with this utility.



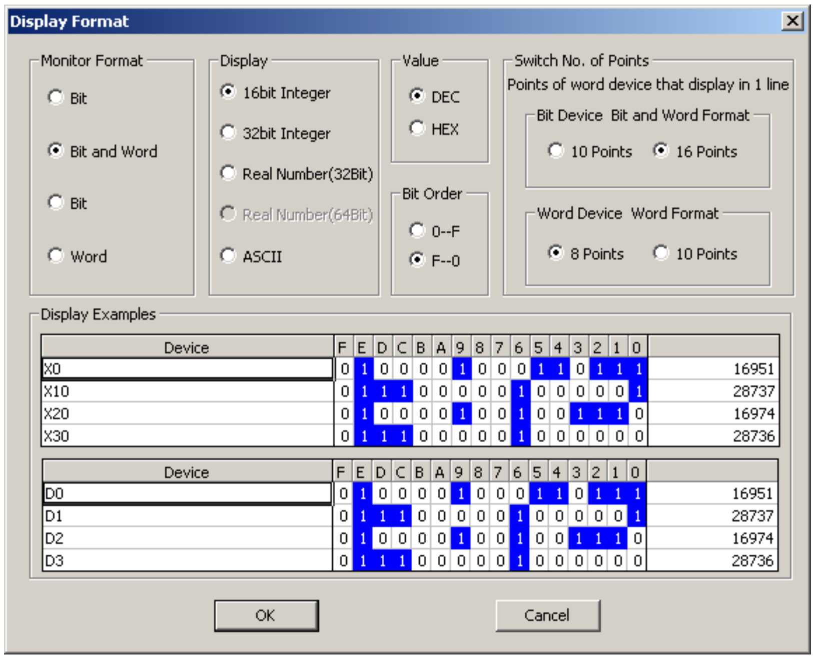
The lower half of the window will change based on the type of data from the address specified at the top and the display format specified.

Notes

Up to 64 device batch monitor screens can be created. Each will be assigned a number. If still open when the program is saved, the screens will be saved as part of the project, and will be available when the project is opened again.

Display Format...

The ‘Display Format’ button allows the configuration of the type of data and method of display to be shown. This will bring up the dialog box shown below. Any settings made on this window will apply to all addresses shown in the device batch monitor.



Monitor Format: ( ) Bit [✓ Bit and Word] ( ) Bit ( ) Word

Display: [✓ 16bit Integer] ( ) 32bit Integer ( ) Real Number(32Bit) ( ) Real Number(64Bit) ( ) ASCII

Value: [✓ DEC] ( ) HEX

Bit Order: ( ) 0--F [✓ F--0]

Switch No. of Points
Points of word device that display in 1 line

Bit Device Bit and Word Format: ( ) 10 Points [✓ 16 Points]

Word Device Word Format: [✓ 8 Points] ( ) 10 Points

Display Examples

Device	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0	
X0	0	1	0	0	0	0	1	0	0	0	1	1	0	1	1	1	16951
X10	0	1	1	1	0	0	0	0	0	1	0	0	0	0	0	1	28737
X20	0	1	0	0	0	0	1	0	0	1	0	0	1	1	1	0	16974
X30	0	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	28736

Device	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0	
D0	0	1	0	0	0	0	1	0	0	0	1	1	0	1	1	1	16951
D1	0	1	1	1	0	0	0	0	0	1	0	0	0	0	0	1	28737
D2	0	1	0	0	0	0	1	0	0	1	0	0	1	1	1	0	16974
D3	0	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	28736

Display formats can be saved and loaded from the Device / Buffer Memory Batch Monitor window. This allows a configuration to be designed and reused in other projects.

Open Display Format...

Utilizes the saved display formats.

Save Display Format...

Saves the current display format in the file.

Notes

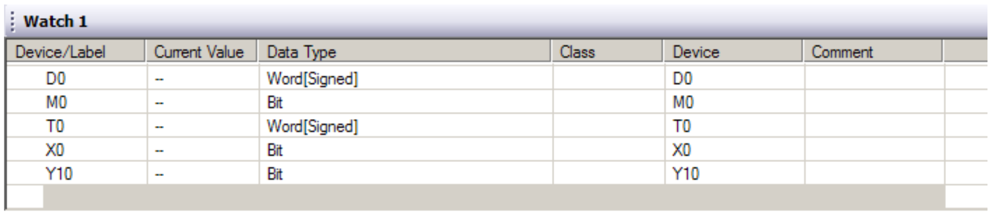
5.3 Watch Windows

In GX Works2, there are 4 data watch windows which can be turned on or off and docked on the screen. Watch windows can be turned on or off from the 'View' menu under 'Docking Window'. Just as the project tree, these windows can be pinned open or minimized to tabs.

Each of the specific devices to watch in a window is configured by the user. Data can be added to a watch window from within the watch window by typing a value into the first column of the table. Each item in the table can have its own data type configured, so monitoring of 16-bit, 32-bit, and floating point data on the same window is possible.

Another method for entering data into the watch window is to right click on and address on the ladder diagram and select 'Register Watch'. Entire structured ladder blocks can be registered by performing this step on the gray box at the left of the block. All addresses used in that ladder block will be added to the watch window. Multiple ladder blocks can also be selected. Items entered from this option go to the lowest number watch window which is open, whether pinned open or not.

Monitoring on each window can be started and stopped separately, and without the ladder monitor window being in monitor mode.


Watch 1

Device/Label	Current Value	Data Type	Class	Device	Comment
D0	--	Word[Signed]		D0	
M0	--	Bit		M0	
T0	--	Word[Signed]		T0	
X0	--	Bit		X0	
Y10	--	Bit		Y10	

Monitoring in the watch windows can be started with Shift-F3 or from the Start Watch command in the Online menu under Monitor. Shift-Alt-F3 will stop monitoring in a watch window.

Closing these windows will not delete the addresses which were registered.

Notes

5.4 Intelligent Module Monitor

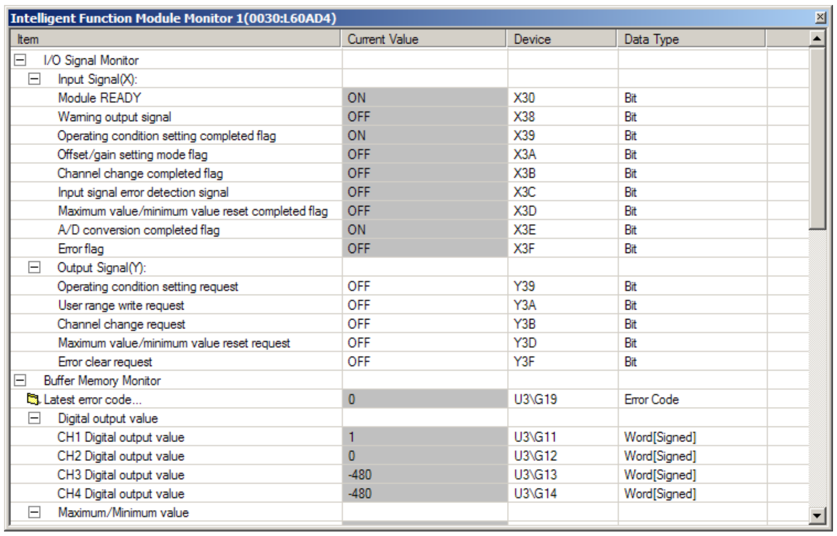
GX Works2 offers a docking window for monitoring of intelligent module data. Up to 10 of these windows can be used to monitor different intelligent modules.

To turn this window in, select the ‘View’ menu, then ‘Docking Window’, then ‘Intelligent Function Module Monitor’, and then select a monitor window number 1 through 10.

Modules can be registered into the monitor windows in one of 3 ways.

- Right click on the intelligent module in the project tree, then select ‘Register to Intelligent Function Module Monitor’
- Drag and drop a module from the project tree into the intelligent module monitor window
- Right click on the intelligent module monitor window and select ‘Register Module Information’

The monitor window below is for a L60AD4 analog module. Some values (shown in white) can be changed in the Current Value column.



Intelligent Function Module Monitor 1(0030:L60AD4)

Item	Current Value	Device	Data Type
⊟ I/O Signal Monitor			
⊟ Input Signal(X):			
Module READY	ON	X30	Bit
Warning output signal	OFF	X38	Bit
Operating condition setting completed flag	ON	X39	Bit
Offset/gain setting mode flag	OFF	X3A	Bit
Channel change completed flag	OFF	X3B	Bit
Input signal error detection signal	OFF	X3C	Bit
Maximum value/minimum value reset completed flag	OFF	X3D	Bit
A/D conversion completed flag	ON	X3E	Bit
Error flag	OFF	X3F	Bit
⊟ Output Signal(Y):			
Operating condition setting request	OFF	Y39	Bit
User range write request	OFF	Y3A	Bit
Channel change request	OFF	Y3B	Bit
Maximum value/minimum value reset request	OFF	Y3D	Bit
Error clear request	OFF	Y3F	Bit
⊟ Buffer Memory Monitor			
Latest error code...	0	U3\G19	Error Code
⊟ Digital output value			
CH1 Digital output value	1	U3\G11	Word[Signed]
CH2 Digital output value	0	U3\G12	Word[Signed]
CH3 Digital output value	-480	U3\G13	Word[Signed]
CH4 Digital output value	-480	U3\G14	Word[Signed]
⊟ Maximum/Minimum value			

Notes

5.5 Online Edits

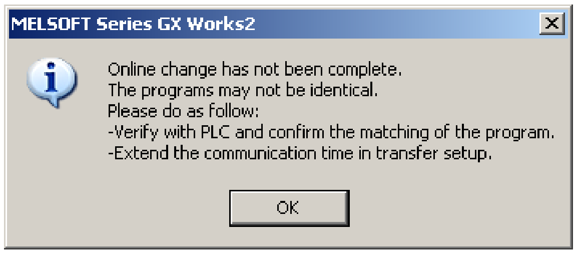
Program changes can be made online with the L Series controller while it is running. It is not necessary to stop the controller for basic program changes.

Each L Series program has a buffer of program steps for modification of a program while it is running. The default is 500 steps of program memory are reserved for each program, but this number is adjustable for each program individually. If this buffer fills, the only way to empty it is to stop the CPU and perform a complete download of the project.

Editing ladder online must be done while in the monitor mode. If F4 is pressed to convert ladder changes, the change is only made offline in the GX Works2 project. The PLC program is not updated at this time.

To make the edits to the PLC program while the PLC runs, a different command is used to convert the code. This command is found in the Compile menu, and is called Online Program Change. It is also accessible via the Shift-F4 shortcut key.

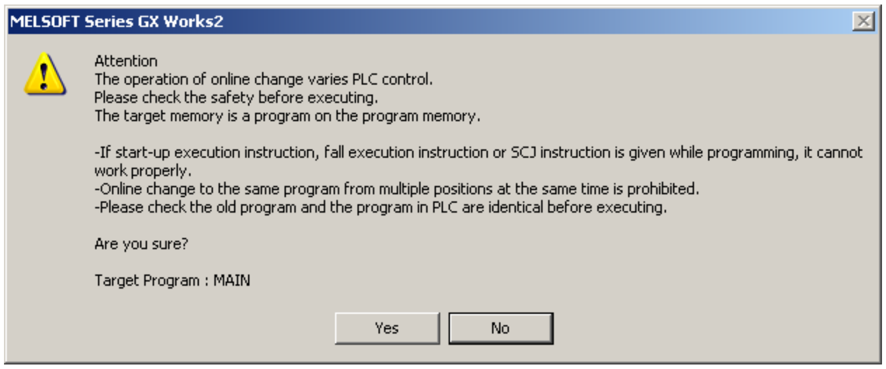
It is imperative that the PLC program and the open project match before an online edit is made. If the programs do not match, the edits will not be applied and this error message will be displayed.



Online change has not been complete.
The programs may not be identical.
Please do as follow:
-Verify with PLC and confirm the matching of the program.
-Extend the communication time in transfer setup.

Notes

When converting online changes, a dialog box appears indicating that the program currently executing in the CPU will be modified while it is running.


MELSOFT Series GX Works2

Attention
The operation of online change varies PLC control.
Please check the safety before executing.
The target memory is a program on the program memory.

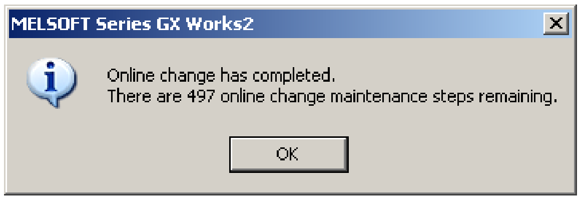
-If start-up execution instruction, fall execution instruction or SCJ instruction is given while programming, it cannot work properly.
-Online change to the same program from multiple positions at the same time is prohibited.
-Please check the old program and the program in PLC are identical before executing.

Are you sure?

Target Program : MAIN

By clicking 'Yes', the program in the CPU will be updated without stopping the controller. At the end of the next scan, program changes will be applied.

When the online edit completes, a dialog box will indicate success, and will display the number of write during run steps left in the buffer.


MELSOFT Series GX Works2

Online change has completed.
There are 497 online change maintenance steps remaining.

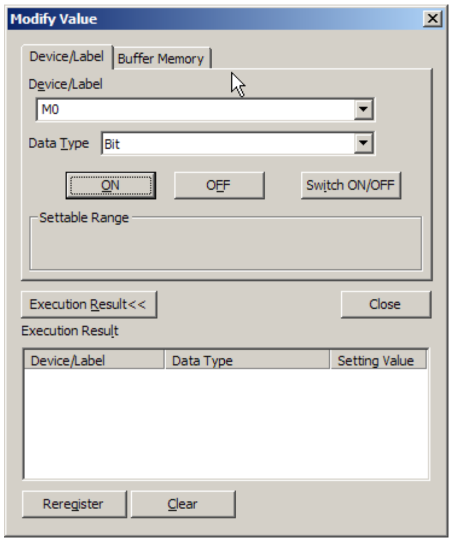
Notes

5.6 Modify Value

The modify value function is used to modify the value of a bit or word address in the PLC. It should not be mistaken for a method of forcing inputs or outputs.

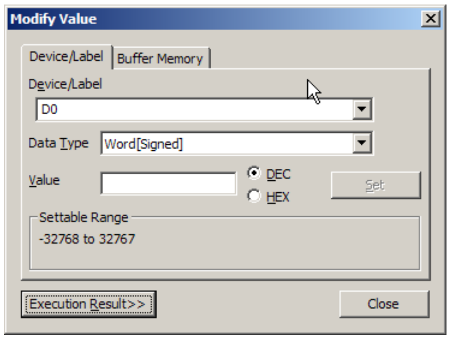
Modify value can be used by:

- Right click on an object and select Debug, Modify Value
- Debug menu, Modify Value
- Holding the shift key and double clicking on a contact or coil
- Shift-Enter on a contact or coil



Modify value can be used to turn a bit on or off. It is considered a momentary override of an address. There is no warning or visible indication that an address was modified. A history at the bottom of the window shows previously used addresses, and can be hidden if desired.

The PLC inputs take priority, so modifying an input will result in a momentary signal. At the beginning of the next scan, the PLC's inputs are read, and overwrite any modified input addresses. The programmed output status also takes priority, so a modified output will revert to its programmed status after one scan. When the program scan finishes, all outputs are updated, which also overwrites any modified output addresses used in the program.



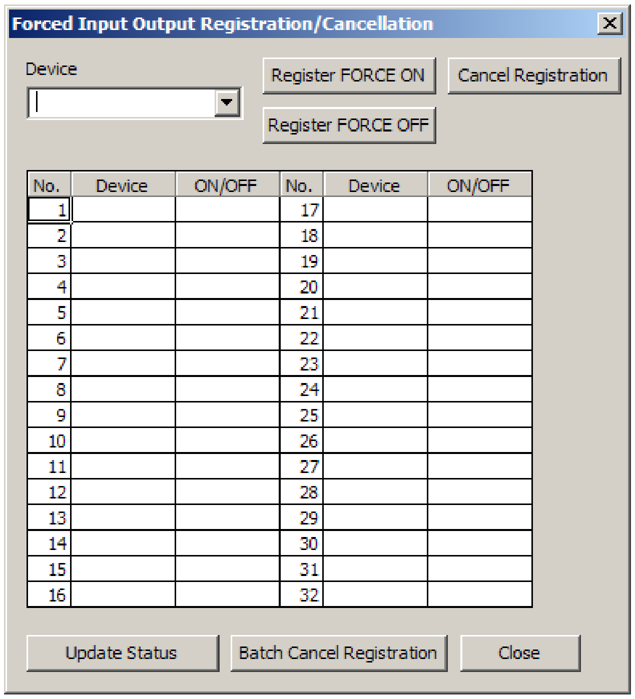
Modify value can also write a number to a data register, or write a number to a buffer memory address in an intelligent module. It can also be accessed from the Device / Buffer Memory batch monitor screen.

Notes

5.7 Forced I/O Registration

Forced I/O Registration is used only for X and Y addresses, and eliminates the limitations mentioned for device test. In this mode, an address can be set forced on or forced off, and it overrides the physical inputs and programmed outputs.

Forced I/O Registration/Cancellation can be found in the Debug menu, or in the Debug menu on the right click menu.



No.	Device	ON/OFF	No.	Device	ON/OFF
1			17		
2			18		
3			19		
4			20		
5			21		
6			22		
7			23		
8			24		
9			25		
10			26		
11			27		
12			28		
13			29		
14			30		
15			31		
16			32		

Forced I/O registration only works in the L Series and Q Series CPUs (excluding basic model).

Only X and Y addresses can be registered. Once registered, that address is not affected by the inputs or programmed outputs in the programs.

To add an address to the list, enter the address in the Device window, and click ‘Set forced ON’ or ‘Set forced OFF’ depending on the desired mode.

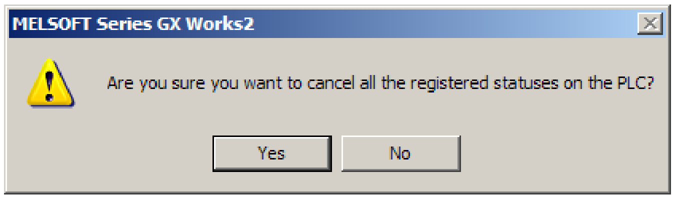
To remove an address which is forced, enter the address in the Device window and click ‘Cancel it’. All forces can be cancelled at once with the ‘Clear all’ button at the bottom of the screen.

The ‘Update Status’ button will read the forced I/O table from the connected CPU.

While forced I/O registration is active, the MODE light on the front of the processor will be flashing. When the registration is cancelled, the LED will return to the solid on state. There is also an icon in the Online toolbar in GX Works2 to indicate the forced I/O registration activity.

Notes

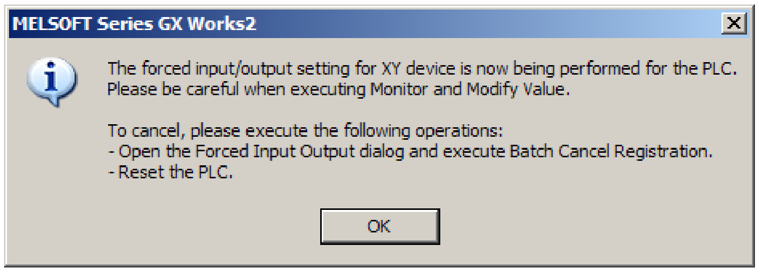
When exiting this window, a message will prompt asking to clear all forced I/O status. If 'Yes' is selected, all forces will be erased, the same as clicking 'Clear all' in the window.


MELSOFT Series GX Works2

Are you sure you want to cancel all the registered statuses on the PLC?

If 'No' is selected, the forces remain in effect.

When connecting GX Works2 or GX Developer to a CPU which has active forces, a message is displayed indicating that there are active forces in the CPU. To see what addresses are forced or modify the forces, open the force I/O registration window.


MELSOFT Series GX Works2

The forced input/output setting for XY device is now being performed for the PLC.
Please be careful when executing Monitor and Modify Value.

To cancel, please execute the following operations:
- Open the Forced Input Output dialog and execute Batch Cancel Registration.
- Reset the PLC.

Forced I/O Registration is cancelled when a CPU is powered off, or by resetting the PLC.

Notes

Notes

LESSON 6 – GX Works2 Utilities

This lesson introduces some of the editing and troubleshooting features of the GX Works2 software.

Lesson Objectives

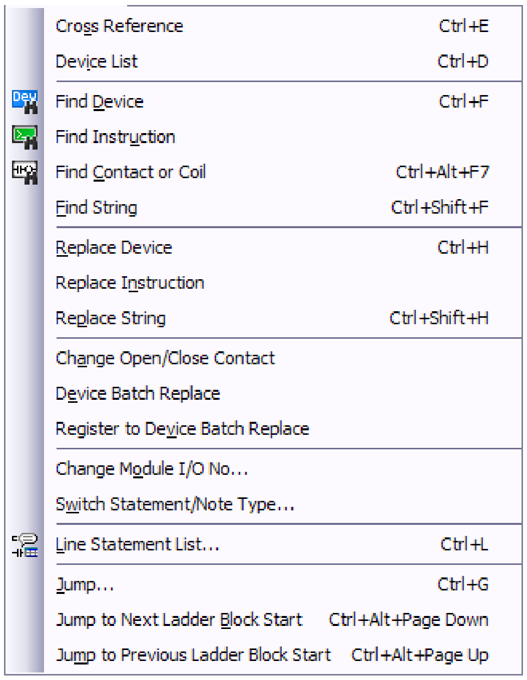
At the conclusion of this lesson, you will be able to…

- Use the options in the find/replace menu.
- Run and interpret the PLC Diagnostics screens.
- Utilize the System Monitor tool to troubleshoot hardware.
- Execute a sampling data trace.

6.1 Find/Replace Menu

GX Works2 provides several different methods for locating devices or instructions. The Find/Replace menu has a variety of tools for searching or modifying programs. The find options will search a program for an occurrence of an address or instruction. The replace options will allow the replacing of addresses, functions, open/closed contacts, and other options. The Cross Reference will show all occurrences of an address in a program, and the Device List shows all used addresses in the program.

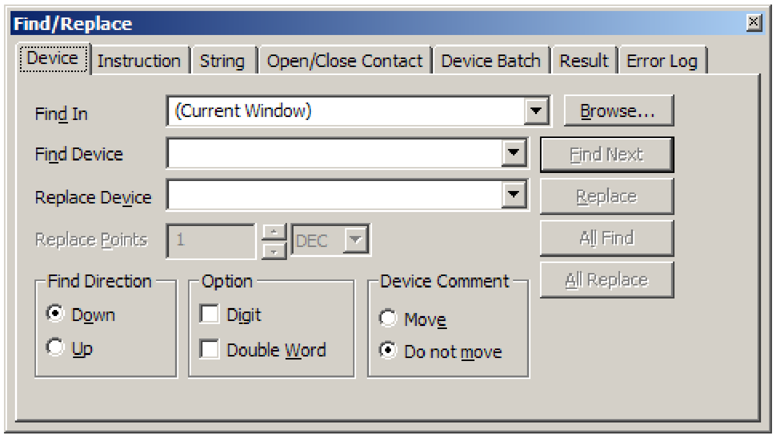
Shown to the right is the list of options on the Find/Replace menu.


Cross Reference Ctrl+E
Device List Ctrl+D
Find Device Ctrl+F
Find Instruction
Find Contact or Coil Ctrl+Alt+F7
Find String Ctrl+Shift+F
Replace Device Ctrl+H
Replace Instruction
Replace String Ctrl+Shift+H
Change Open/Close Contact
Device Batch Replace
Register to Device Batch Replace
Change Module I/O No…
Switch Statement/Note Type…
Line Statement List… Ctrl+L
Jump… Ctrl+G
Jump to Next Ladder Block Start Ctrl+Alt+Page Down
Jump to Previous Ladder Block Start Ctrl+Alt+Page Up

Notes

Find Device is used to find a device address regardless of the instruction. It will search the entire program for a particular address. There are a couple of options on this screen. At the top, you can limit the search area.

This same dialog can be used to replace devices.


Find/Replace

Device | Instruction | String | Open/Close Contact | Device Batch | Result | Error Log

Find In: (Current Window)

Find Device:

Replace Device:

Replace Points: 1 DEC

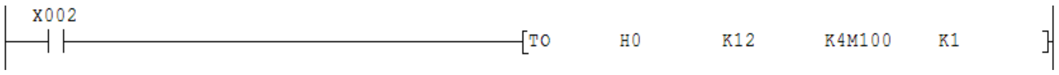
Find Direction: [✓ Down] Up

Option: ☐ Digit ☐ Double Word

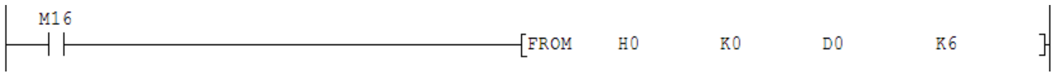
Device Comment: Move [✓ Do not move]

The option setting includes other possible matches in the results.

Digit allows the find to look for a bit address in a word of bits. A digit search of the following code will find that M110 has been used as part of this TO instruction.



Double Word expands the search to include a word that is used by an instruction using multiple words. A double word search of the following code will find that D5 has been used as part of this FROM instruction, since it writes to 6 addresses starting at D0.



Notes

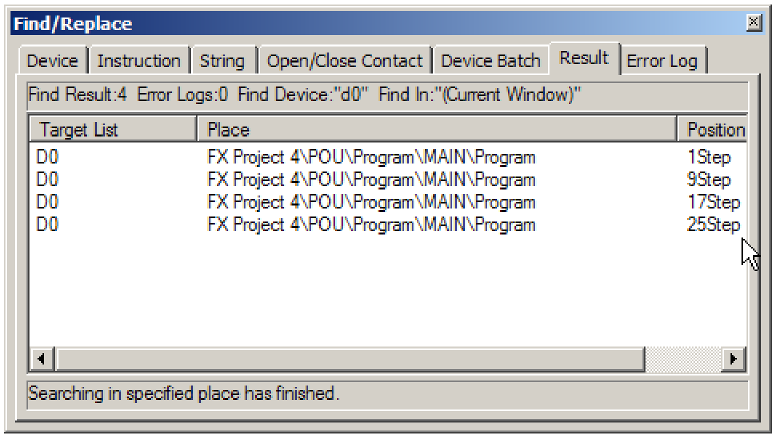
Replace Device allows search and replace of a device or a range of devices within the program. A single address can be specified for the earlier device and for the new device. The number of substitute points sets how many consecutive addresses to move. The radio buttons for move comments allows comments assigned to an address to be moved with the address. The find direction options allow the search and replace to be restricted to a certain portion of the program.

Replace of addresses is not done as an online edit. The program must be downloaded after a replace is complete.

The Find Next button will find the next location of the searched data. There is no find previous button, but by switching the direction to up and picking Find Next, the same function can be achieved.

The All Find or All Replace buttons will find or replace all locations of the desired text.

The second tab from the right is called Result. In this window, all of the found or replaced data can be viewed. Items in the results window can be double clicked for easy program navigation.



Find Result:4 Error Logs:0 Find Device:"d0" Find In:"(Current Window)"

Target List	Place	Position
D0	FX Project 4\POU\Program\MAIN\Program	1Step
D0	FX Project 4\POU\Program\MAIN\Program	9Step
D0	FX Project 4\POU\Program\MAIN\Program	17Step
D0	FX Project 4\POU\Program\MAIN\Program	25Step

Searching in specified place has finished.

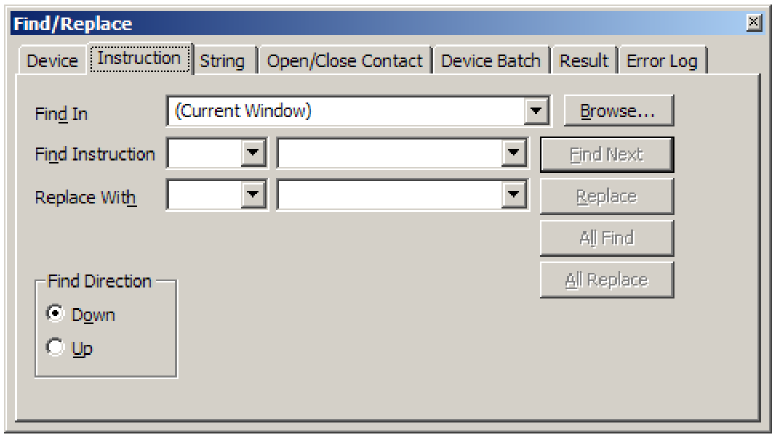
The last tab is called Error Log, and it is used with the replace window to indicate any errors which occurred in the replace process.

Notes

Find/Replace can also be used as a docking window. To turn on this window, open the View menu, select Docking Window, and then select Find/Replace Window.

Find Instruction is used to locate all instances of a particular instruction. The first selection box selects the ladder symbol and the second selection box selects the command to search for. The find direction section works as previously discussed.

This same dialog can be used to replace instructions.

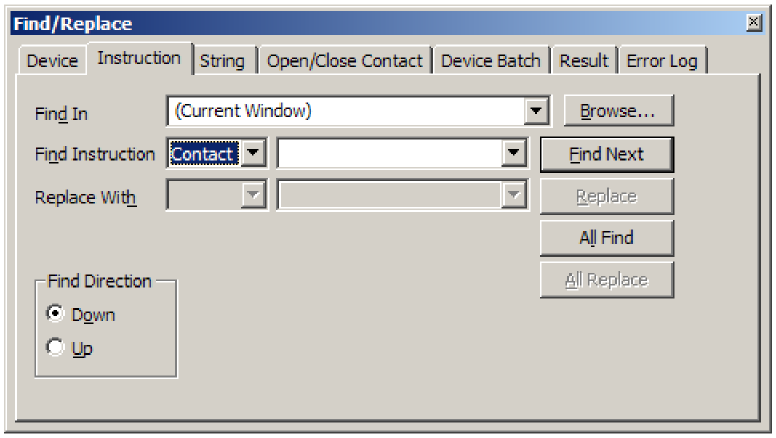


Replace Instruction will change one instruction type to another. The earlier instruction is the type of instruction to replace, and the new instruction is the instruction to substitute into the program. This can be useful for example to replace all occurrences of INC with INCP in a program.

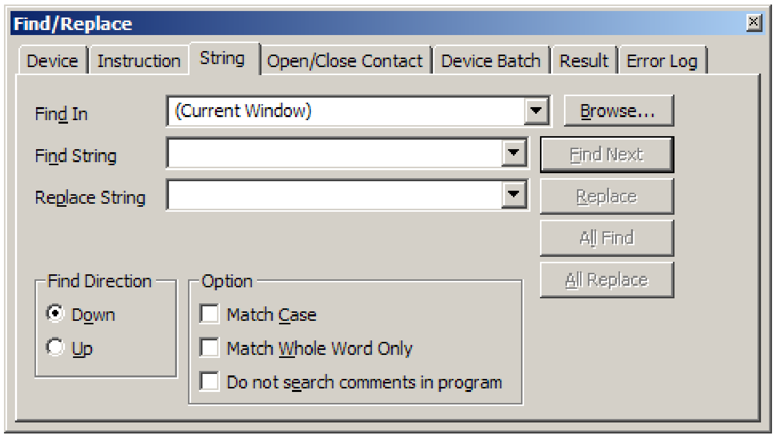
- ➢ Replace of instructions is not done as an online edit. A download of the program must be done after a replace is complete.

Notes

Find Contact or Coil uses the same dialog and will search the program for an address used as either a contact or a coil. In the first window, select contact or coil. Then in the right window, enter an address. This can be handy when an address has been used as a contact numerous times in a program to find the one location it was used as a coil.



The **Find String** option searches a target for a text string. This string can be in addresses, comments, statements, notes or labels.



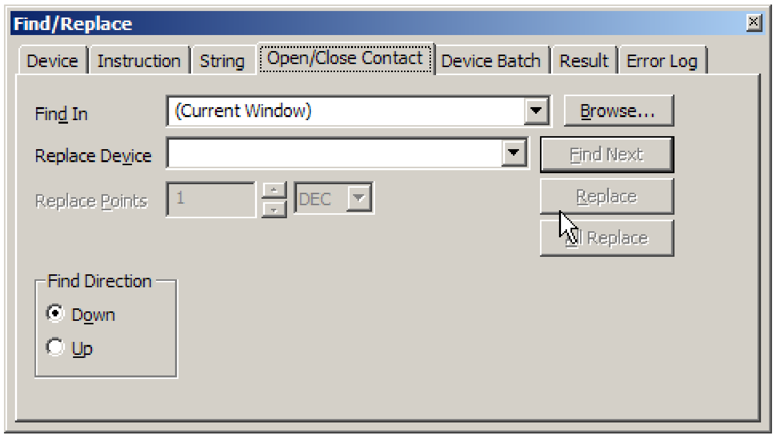
A setting at the top of the window allows the search range to be limited.

Notes

The same dialog is used with replace, simply by entering data in the Replace String field. It can also be opened from the **Replace String** option in the Find/Replace menu.

Settings on the bottom of the window allow adjustment of the direction of search, whether the search is case sensitive, and whether to only match whole words.

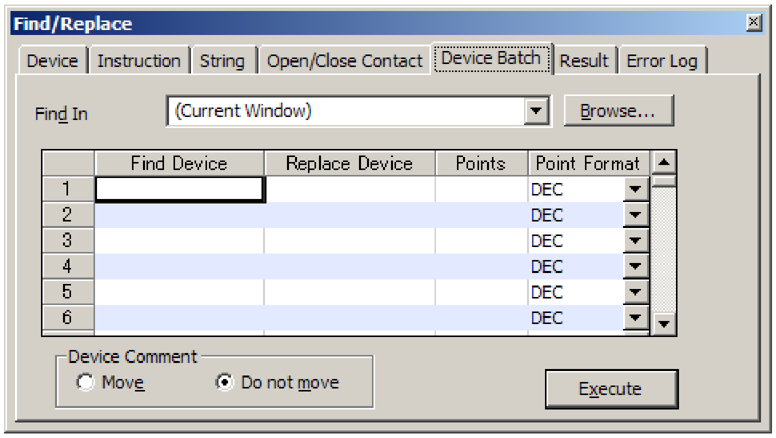
Change Open/Close Contact will change all occurrences of an open contact to a closed contact and change closed contacts to open. A single command will invert the state of all occurrences of the address. This is useful if the type of input provided to a machine changes or does not match the code, such as a program which is expecting a normally closed stop signal and the machine is wired with a normally open switch.



Find In: (Current Window)
Replace Device:
Replace Points: 1 DEC
Find Direction: [✓ Down] Up

Notes

Device Batch Replace takes the Replace Device option a step farther. It allows the programmer to replace multiple ranges of devices in a single command. Groups of timers, counters, inputs, and data registers can all be acted on at once. Each line in the chart is a different range of addresses. Addresses cannot overlap in any rows.



Find In: (Current Window)

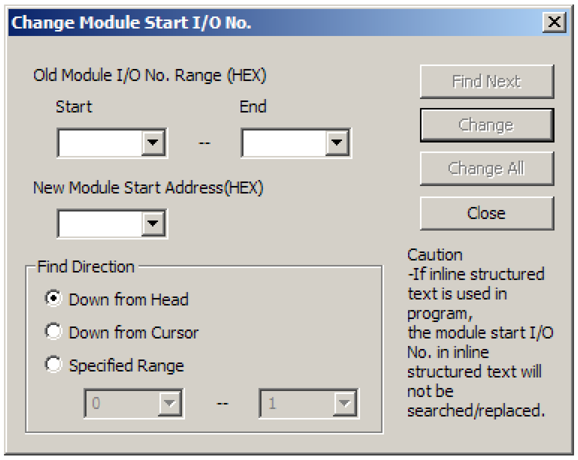
	Find Device	Replace Device	Points	Point Format
1				DEC
2				DEC
3				DEC
4				DEC
5				DEC
6				DEC

Device Comment: Move [✓ Do not move]

Notes

Change Module I/O Number allows the programmer to change the head address of a special function module. This code will search a program and change all occurrences of the head address, such as TO or FROM instructions. By specifying a start and end as different values, a range of intelligent modules can be moved at once. Enter the start and end SFM numbers, and enter a single new module number. The software will move the old addresses to a range of equal size starting at the new module address.

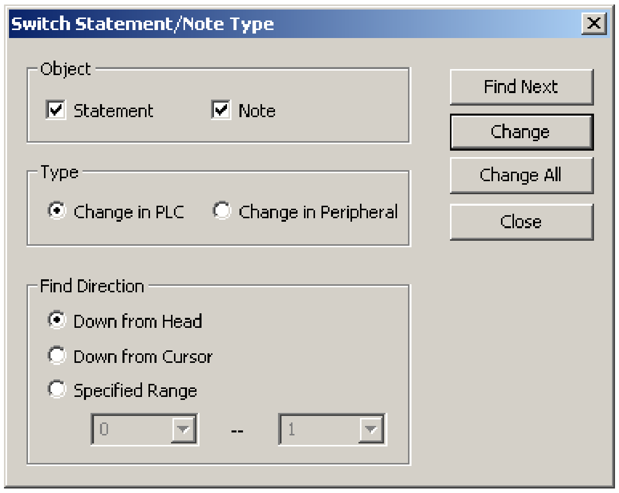
This tool does not change U\G addresses. It is only supported in the basic ladder language.



Caution
-If inline structured text is used in program,
the module start I/O No. in inline structured text will not be searched/replaced.

Notes

Switch Statement/Note Type is used to change embedded status of statements and notes. Q Series and L Series offer the option to embed statements and notes into the program when it is downloaded to the CPU. 'Change in Peripheral' means the statements or notes only exist in GX Works2. 'Change in PLC' means they are part of the PLC code and are sent to the PLC with the download of the program code.


Switch Statement/Note Type

Object: [✓ Statement] [✓ Note]

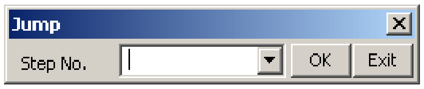
Type: [✓ Change in PLC] Change in Peripheral

Find Direction: [✓ Down from Head] Down from Cursor Specified Range 0 -- 1

The replace can be limited to a certain section of a program or the entire program, only statements, only notes, or both statements and notes.

If the CPU is low on memory, this tool can be used to remove the documentation from the CPU memory. All documentation can still be viewed with a copy of the GX Works2 project, but will not be in the CPU to be uploaded.

Jump will jump to a step number in the program. This can be useful with the error information previously discussed to search out the location of a program error in the program.

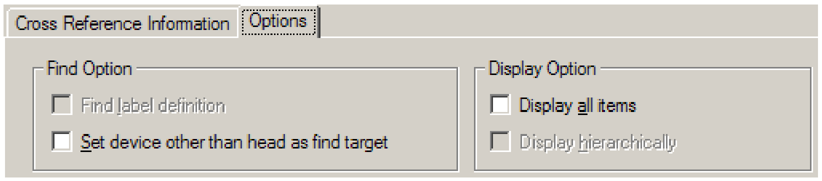


Notes

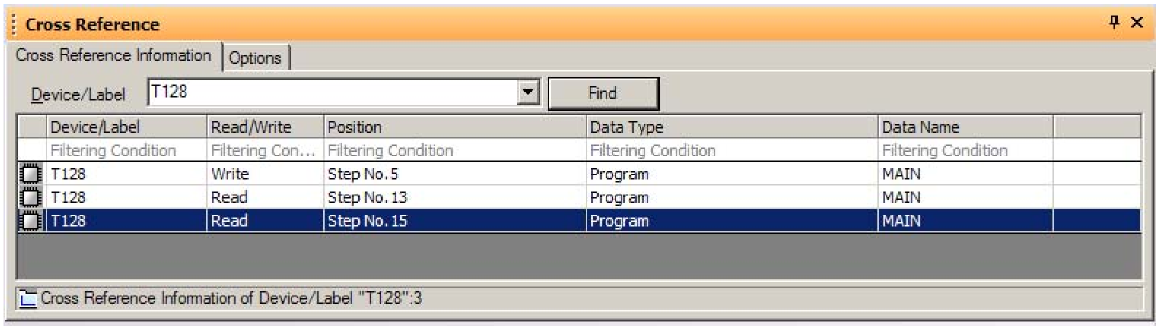
6.2 Cross Reference

A cross reference is a list showing all of the times an address was used in the program and in what type of instruction. Cross reference can search the open program, or all programs in the open project. Cross reference information is displayed in a docking window. Cross reference can be used on a single address or all addresses at once.

To configure options, use the Options tab. Settings here include an option similar to the digit/multiple word (as discussed with Find Device). The 'Set device other than head as find target' option replaces the digit and multiple word option with a single check box.



To perform a cross reference and see the results, enter the address and click the Find on the Cross Reference Information tab.



Device/Label	Read/Write	Position	Data Type	Data Name
Filtering Condition	Filtering Con...	Filtering Condition	Filtering Condition	Filtering Condition
T128	Write	Step No.5	Program	MAIN
T128	Read	Step No.13	Program	MAIN
T128	Read	Step No.15	Program	MAIN

Cross Reference Information of Device/Label "T128":3

Double clicking on an item in the cross reference window will jump the program display to that location.

Cross reference can also be accessed from right-clicking on any address in the ladder diagram.

Notes

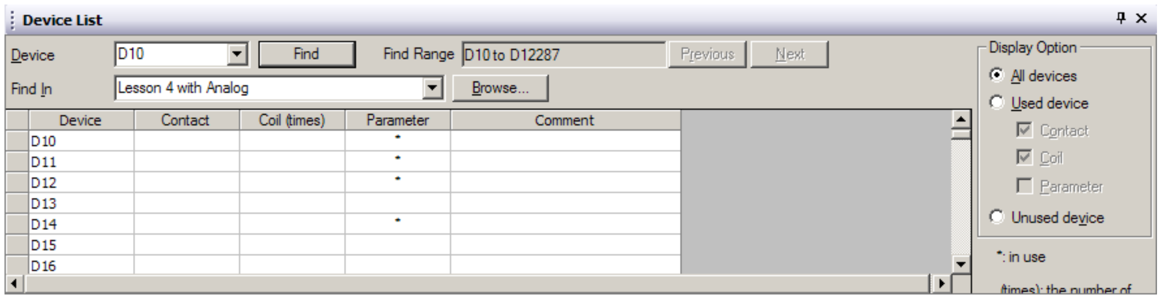
6.3 Device List

The Device List will show a list of all of the devices used in the program. The list can be configured to show a single program or all programs in the CPU.

The locations to search can be defined in the ‘Find in’ box. This location can be a program, the program pool, PLC parameters, intelligent function utility, or the entire project.

An address is entered in the device box, and after pressing the Find button, the list starting at that address is shown in the window below.

There are columns which will show an asterisk (*) if that address has been used as an input, output, or in parameter settings. Inputs include source data for applied instructions and outputs include destinations of applied instructions. If comments are registered, they will show in the Comment column



Device: D10
Find Range: D10 to D12287
Find In: Lesson 4 with Analog

Device	Contact	Coil (times)	Parameter	Comment
D10			*	
D11			*	
D12			*	
D13				
D14			*	
D15				
D16				

Display Option
- [✓ All devices]
- Used device
  - [✓ Contact]
  - [✓ Coil]
  - Parameter
- Unused device

*: in use

(times): the number of

The Device List is also a docking window.

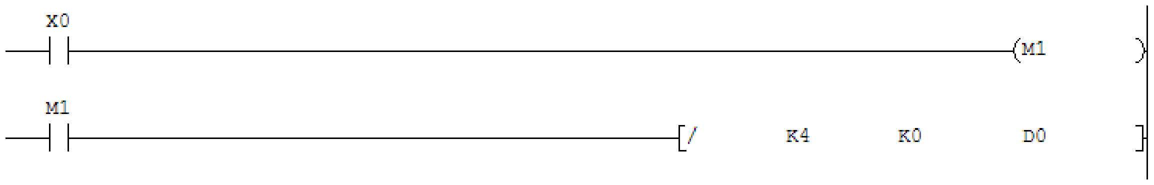
Device List is also accessible from the right click menu in a ladder diagram or by the Ctrl-D shortcut key. The window will be populated with addresses starting at the address of the item clicked on, and automatically executed.

Notes

6.4 PLC Diagnostics

GX Works2 has built in diagnostics capability to assist the user with troubleshooting a PLC error. The PLC Diagnostics screen provides an easy to understand interface for data which is stored inside the CPU in the special relays and registers.

Please create a new simple project and enter the rung shown below:



Take a moment to examine the above logic. What happens when X0 is turned on? The first value (4) is divided by the second value (0) and the result is placed into D0.

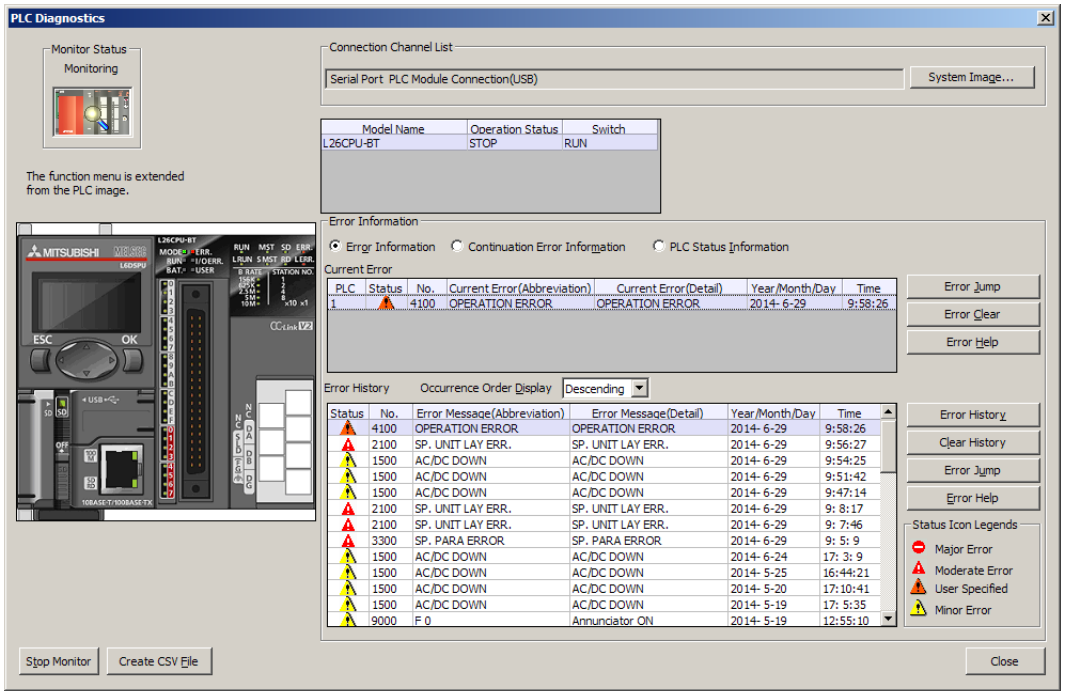
Dividing by zero is an illegal operation, since the result is an infinite number. Trigger X0 and watch what happens. The error light on the CPU comes on. The RUN light also goes out, which indicates the PLC is stopped.

To troubleshoot do the following steps:

- Click on the Diagnostics pull down menu.
- Click on PLC Diagnostics.

Notes

A screen like the one below appears:



Serial Port PLC Module Connection(USB)

Model Name	Operation Status	Switch
L26CPU-BT	STOP	RUN

Current Error

PLC	Status	No.	Current Error(Abbreviation)	Current Error(Detail)	Year/Month/Day	Time
1	⚠	4100	OPERATION ERROR	OPERATION ERROR	2014- 6-29	9:58:26

Error History Occurrence Order Display: Descending

Status	No.	Error Message(Abbreviation)	Error Message(Detail)	Year/Month/Day	Time
⚠	4100	OPERATION ERROR	OPERATION ERROR	2014- 6-29	9:58:26
⚠	2100	SP. UNIT LAY ERR.	SP. UNIT LAY ERR.	2014- 6-29	9:56:27
⚠	1500	AC/DC DOWN	AC/DC DOWN	2014- 6-29	9:54:25
⚠	1500	AC/DC DOWN	AC/DC DOWN	2014- 6-29	9:51:42
⚠	1500	AC/DC DOWN	AC/DC DOWN	2014- 6-29	9:47:14
⚠	2100	SP. UNIT LAY ERR.	SP. UNIT LAY ERR.	2014- 6-29	9: 8:17
⚠	2100	SP. UNIT LAY ERR.	SP. UNIT LAY ERR.	2014- 6-29	9: 7:46
⚠	3300	SP. PARA ERROR	SP. PARA ERROR	2014- 6-29	9: 5: 9
⚠	1500	AC/DC DOWN	AC/DC DOWN	2014- 6-24	17: 3: 9
⚠	1500	AC/DC DOWN	AC/DC DOWN	2014- 5-25	16:44:21
⚠	1500	AC/DC DOWN	AC/DC DOWN	2014- 5-20	17:10:41
⚠	1500	AC/DC DOWN	AC/DC DOWN	2014- 5-19	17: 5:35
⚠	9000	F 0	Annunciator ON	2014- 5-19	12:55:10

At the top of this window are the current operating mode and the run/stop switch setting on the CPU.

On the left is a picture of the CPU showing the active LEDs. Moving the mouse over various sections of the processor shows different popups, showing memory card options, remote operation, and diagnostic options.

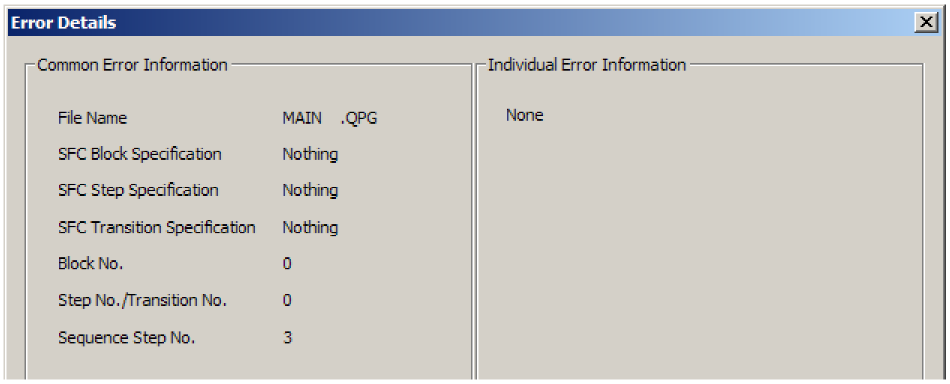
In the center of the screen is the active alarms list. This shows the currently active alarm number in the CPU.

Below this is the PLC's alarm history. This is the alarm history for the PLC selected in the top window. It can be exported to a CSV file with the button at the bottom left of the window.

Notes

Divide by zero causes an error number 4100. So 4100 is the error number that appears in register SD0.

Double click on an active error to see more detailed error information. This is the information found in SD1 through SD26.


Error Details

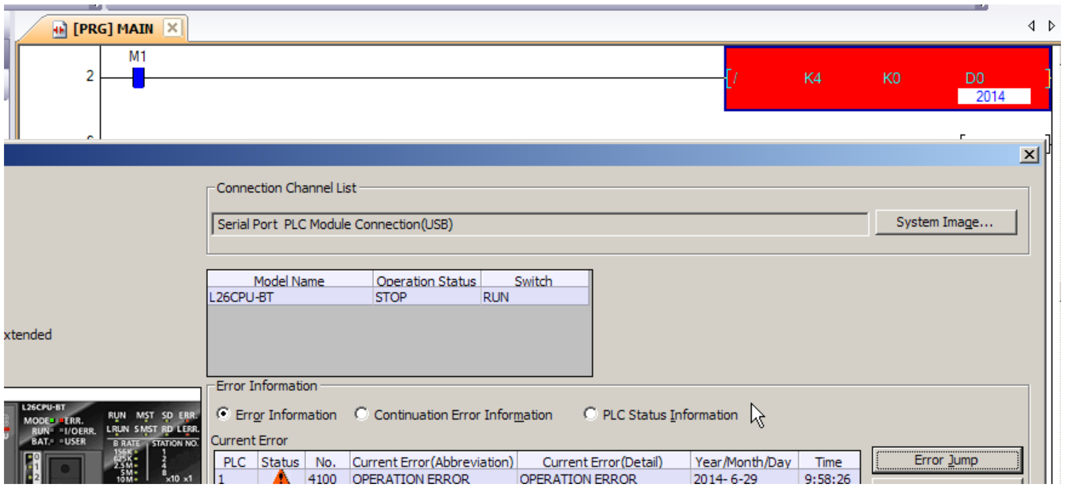
Common Error Information

File Name	MAIN .QPG
SFC Block Specification	Nothing
SFC Step Specification	Nothing
SFC Transition Specification	Nothing
Block No.	0
Step No./Transition No.	0
Sequence Step No.	3

Individual Error Information

None

Click on the Error Jump button in PLC Diagnostics to move the ladder diagram to the instruction responsible for the error. The divide operation is highlighted in red. This is the instruction (step 3) that is faulting out. While we could guess this in a small program such as this, it is very handy for large programs.



Connection Channel List

Serial Port PLC Module Connection(USB)

Model Name	Operation Status	Switch
L26CPU-BT	STOP	RUN

Error Information

[✓ Error Information] Continuation Error Information PLC Status Information

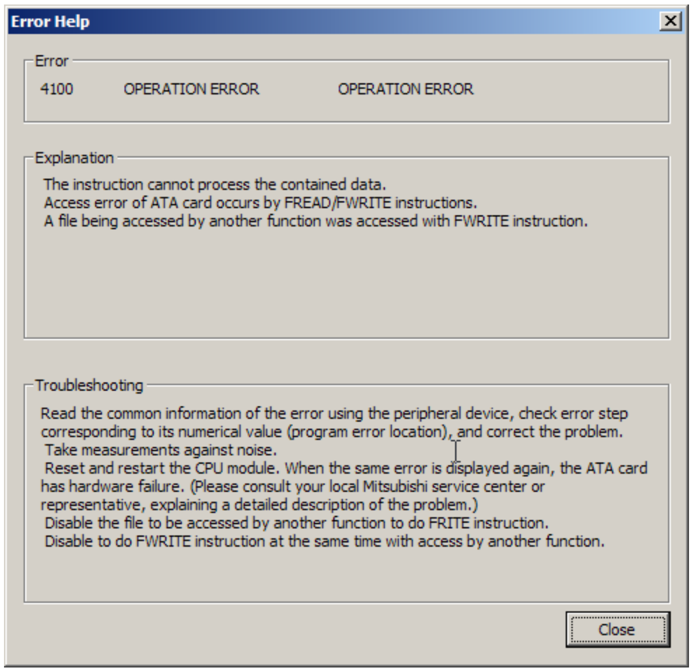
Current Error

PLC	Status	No.	Current Error(Abbreviation)	Current Error(Detail)	Year/Month/Day	Time
1	⚠	4100	OPERATION ERROR	OPERATION ERROR	2014- 6-29	9:58:26

Notes

Click on the Help button in PLC Diagnostics for more information on the error message. A help screen for the selected error should appear. The following screen should appear for the 4100 error.

This screen explains possible causes of the error state and possible solutions. While divide by 0 isn't explicitly stated here, it does indicate that there are references to data which cannot be processed by an instruction.



4100 OPERATION ERROR OPERATION ERROR

The instruction cannot process the contained data.
Access error of ATA card occurs by FREAD/FWRITE instructions.
A file being accessed by another function was accessed with FWRITE instruction.

Read the common information of the error using the peripheral device, check error step corresponding to its numerical value (program error location), and correct the problem.
Take measurements against noise.
Reset and restart the CPU module. When the same error is displayed again, the ATA card has hardware failure. (Please consult your local Mitsubishi service center or representative, explaining a detailed description of the problem.)
Disable the file to be accessed by another function to do FRITE instruction.
Disable to do FWRITE instruction at the same time with access by another function.

Any errors which can be caused by PLC code indicate the possible errors and their causes in the section for each command in the Q/LSeries Programming Manual (Common). This information is also now available from the Help menu in GX Works2.

By looking up the divide instruction in the manual or help file, we can see that the only reason for a divide instruction to create a 4100 error is a divide by zero.

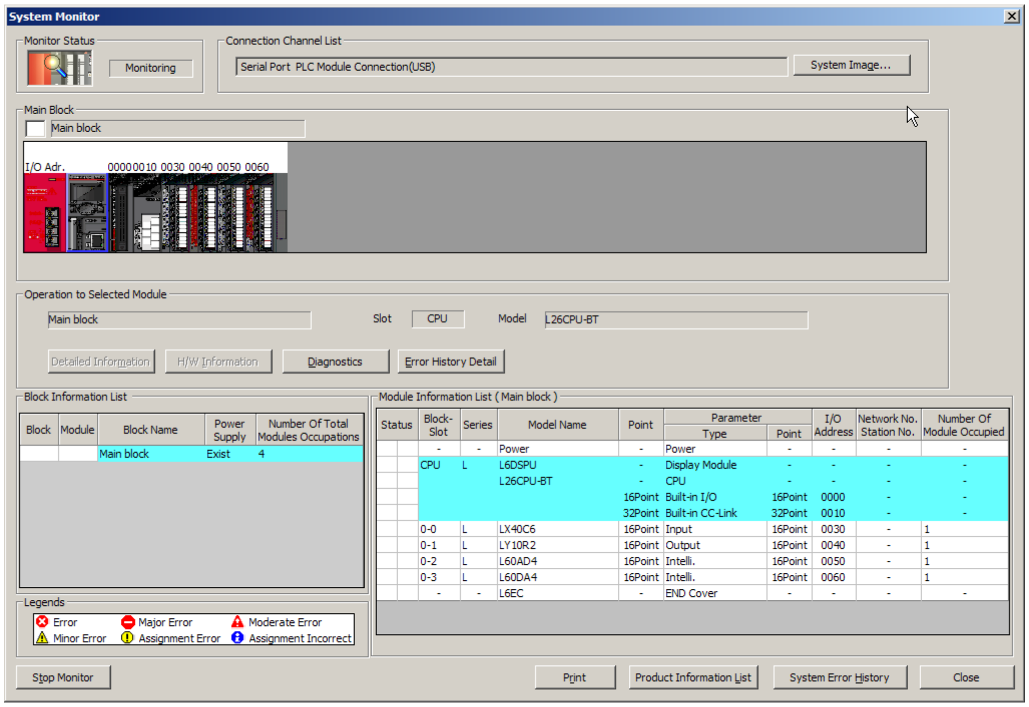
GX Works2 has error help screens for most CPU error codes. If the selected error has a help screen, it will be shown automatically when the Help button is pressed. Detailed PLC error code information is also found in the Help menu. A complete list of error codes and their details is found here.

More detailed information on the error numbers and their causes is detailed in the appropriate programming manual. This information is also available in the F1 help for the command.

Notes

6.5 System Monitor

The system monitor screen provides a snapshot of key information about the system. Data such as the module part numbers, module types (input, output, special function modules, etc.), space occupied, and addressing are available.



Serial Port PLC Module Connection(USB)

Main block

Main block Slot CPU Model L26CPU-BT

Block Information List

Block	Module	Block Name	Power Supply	Number Of Total Modules Occupations
		Main block	Exist	4

Module Information List (Main block)

Status		Block-Slot	Series	Model Name	Point	Parameter		I/O Address	Network No. Station No.	Number Of Module Occupied
						Type	Point			
		-	-	Power	-	Power	-	-	-	-
		CPU	L	L6DSPU	-	Display Module	-	-	-	-
				L26CPU-BT	-	CPU	-	-	-	-
					16Point	Built-in I/O	16Point	0000	-	-
					32Point	Built-in CC-Link	32Point	0010	-	-
		0-0	L	LX40C6	16Point	Input	16Point	0030	-	1
		0-1	L	LY10R2	16Point	Output	16Point	0040	-	1
		0-2	L	L60AD4	16Point	Intelli.	16Point	0050	-	1
		0-3	L	L60DA4	16Point	Intelli.	16Point	0060	-	1
		-	-	L6EC	-	END Cover	-	-	-	-

The top half of the system monitor window shows a pictorial of the connected controller based in the information read in from the units. Above each module is its starting address, as well as a symbol to indicate module status as shown in the legend in the bottom left corner.

By clicking on any module in any base, the Operation to Selected Module section in the upper right will be updated. Depending on the module type, there are 4 buttons which may become available. These buttons offer detailed module information, hardware information, diagnostics, or module error history. The bottom left of the system monitor shows the number of installed modules.

Notes

The bottom right of the system monitor shows parameter allocation and module head addresses. This is settings made in the PLC parameters, or automatic allocation if parameters were not set.

6.5.1 Module Detailed Information

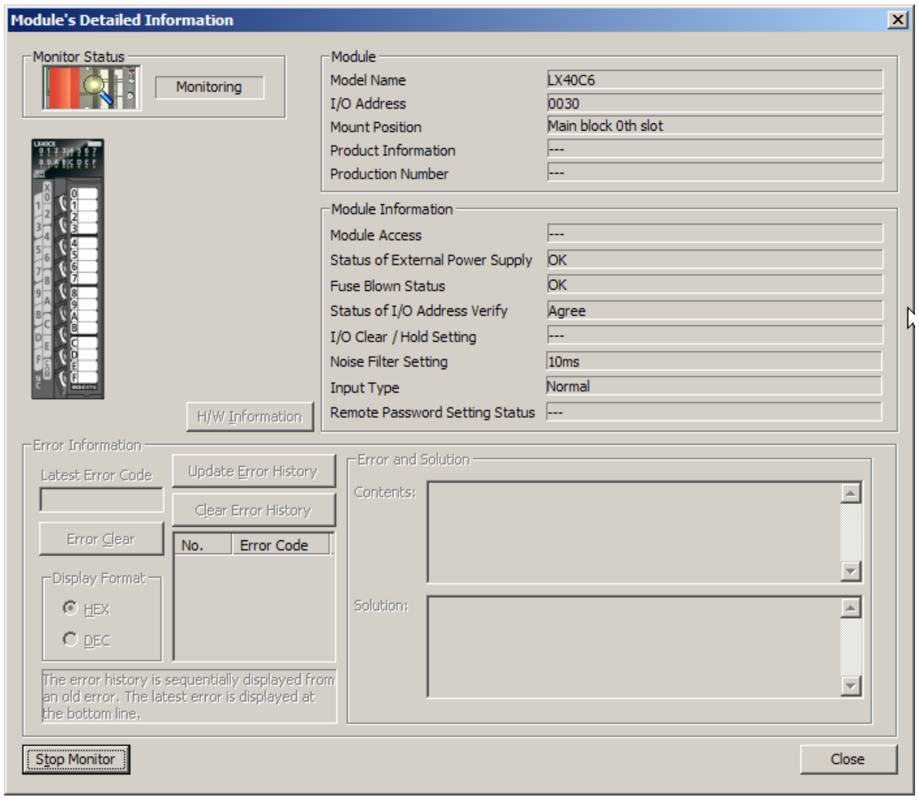
GX Works2 has built-in screens which will provide a more detailed look at a particular module. The information that appears on this screen will vary according to the type of module selected. Some of the information shown on the screens for all modules would include:

- Module name (part number)
- I/O address
- Location within the extension bus
- Serial number (if available)

The detailed information for the module can be accessed from the 'Detailed Information' button or by double clicking on the module in the top half of the System Monitor screen.

Notes

This screen capture is from an input module. It gives the status of the power supply, fuses, and whether or not the module address is in agreement with the I/O Assignment Table, if it is used. Since this is an input module, which has an adjustable response time, this setting is displayed as well (Noise Filter Setting). Input and output cards do not have onboard diagnostics and error tracking, so the error information section is grayed out. No product information is provided as this electronic serial number is not part of an I/O module.



Module

Model Name	LX40C6
I/O Address	0030
Mount Position	Main block 0th slot
Product Information	---
Production Number	---

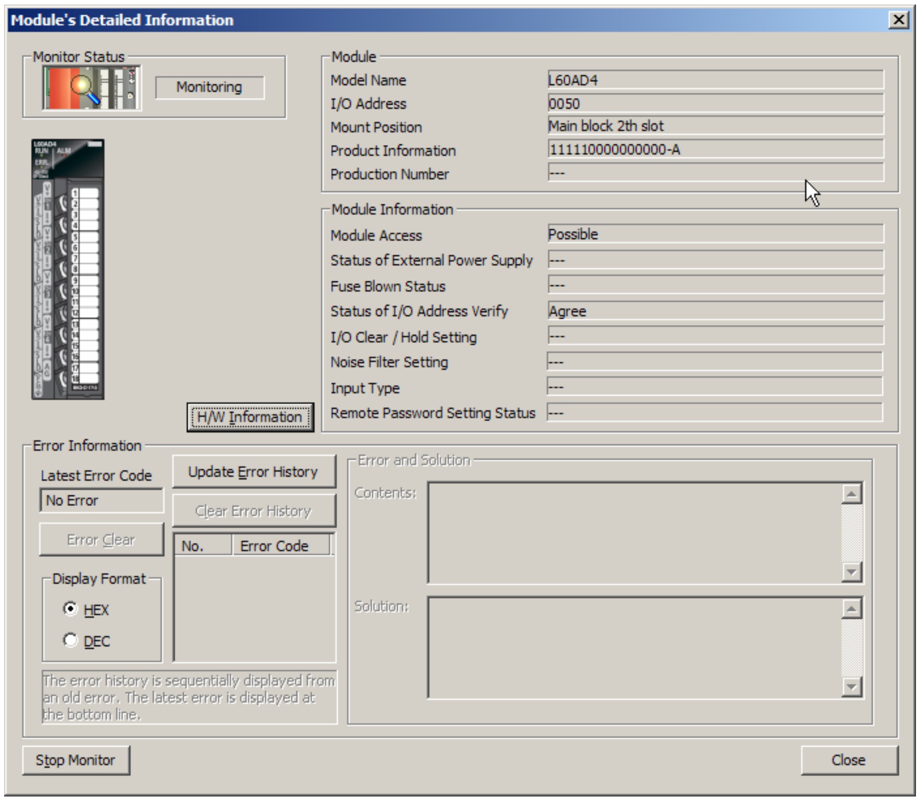
Module Information

Module Access	---
Status of External Power Supply	OK
Fuse Blown Status	OK
Status of I/O Address Verify	Agree
I/O Clear / Hold Setting	---
Noise Filter Setting	10ms
Input Type	Normal
Remote Password Setting Status	---

The error history is sequentially displayed from an old error. The latest error is displayed at the bottom line.

Notes

Other modules, such as intelligent modules, offer even more information. Intelligent modules have an electronic serial number, which shows in the upper right module information box. Intelligent modules also offer their own on-board error monitoring and history. This is shown in the bottom half of the window. The screen below is for an analog input module.



Module's Detailed Information

Monitor Status: Monitoring

Module

Model Name	L60AD4
I/O Address	0050
Mount Position	Main block 2th slot
Product Information	111110000000000-A
Production Number	---

Module Information

Module Access	Possible
Status of External Power Supply	---
Fuse Blown Status	---
Status of I/O Address Verify	Agree
I/O Clear / Hold Setting	---
Noise Filter Setting	---
Input Type	---
Remote Password Setting Status	---

Error Information

Latest Error Code: No Error

Display Format: [✓ HEX] DEC

No.	Error Code

The error history is sequentially displayed from an old error. The latest error is displayed at the bottom line.

Error and Solution

Contents:

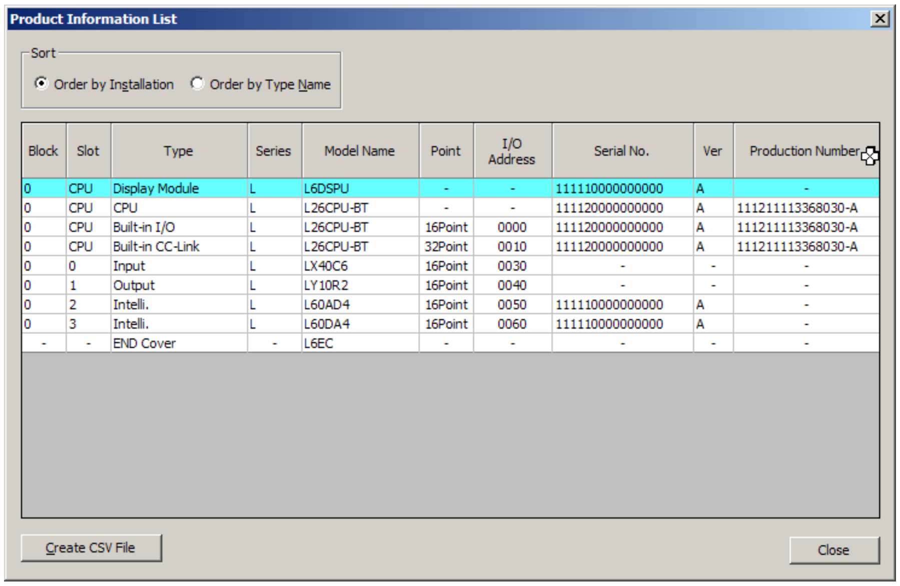
Solution:

Notes

6.5.2 Product Information List

This provides a snapshot of the connected hardware. Some of the useful troubleshooting data includes module rack location and configuration, occupied space, starting address, electronic serial number (if present in module), and product revision info. This screen is available from the ‘Product Information List’ button on the bottom right of the System Monitor screen.

What makes this screen especially useful is that by clicking the ‘Create CSV File’ button in the lower left corner, this information can be exported into an Excel spreadsheet and used for documentation or sent out for analysis.



[✓ Order by Installation] Order by Type Name

Block	Slot	Type	Series	Model Name	Point	I/O Address	Serial No.	Ver	Production Number
0	CPU	Display Module	L	L6DSPU	-	-	111110000000000	A	-
0	CPU	CPU	L	L26CPU-BT	-	-	111120000000000	A	111211113368030-A
0	CPU	Built-in I/O	L	L26CPU-BT	16Point	0000	111120000000000	A	111211113368030-A
0	CPU	Built-in CC-Link	L	L26CPU-BT	32Point	0010	111120000000000	A	111211113368030-A
0	0	Input	L	LX40C6	16Point	0030	-	-	-
0	1	Output	L	LY10R2	16Point	0040	-	-	-
0	2	Intelli.	L	L60AD4	16Point	0050	111110000000000	A	-
0	3	Intelli.	L	L60DA4	16Point	0060	111110000000000	A	-
-	-	END Cover	-	L6EC	-	-	-	-	-

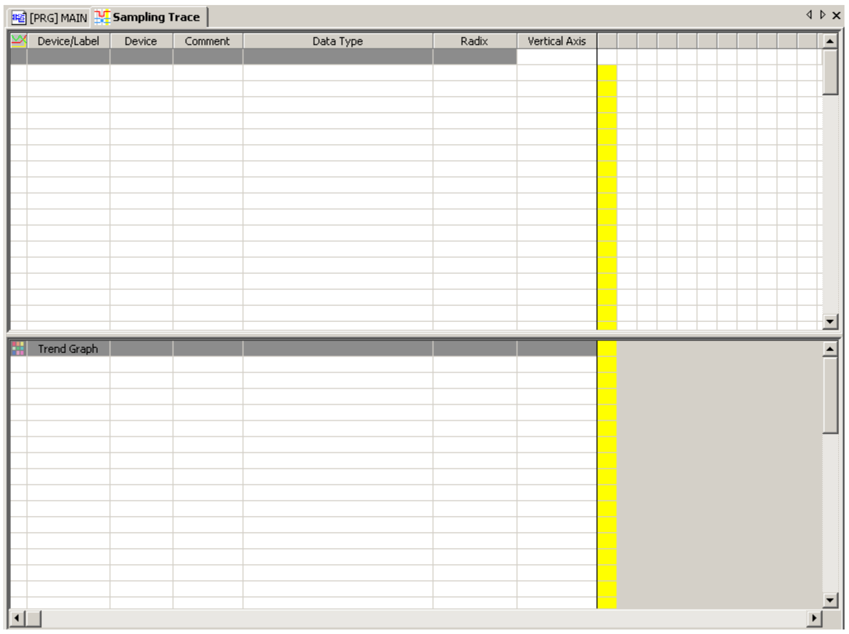
Notes

6.6 Sampling Trace

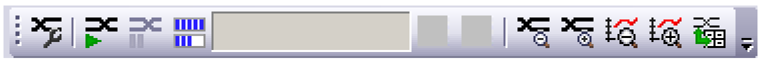
Sampling trace is a diagnostic function available in the L Series and Q Series processors (except the basic models). This utility is used to log data over time when something happens in the program. The data is stored inside the CPU, so there is no time delay for communications. Data trace executes completely in the CPU, and then the results can be uploaded.

This tool can be used to trace up to 8192 samples of 50 bits and 50 words in the High Performance and Universal Q CPUs (except Q00UJCPU) or L Series CPUs.

To start the utility, go to the Debug menu, select Sampling Trace, and then Open Sampling Trace. The following screen is displayed.

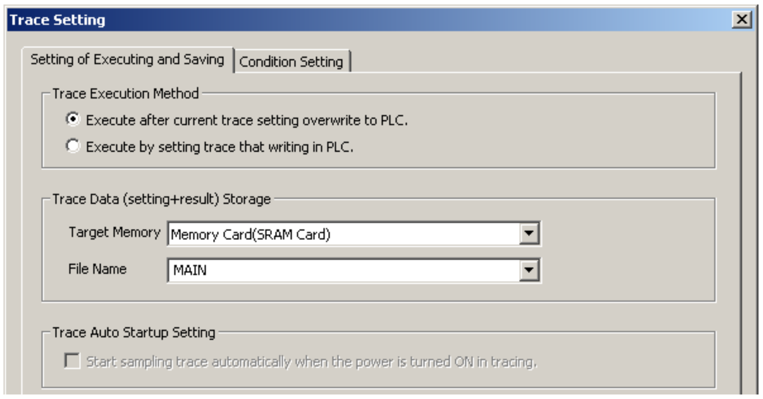


Once the sampling trace window is open, the sampling trace toolbar will be shown, and several new options are added to the Sampling Trace option in the Debug menu.



Notes

The first thing to do after opening the sampling trace screen is configure the trace. This can be done with the toolbar button or Selecting the Debug menu, then Sampling Trace, and then Trace Setting.


Trace Setting

Setting of Executing and Saving | Condition Setting

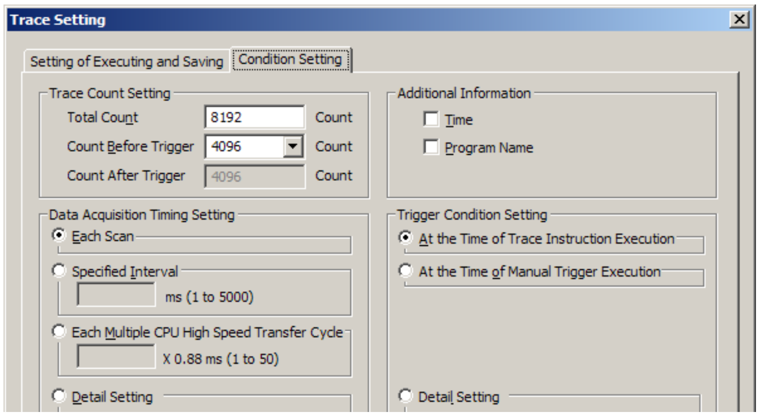
Trace Execution Method
- (●) Execute after current trace setting overwrite to PLC.
- ( ) Execute by setting trace that writing in PLC.

Trace Data (setting+result) Storage
- Target Memory: Memory Card(SRAM Card)
- File Name: MAIN

Trace Auto Startup Setting
- [ ] Start sampling trace automatically when the power is turned ON in tracing.

On this first tab, the choice is given to create new trace settings or use the trace settings in the current connected PLC. The location to store the data is also selected, and the filename for storage. In the Q Series, the data file can be stored on a RAM memory card. On L Series. iQ Series, and Q processors with serial numbers beginning with 07032 or newer, the Standard RAM drive can be used instead of a memory card.

On the Condition Setting tab, the trace is configured. The number of samples, after trigger samples, additional information to log, trace point setup and trigger point setup are configured.


Trace Setting

Setting of Executing and Saving | Condition Setting

Trace Count Setting
- Total Count: 8192 Count
- Count Before Trigger: 4096 Count
- Count After Trigger: 4096 Count

Additional Information
- [ ] Time
- [ ] Program Name

Data Acquisition Timing Setting
- (●) Each Scan
- ( ) Specified Interval: ms (1 to 5000)
- ( ) Each Multiple CPU High Speed Transfer Cycle: X 0.88 ms (1 to 50)
- ( ) Detail Setting

Trigger Condition Setting
- (●) At the Time of Trace Instruction Execution
- ( ) At the Time of Manual Trigger Execution
- ( ) Detail Setting

Notes

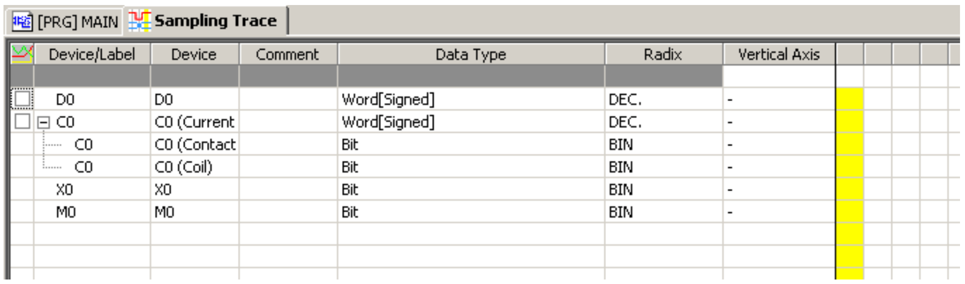
Trace Count Settings configure the number of samples to store, and how many of those samples are stored from before the trigger signal was received.

Additional Information adds information to each entry in the sample trace.

Data Acquisition Timing Setting sets the frequency of the data collection. The choices are each PLC scan, specified time intervals, or detail settings. Detail setting allows selection of an address and the criteria to be met by that address, such as rising or falling edge of a bit, or a word equal to a specified value.

The Trigger Condition Setting section allows the configuration of the trigger signal. This trigger signal can be set to the TRACE instruction execution in the PLC program, manually triggered from GX Works2, or detail settings for an address. The detail setting section is the same as for the timing section, allowing bit or word addresses.

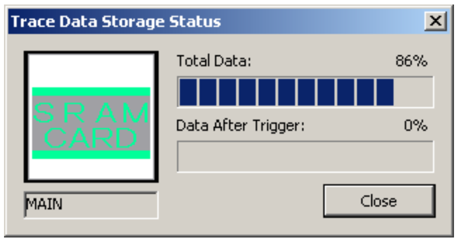
In the top half of the sampling trace window, register the addresses to be logged by entering values into the Device/Label column. Items can also be added to the trace by dragging and dropping them in from the ladder window when the sampling trace window has been opened.



Device/Label	Device	Comment	Data Type	Radix	Vertical Axis
D0	D0		Word[Signed]	DEC.	-
C0	C0 (Current		Word[Signed]	DEC.	-
C0	C0 (Contact		Bit	BIN	-
C0	C0 (Coil)		Bit	BIN	-
X0	X0		Bit	BIN	-
M0	M0		Bit	BIN	-

If the trace is to be started from conditions in the PLC and not from GX Works2, the trace configuration must be registered to the PLC. Select the Register Trace option from the Sampling Trace menu in the Debug menu. If the trace will be started from GX Works2, select Start Trace.

Notes



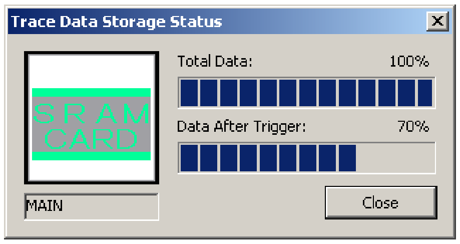
Trace Data Storage Status

Total Data: 86%

Data After Trigger: 0%

MAIN

Once the trace has been registered to the PLC, a window will appear which shows the current trace buffer status. On the left will be a picture of the storage location and the filename selected for the storage data. Once started, the total data bar will begin to move. If the trace is not going to be monitored online, then this window can be closed. It can also be displayed again with the 'Display Trace Buffer Condition' option in the menu or on the toolbar.



Trace Data Storage Status

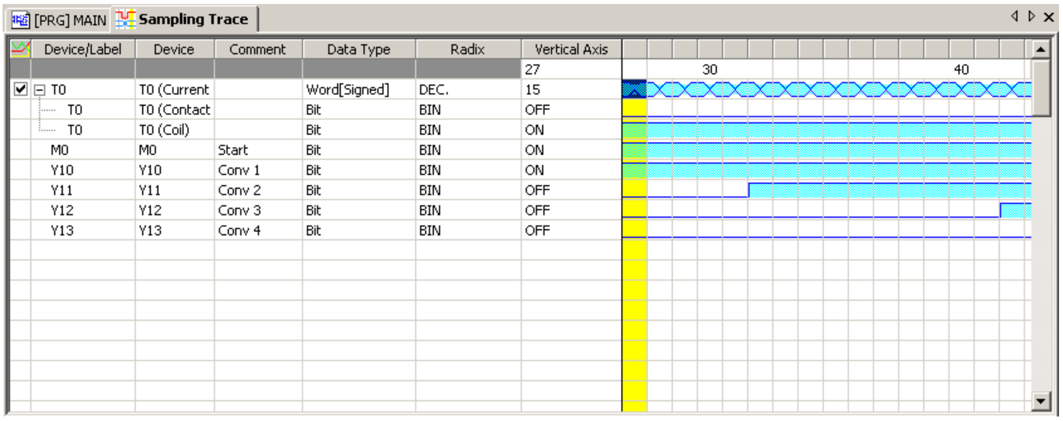
Total Data: 100%

Data After Trigger: 70%

MAIN

Once the trigger signal has been received, the data after trigger bar will begin to move. Once this bar reaches 100%, the data will be uploaded and displayed in the sampling trace window.

Each column on the upper right side of the window will indicate a sample of the data. Bit data will show as a line going up or down to indicate on or off. Numeric data will show crossed lines each time it changes.

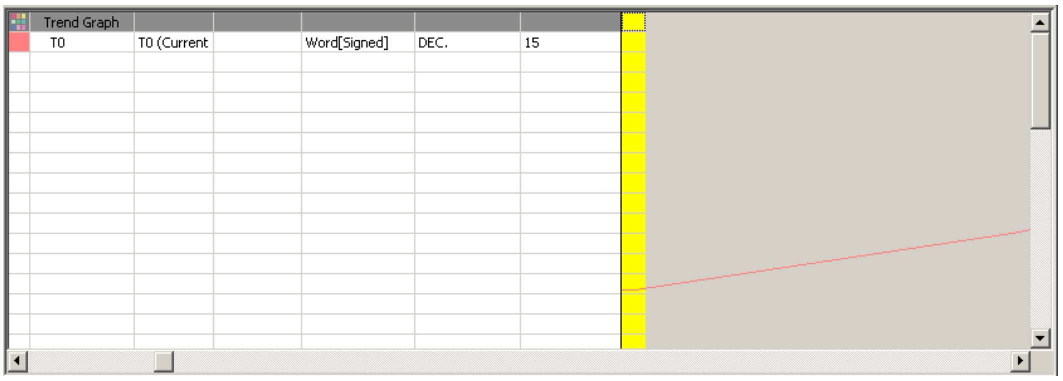


Device/Label	Device	Comment	Data Type	Radix	Vertical Axis
					27
✓ ⊟ T0	T0 (Current		Word[Signed]	DEC.	15
T0	T0 (Contact		Bit	BIN	OFF
T0	T0 (Coil)		Bit	BIN	ON
M0	M0	Start	Bit	BIN	ON
Y10	Y10	Conv 1	Bit	BIN	ON
Y11	Y11	Conv 2	Bit	BIN	OFF
Y12	Y12	Conv 3	Bit	BIN	OFF
Y13	Y13	Conv 4	Bit	BIN	OFF

Samples can be viewed by moving the scrollbar at the bottom of the window left or right. The yellow bar is the cursor, and it can be moved with the arrow keys. The values shown in the Vertical Axis column are the values at the cursor.

Notes

GX Works2 also has the ability to perform trend graphing of the numeric data. To add numeric data to the graph at the bottom of the screen, check the box in front of the address in the top window. This will add a line to the graph in the bottom window. The color of the line can be changed by double clicking on the colored box at the start of the line.



🟥	T0	T0 (Current	Word[Signed]	DEC.	15

If the trace was executed while the PLC was not connected to the PLC, then the data must be uploaded from the PLC. This can be done from the Sampling Trace menu.

Once the data is in GX Works2, it can be easily exported to a .CSV file for reference in other applications like Microsoft Excel. This is done from the Sampling trace menu.

To erase the settings and start over, select Delete All Data from the Sampling Trace menu. Changing the sampling trace configuration will erase the trace data already in GX Works2.

Notes

6.7 EXERCISE – Sampling Trace

Load the program from Lesson 4 back into the CPU.

Configure a sampling trace with the following settings:

- Bit Addresses
  - M0
  - M1
  - Y10
  - Y11
  - Y12
  - Y13
- Words
  - T0
  - T1
  - T2
  - T3
  - T4
  - T5
  - D3
  - D4
  - D5
- Trigger point should be bit M0 turning on
- Take 200 samples at 100msec intervals
- Samples before trigger should be set to 10

This will provide 20 seconds of sample data, with 1 second of data stored from before the trigger is activated. Execute the program by pressing M0 and view the trace results.

Notes

LESSON 7 – Special Addresses

This lesson introduces the special relay and special register areas of the L Series processors.

Lesson Objectives

At the conclusion of this lesson, you will be able to…

- Understand the function of SM and SD memory areas.
- Write code utilizing diagnostic information in the ladder program.

The L Series has a dedicated area of memory with useful system information. There are both bit and word addresses which are controlled by the CPU.

7.1 Special Memory Bits

Special memory bits are in a separate memory location in the L Series processors. The basic layout of these addresses is identical to the Q Series.

The SM bits are listed in Appendix 3 of the MELSEC-Q/L Programming Manual (Common Instructions). The complete list is also available in the online help menu in GX Works2. These relays can be useful when developing your ladder program, and developing diagnostics within your program.

Some of the most commonly used contacts are:

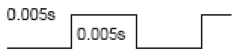
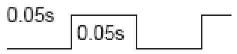
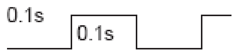
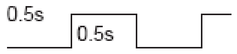
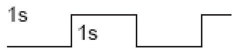
• SM0	General Error Flag	Any PLC Error
• SM52	Battery Low	PLC battery under 2.4V
• SM60	Fuse Blown	Bad fuse or I/O module power problem

Some bits which are not diagnostic related are:

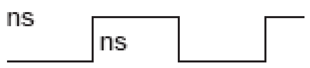
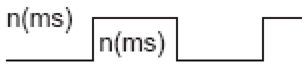
• SM400	Always ON
• SM401	Always OFF
• SM402	On for First Scan only, then off
• SM403	Off for First Scan only, then on

Notes

There are some free running clock pulses in the CPU as well. These can be used for timed operations, such as flashing a warning light on an output. Each pulse is on for the specified time and then off for the same amount of time.

SM409	0.01 second clock	
SM410	0.1 second clock	
SM411	0.2 second clock	
SM412	1 second clock	
SM413	2 second clock	

Two additional free running clocks have a user-defined pulse length. The length of time to run is adjusted in the SD register of the same address (SD414 or SD415). The bit will remain on for the specified time, and then off for the same amount of time.

SM414	2n second clock	
SM415	2n (ms) clock	

As an example, if SD414 is set to 10, SM414 will be on for 10 seconds, then off for 10 seconds, and will repeat constantly.

Notes

7.2 Special Registers

Similar to the special relays, there are a large number of special registers that hold diagnostic and parameter information about the CPU. Information on these registers is in the MELSEC-Q/L Programming Manual (Common Instructions) in Appendix 4. The complete list is also available in the online help menu in GX Works2.

Whenever a fault occurs in the CPU, the ERROR LED on the CPU module will illuminate. The error will not necessarily cause the ladder program to stop executing, although it may. This action is dependent on settings in the PLC parameters.

There are 27 dedicated D registers that hold critical information regarding faults.

- SD0 General PLC Error Code
- SD4 Error Code Type Information
- SD5-SD15 Common Error Information
    - Contents vary by type of error
- SD16-SD26 Individual Error Information
    - Contents vary by type of error

Some of the most common errors are:

- 1600 Battery Error
    - This error occurs when the CPU module battery drops below 2.4 volts. The battery is a 3.6V battery; however it typically measures around 3.69VDC.
    - This fault occurs at initial power up of the unit because the battery is disconnected from the factory so it will not discharge while on the warehouse shelf. The battery is mounted on the inside of the maintenance door, on the bottom of the CPU module.

Notes

- 2200 Missing Parameter File
    - The parameter file holds such information as the type of I/O modules, the amount of memory devices, time bases, etc. Without it, the CPU cannot function.
    - The error is usually the result of forgetting to download the parameters when downloading the program

- 2500 Can't Execute Program
    - Typically occurs when a PLC parameter issue is present. This could be a program in the CPU which is not referenced in the Program tab of PLC Parameters, or a missing program file.

- 3300 Special Parameter Error
    - This error is caused when intelligent module parameters exist in the CPU for modules which do not exist on the controller, or if the head addresses of the modules in the intelligent module parameters do not match the connected PLC.

- 4100 Operation Error or 4101 Operation Error
    - This error results when an instruction executes in an illegal manner- usually due to a programming error.
    - Typical examples are a divide by 0, BCD conversion on a number that is too large, etc.
    - When the instruction that is causing the error is discovered, good practice is to consult the programming manual description for the instruction. Each instruction description has a troubleshooting section detailing common errors.

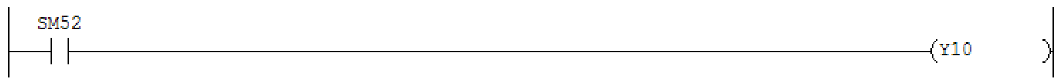
There are many other possible error codes. These codes, with descriptions and suggestions for possible causes, are listed in Chapter 12 of the MELSEC-Q/L Programming Manual (Common Instructions).

Notes

7.3 Troubleshooting Examples

7.3.1 Battery Low Warning Indicator

Since the PLC battery low indicator LED is on the CPU, and since the CPU is typically inside of a control cabinet where it cannot be seen, it is sometimes desirable to have an external indication. To do this, and output can be connected to the low battery warning bit.



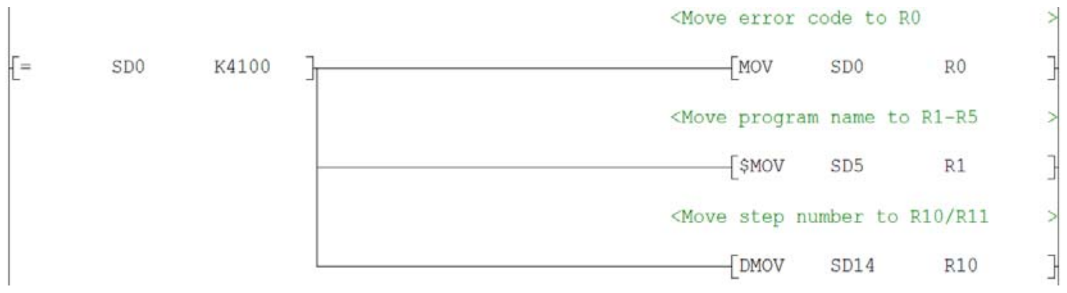
Now when the battery low alarm happens, output Y10 will be energized.

7.3.2 PLC Error Code Backup

Typically, when the power is removed from the PLC, the contents of the diagnostic registers are cleared. Many times the first test performed when a PLC is not functioning properly is to power off and reset the PLC. This will result in the loss of the active error code number and information if the error is not still active when power returns.

To ensure the data is backed up, we can use file registers. As previously discussed, file registers are retentive. So removing the power from the CPU will not clear this information.

To do this, we will monitor for the error code 4100 in SD0, and then use the move instructions covered previously to move the error information to file registers.



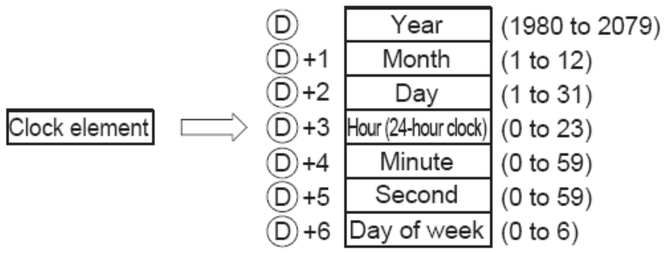
Notes

7.4 Real Time Clock

All Q and L Series processors have a built-in real time clock. This clock is backed up by the PLC battery, and so will maintain its value through power loss.

The clock is accessible in special data registers SD210-SD213 in BCD format. Each byte of the register is used to store a different piece of the real time clock data. It is easier to access the clock using dedicated commands.

The **DATERD** command will read the 4 words of clock data and break it down into 7 consecutive data registers. DATERD requires one parameter, the first of the 7 addresses to store the clock data.



Day of week	Sun	Mon	Tue	Wed	Thu	Fri	Sat
Stored data	0	1	2	3	4	5	6

The same 7 registers can be written back to the CPU clock using the **DATEWR** instruction. It also takes one parameter, which is the first of the 7 addresses containing the data to be written to the clock.

This data can be used for such applications as time stamping of data tables, comparison instructions for time of day operations, or display on an operator interface.

Notes

LESSON 8 – Intelligent Modules

This lesson covers the methods used for communication with intelligent modules.

Lesson Objectives

At the conclusion of this lesson, you will be able to…

- Identify intelligent modules.
- Write instructions to send/receive data to/from a module.
- Use direct addressing to access data within intelligent modules.
- Understand the application of the Intelligent Module Utility.

8.1 Intelligent Module Introduction

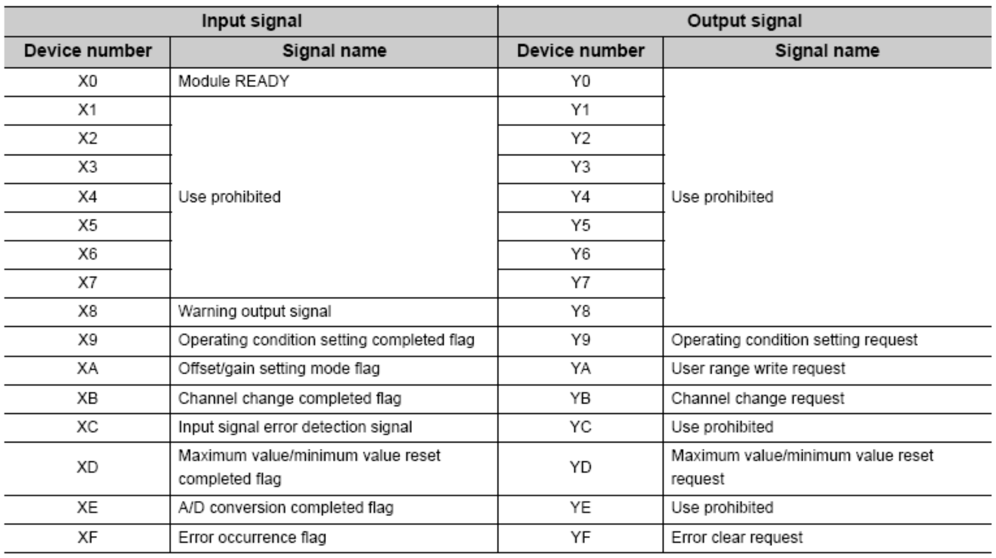
Any module which is not a discrete input or output module is considered an intelligent module. Intelligent modules are used by the controllers in a different way than regular I/O modules. The intelligent modules all have internal memory and perform a higher level hardware function outside of the PLC's typical scan cycle. They are constantly updating data stored within the module.

Communication with these modules can be done one of several ways.

To understand the way the PLC sets allocation of these modules is important. Intelligent modules do still occupy I/O addresses. Depending on the card, it may require 16 bits or 32 bits. The module may make use of both inputs and outputs in these addresses. So if the head address of the module is 0020 and it occupies 32 bits, the module is assigned X20-X3F and Y20-Y3F.

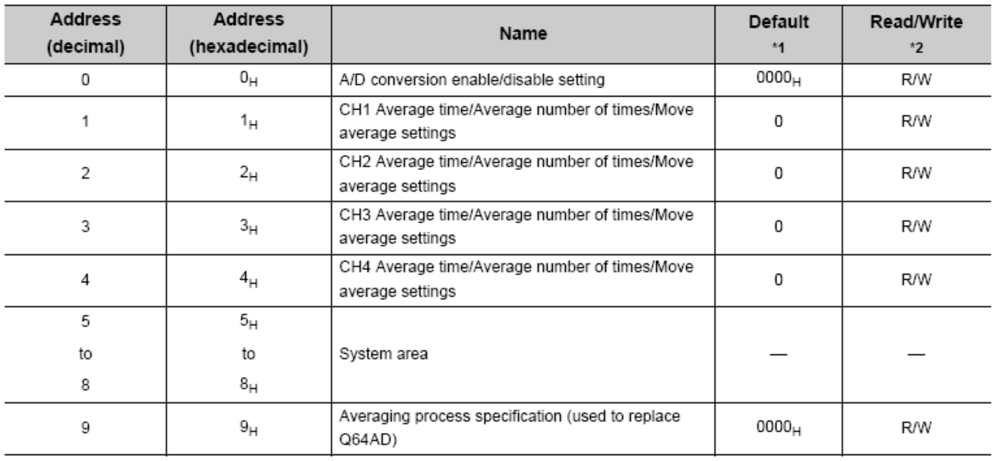
The use of each of these I/O points varies by module. The purpose of the I/O points is detailed in the manual for each module. The table on the next page shows the input and output allocation for the L60AD4 Analog Input Module. The addresses marked as 'use prohibited' should not be accessed within the program.

Notes



Input signal		Output signal	
Device number	Signal name	Device number	Signal name
X0	Module READY	Y0	Use prohibited
X1	Use prohibited	Y1	
X2		Y2	
X3		Y3	
X4		Y4	
X5		Y5	
X6		Y6	
X7		Y7	
X8	Warning output signal	Y8	
X9	Operating condition setting completed flag	Y9	Operating condition setting request
XA	Offset/gain setting mode flag	YA	User range write request
XB	Channel change completed flag	YB	Channel change request
XC	Input signal error detection signal	YC	Use prohibited
XD	Maximum value/minimum value reset completed flag	YD	Maximum value/minimum value reset request
XE	A/D conversion completed flag	YE	Use prohibited
XF	Error occurrence flag	YF	Error clear request

There is no automatic transmission of data from or to the intelligent modules. Instead, intelligent modules have a series of buffer memory locations within the module's internal memory. The function of each buffer memory location varies by module. The purpose of the buffer memory locations is detailed in the manual for each module. Below is a section of the list for the L60AD4 module.



Address (decimal)	Address (hexadecimal)	Name	Default *1	Read/Write *2
0	0_H	A/D conversion enable/disable setting	0000_H	R/W
1	1_H	CH1 Average time/Average number of times/Move average settings	0	R/W
2	2_H	CH2 Average time/Average number of times/Move average settings	0	R/W
3	3_H	CH3 Average time/Average number of times/Move average settings	0	R/W
4	4_H	CH4 Average time/Average number of times/Move average settings	0	R/W
5 to 8	5_H to 8_H	System area	—	—
9	9_H	Averaging process specification (used to replace Q64AD)	0000_H	R/W

These locations work like mailboxes, storing the data for the module. The data is accessed by reading or writing from or to the buffer memory location.

Notes

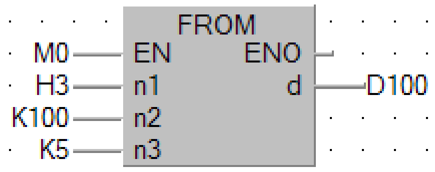
8.2 TO/FROM Instructions

The method used to read or write data to buffer memory locations on intelligent modules is identical no matter which type of intelligent module is being used. There are dedicated instructions in the CPU for reading data from a module or writing data to a module.

The FROM command is used to read data from an intelligent module. The TO command is used to write data to an intelligent module.

These commands can be coded in ladder, structured ladder, or structured text. Examples of each are shown below. The commands shown below will all read 5 words of data from the module at address 30, starting at buffer memory 100, and place the results in 5 registers starting at D100.

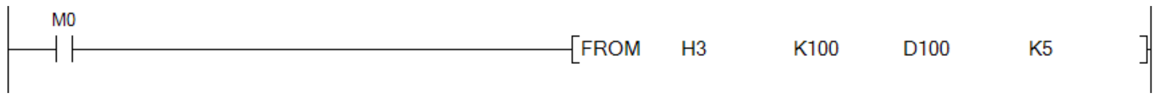
Structured Ladder



Structured Text

```
FROM( M0 , H3 , K100 , K5 , D100 );
```

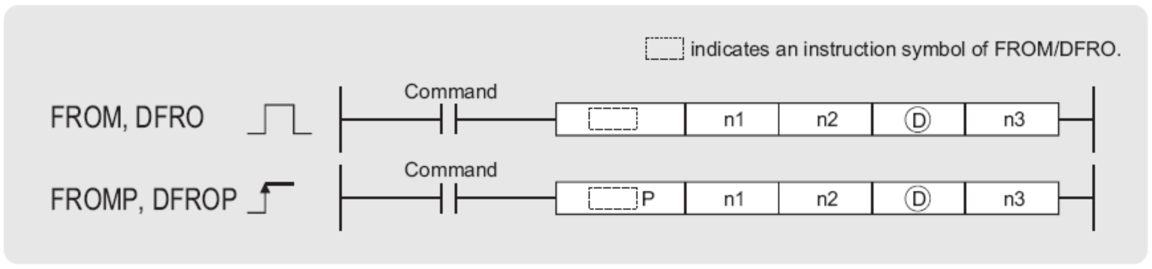
Ladder



Notes

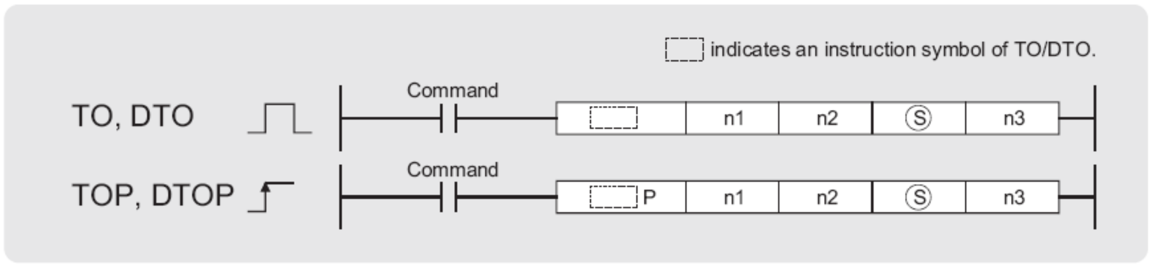
In the ladder programming language, the commands are coded using the standard square brackets.

The FROM instruction retrieves data from one or more sequential buffer memory locations and stores it into consecutive data registers in the CPU.



n1 : Head I/O number of an intelligent function module (BIN 16 bits)
n2 : Head address of data to be read (BIN 16 bits)
Ⓓ : Head number of the devices where the read data will be stored (BIN 16/32 bits)
n3 : Number of data blocks to be read (BIN 16 bits)

The TO instruction sends data from one or more consecutive addresses in the CPU memory to consecutive buffer memory locations in the intelligent module.

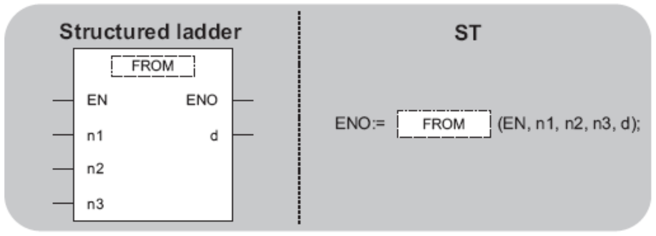


n1 : Head I/O number of an intelligent function module (BIN 16 bits)
n2 : Head address of the area where data is written (BIN 16 bits)
Ⓢ : Data to be written or head number of the devices where the data to be written is stored (BIN 16/32 bits)
n3 : Number of data blocks to be written (BIN 16 bits)

When the data designated as the source (S) is a fixed numeric value, that numeric value is written to all of the destination addresses.

Notes

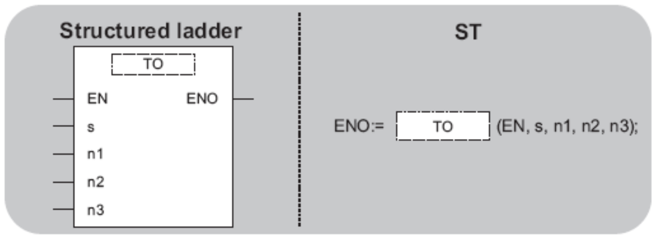
In the structured programming languages, these instructions are coded in a slightly different way. Parameters all have the same designators (n1, n2, n3, s, d) and still serve the same functions, but are input in a different order.



▭ indicates any of the following instructions.

FROM	FROMP
DFRO	DFROP

Input argument,	EN:	Executing condition	:Bit
	n1:	Start I/O number of the intelligent function module	:ANY16
	n2:	Start address of data to be read	:ANY16
	n3:	Number of data to be read	:ANY16
Output argument,	ENO:	Execution result	:Bit
	d:	Read data	:ANY16/32



▭ indicates any of the following instructions.

TO	TOP
DTO	DTOP

Input argument,	EN:	Executing condition	:Bit
	s:	Data to be written	:ANY16/32
	n1:	Start I/O number of the intelligent function module	:ANY16
	n2:	Start device for writing data	:ANY16
	n3:	Number of data to be written	:ANY16
Output argument,	ENO:	Execution result	:Bit

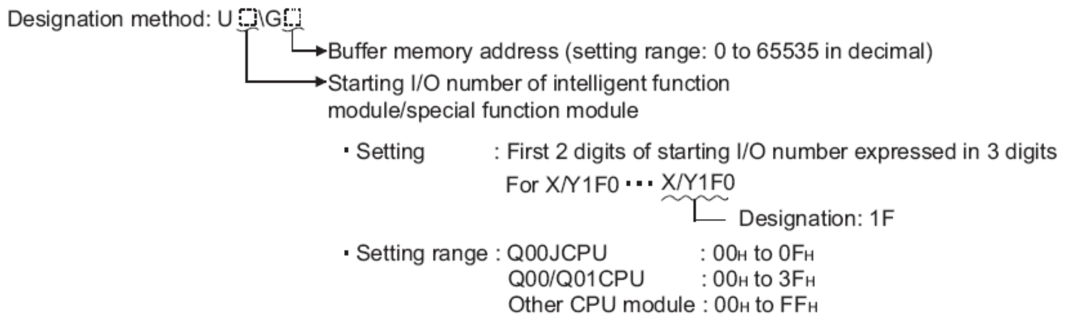
Notes

8.3 U\G Addresses

In newer controllers, such as the Q Series, L Series, and FX3U, there is a direct access method for reading and writing to buffer memory locations with standard commands. Instead of using TO and FROM, a special address allows most 16-bit or 32-bit word commands to access buffer memory.

The U\G address consists of 2 parts.

- The U number is the head address of the module (without the 0 on the end, just like the TO or FROM instruction).
- The G number is the buffer memory address (in decimal).


Designation method: U ◻\G◻
Buffer memory address (setting range: 0 to 65535 in decimal)
Starting I/O number of intelligent function module/special function module
· Setting : First 2 digits of starting I/O number expressed in 3 digits
For X/Y1F0 ··· X/Y1F0
Designation: 1F
· Setting range : Q00JCPU : 00H to 0FH
Q00/Q01CPU : 00H to 3FH
Other CPU module : 00H to FFH

So the ends result of the two commands below is the same:

- FROM H4 K20 D0 K3
- BMOV U4\G20 D0 K3

These 2 commands are also identical:

- TO H7 K15 D10 K1
- MOV D10 U7\G15

The U\G addresses actually operate slightly faster than TO and FROM instructions. But each time the U\G addresses are used, the data is read. So using the same U\G address several times will in fact be slower than using a FROM instruction and then referencing the data registers which it was stored to.

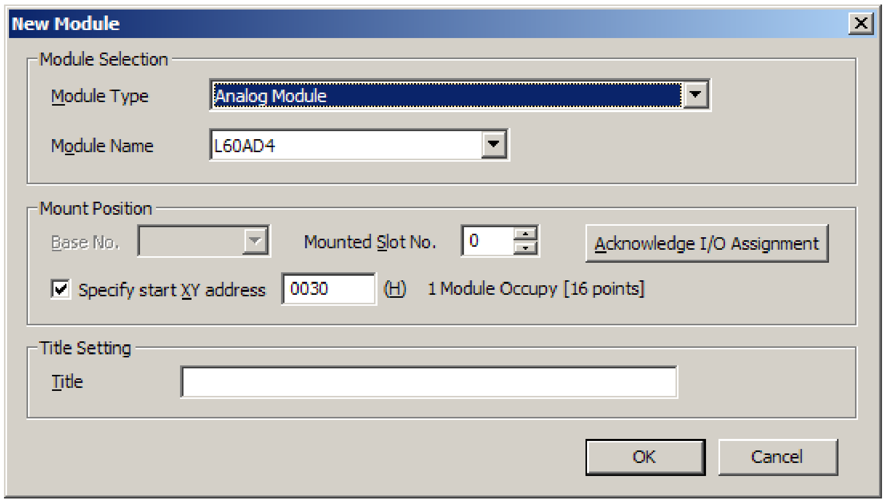
Notes

8.4 Intelligent Function Utility

For the L Series, GX Works2 offers an intelligent function module utility designed to simplify the process of communication with intelligent function modules. There are many types of intelligent function module utility available.

- Analog Modules
- High Speed Counters
- LD75 Positioning Modules
- Serial Communication Modules
- Simple Motion Modules

In GX Works2, these utilities are accessed by adding an intelligent module into the Intelligent Function Module folder in the project navigation tree. To do this, right click on the folder and choose New Module. The screen below will be shown.



Module Type: Analog Module
Module Name: L60AD4
Mounted Slot No.: 0
[✓] Specify start XY address: 0030 (H) 1 Module Occupy [16 points]
Title:

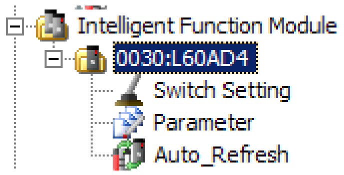
The top list shows the various configuration utilities available in GX Works2. Any of these modules can be configured without writing logic with this utility. Once the module type is selected in the top window, the modules offered within that type are selected in the second box.

Notes

The middle portion of the window is used to configure the location of the module. It can be configured via the slot number or the starting I/O address of the desired module. The Acknowledge I/O Assignment button shows the PLC layout as configured in the PLC Parameters for easy selection.

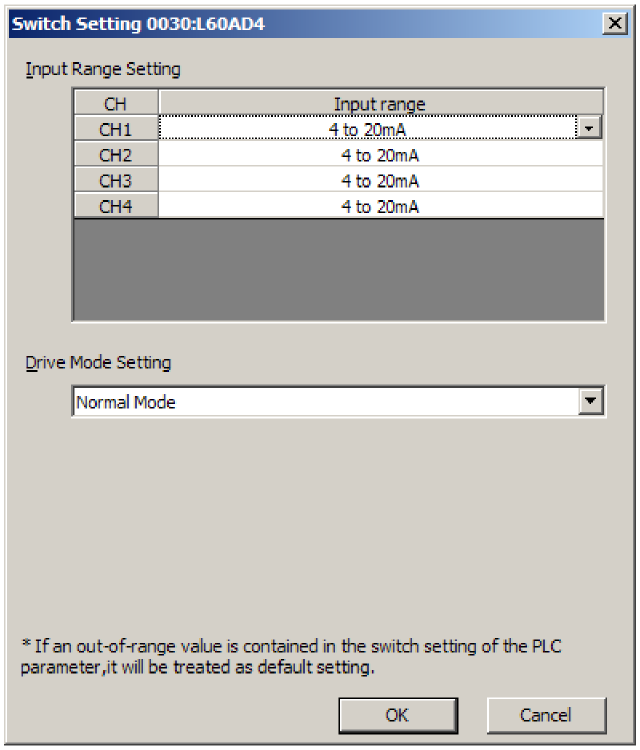
The title setting is a method of documenting the function of this module. It is not required that this field be filled in.

Once the module is added, the settings for that card can be adjusted by clicking the [+] next to the module to reveal the configuration options. Different modules will offer different options. These are intelligent module switch settings, parameters, and automatic refresh data.



The switch setting option allows the adjustment of the software switch settings. Software switches were used in the L Series in place of physical DIP switches for module configuration. By storing these in the PLC parameters instead of on the modules, replacement of a damaged module is easier.

Settings are chosen in the selection dialog, and when finished, the software switches for that module are updated in the PLC parameters. Shown here is the switch setting screen for the L60AD4 analog input module. Each module's options will be different.



Input Range Setting

CH	Input range
CH1	4 to 20mA
CH2	4 to 20mA
CH3	4 to 20mA
CH4	4 to 20mA

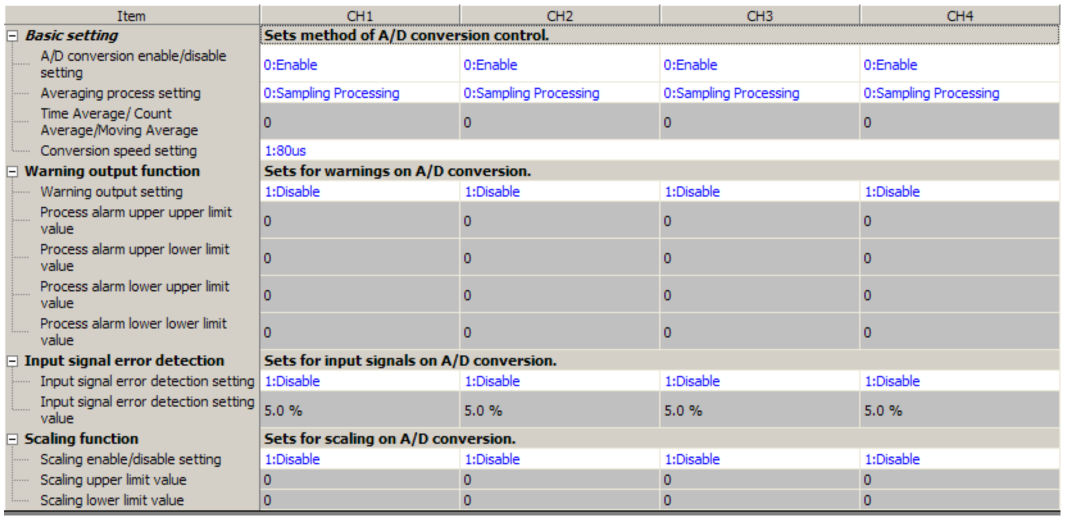
Drive Mode Setting

Normal Mode

* If an out-of-range value is contained in the switch setting of the PLC parameter,it will be treated as default setting.

Notes

Parameter is the window where configuration information about the module is modified. This window opens into the tabbed workspace on the right side. Each module’s options will be different. Help for the currently selected setting is always shown across the bottom of the window. Shown here are the parameters for an L60AD4 module.

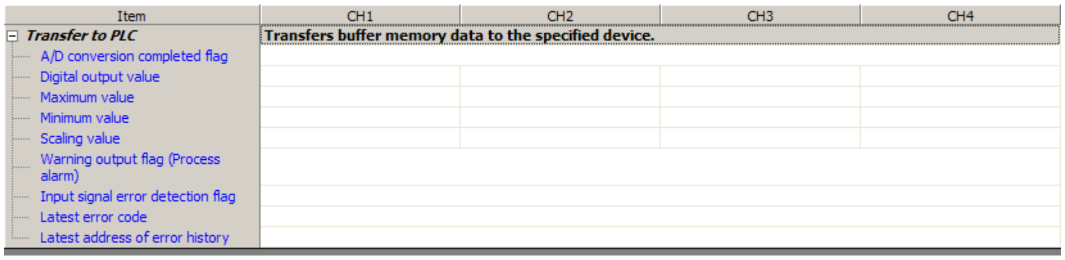


Item	CH1	CH2	CH3	CH4
⊟ ***Basic setting***	**Sets method of A/D conversion control.**			
A/D conversion enable/disable setting	0:Enable	0:Enable	0:Enable	0:Enable
Averaging process setting	0:Sampling Processing	0:Sampling Processing	0:Sampling Processing	0:Sampling Processing
Time Average/ Count Average/Moving Average	0	0	0	0
Conversion speed setting	1:80us			
⊟ **Warning output function**	**Sets for warnings on A/D conversion.**			
Warning output setting	1:Disable	1:Disable	1:Disable	1:Disable
Process alarm upper upper limit value	0	0	0	0
Process alarm upper lower limit value	0	0	0	0
Process alarm lower upper limit value	0	0	0	0
Process alarm lower lower limit value	0	0	0	0
⊟ **Input signal error detection**	**Sets for input signals on A/D conversion.**			
Input signal error detection setting	1:Disable	1:Disable	1:Disable	1:Disable
Input signal error detection setting value	5.0 %	5.0 %	5.0 %	5.0 %
⊟ **Scaling function**	**Sets for scaling on A/D conversion.**			
Scaling enable/disable setting	1:Disable	1:Disable	1:Disable	1:Disable
Scaling upper limit value	0	0	0	0
Scaling lower limit value	0	0	0	0

For this analog module, configuration for all 4 analog input channels is shown. Channels can be enabled or disabled, configured for sampling or averaging, and number of times to sample or samples over a time period. This module also offers a built-in scaling function, configurable at the bottom of the columns.

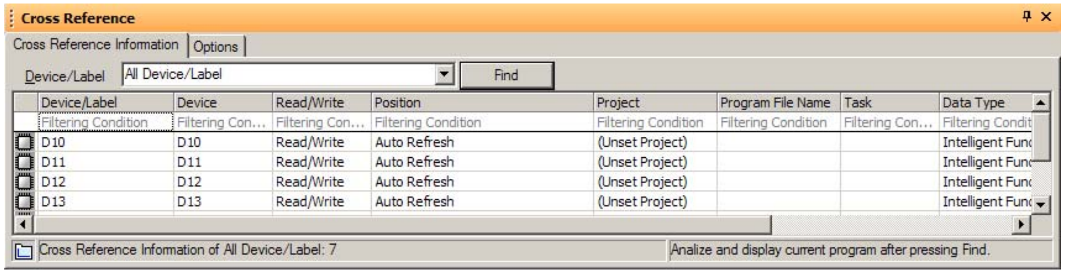
Notes

Auto refresh allows the data from the module, such as analog input values, maximum and minimum values, and module error code can be automatically sent to addresses in the CPU. Shown here is the L60AD4 module. Clicking in any of the boxes will allow the PLC address of this data to be edited.



Item	CH1	CH2	CH3	CH4
⊟ ***Transfer to PLC***	**Transfers buffer memory data to the specified device.**			
A/D conversion completed flag				
Digital output value				
Maximum value				
Minimum value				
Scaling value				
Warning output flag (Process alarm)				
Input signal error detection flag				
Latest error code				
Latest address of error history				

When performing a cross reference on one address or the entire project, addresses used in the intelligent module utilities will show as used. Which module they were used in can be determined from the cross reference window.



Cross Reference

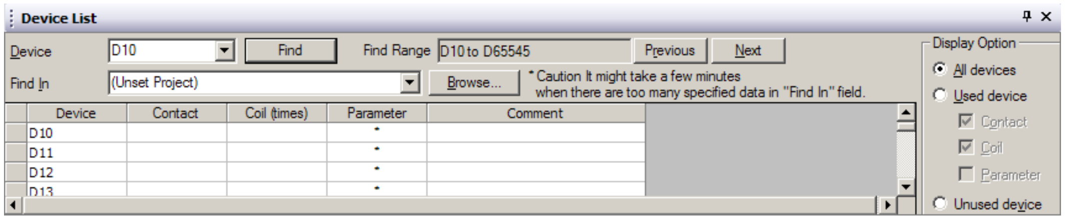
Cross Reference Information | Options

Device/Label: All Device/Label — Find

Device/Label	Device	Read/Write	Position	Project	Program File Name	Task	Data Type
Filtering Condition	Filtering Con...	Filtering Con...	Filtering Condition	Filtering Condition	Filtering Condition	Filtering Con...	Filtering Condit
D10	D10	Read/Write	Auto Refresh	(Unset Project)			Intelligent Func
D11	D11	Read/Write	Auto Refresh	(Unset Project)			Intelligent Func
D12	D12	Read/Write	Auto Refresh	(Unset Project)			Intelligent Func
D13	D13	Read/Write	Auto Refresh	(Unset Project)			Intelligent Func

Cross Reference Information of All Device/Label: 7 | Analize and display current program after pressing Find.

Devices used in the automatic refresh tab will show as used with an asterisk in the parameter column of the Device List window.



Device List

Device: D10 — Find — Find Range: D10 to D65545 — Previous — Next

Find In: (Unset Project) — Browse...

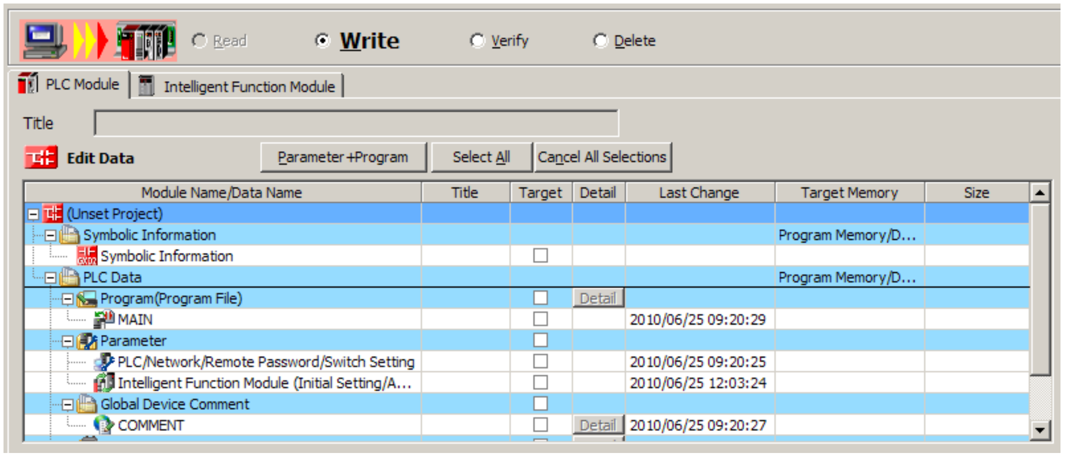
* Caution It might take a few minutes when there are too many specified data in "Find In" field.

Display Option: [✓ All devices] / Used device (☑ Contact, ☑ Coil, ☐ Parameter) / Unused device

Device	Contact	Coil (times)	Parameter	Comment
D10			*	
D11			*	
D12			*	
D13			*	

Notes

When downloading to the CPU, there will be a new check box for the intelligent function module parameters. Under the Parameter section, this option is called ‘Intelligent Function Module’. Check this box to send the intelligent module parameters to the PLC.


Read [✓ Write] Verify Delete

PLC Module | Intelligent Function Module

Title

Edit Data — Parameter+Program | Select All | Cancel All Selections

Module Name/Data Name	Title	Target	Detail	Last Change	Target Memory	Size
(Unset Project)						
Symbolic Information					Program Memory/D...	
Symbolic Information		☐				
PLC Data					Program Memory/D...	
Program(Program File)		☐	Detail			
MAIN		☐		2010/06/25 09:20:29		
Parameter		☐				
PLC/Network/Remote Password/Switch Setting		☐		2010/06/25 09:20:25		
Intelligent Function Module (Initial Setting/A...		☐		2010/06/25 12:03:24		
Global Device Comment		☐				
COMMENT		☐	Detail	2010/06/25 09:20:27		

It is important to note that if intelligent module parameters were sent to the CPU and then cards are reordered or removed, the parameters must be updated or deleted. The PLC will fault with a parameter error if parameters are set for a module which does not exist, or the wrong module type at a starting address.

Notes

8.5 EXERCISE – Intelligent Module Access

An L Series controller has the following modules installed.

L26CPU-BT, L60AD4, D60DA4

Write the TO/FROM instructions or logic using U/G addresses to accomplish the following:

The L60AD4 has 4 analog inputs. The analog inputs will be a value from 0 to 20,000 indicating a voltage reading of 0 to 10VDC.

- Read the 4 values into the CPU and store them in D100 through D103.
- Read the error code information from the module and store it to D105.

The L60DA4 has 4 analog outputs. These outputs are a numeric value 0 to 20,000 to cause an output of 0 to 10VDC.

- Write the PLC code to limit the output data in D120 through D123 in the CPU to the range of 0-20,000 and then output the limited values to the analog output module.
- Read the error code from the module into D125.

Notes

LESSON 9 – PLC Parameters

This lesson explains the configuration of PLC operation using the PLC parameters.

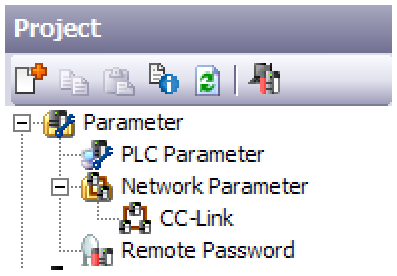
Lesson Objectives

At the conclusion of this lesson, you will be able to…

- Understand the use of the PLC parameters function.
- Configure CPU operation settings.
- Create I/O assignment and configure I/O modules.

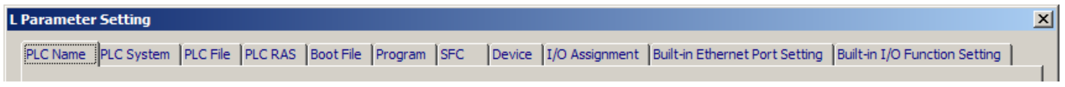
9.1 PLC Parameters

PLC configuration parameters are specific to the project, and are stored with the GX Works2 project. Parameters adjust such things as the operation at error, I/O assignment, memory allocation, and file storage.



To access the PLC parameters, click on the [+] next to Parameters in the project tree, and then double click on PLC Parameters. Network parameters are found in the same place, in a subfolder called Network Parameter.

The color of the text on the tabs in PLC Parameters is important. If the text on the tab is pink, no settings on that tab have been changed from defaults. If the tab color is blue, settings have been made on that tab.



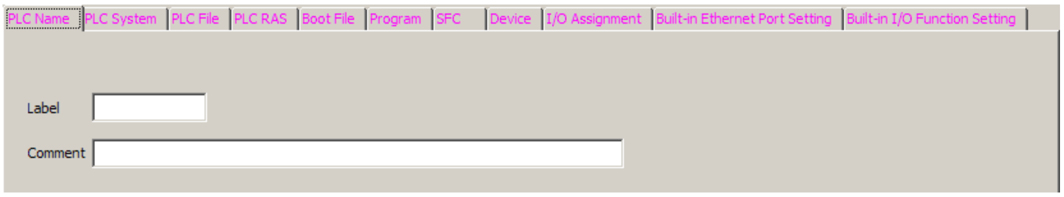
On the bottom of each tab is a button labeled ‘Default’ which will set only the settings on that one tab to the factory defaults.

Parameters can be printed with the Print button on the bottom of the parameter tabs. All parameters are printed no matter which tab is open.

Notes

9.2 PLC Name

The PLC Name tab allows the programmer to assign a label and a comment that helps to define the application of this program. This information is stored with the project in the CPU.

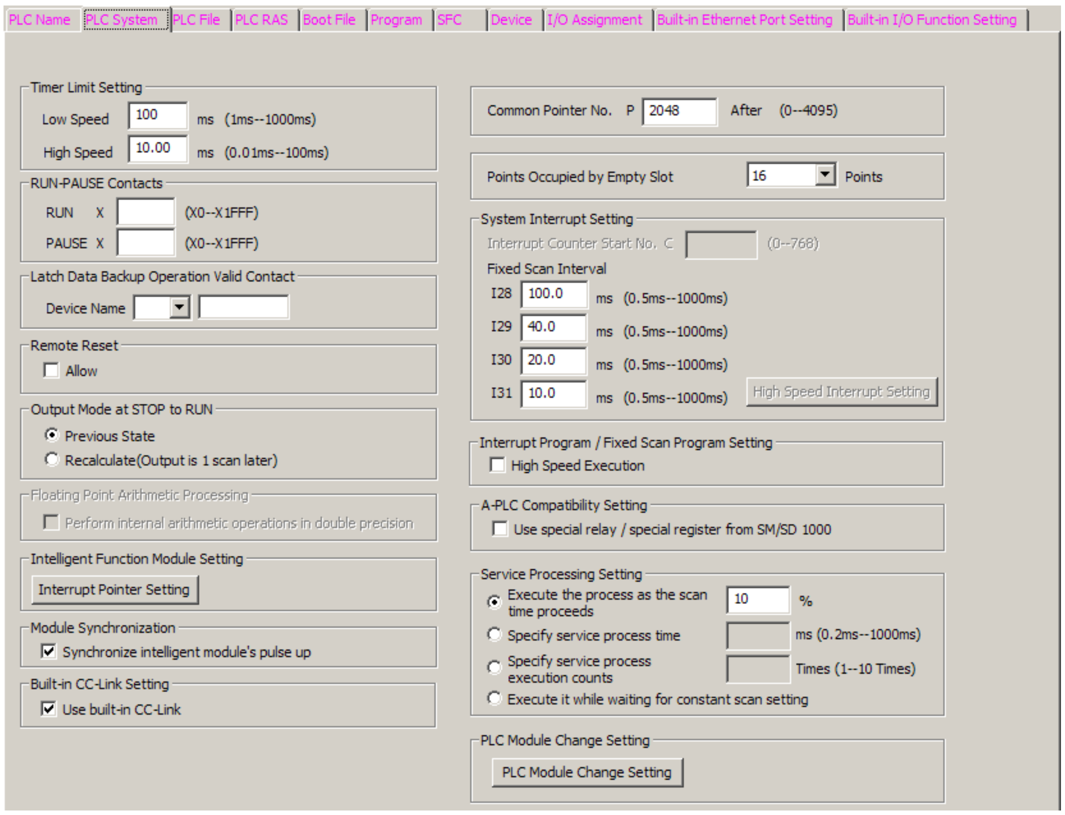


The Label setting is useful when working with the L Series and Q Series processors with built in Ethernet. The label is shown in the Connection dialog when searching the network for available CPUs with built-in Ethernet.

Notes

9.3 PLC System

The PLC system tab deals with basic configuration parameters for the CPU. The two most common changes on this tab are the time bases for low speed and high speed timers and the points occupied by an empty slot.



PLC Name | PLC System | PLC File | PLC RAS | Boot File | Program | SFC | Device | I/O Assignment | Built-in Ethernet Port Setting | Built-in I/O Function Setting

Timer Limit Setting
Low Speed 100 ms (1ms--1000ms)
High Speed 10.00 ms (0.01ms--100ms)

RUN-PAUSE Contacts
RUN X (X0--X1FFF)
PAUSE X (X0--X1FFF)

Latch Data Backup Operation Valid Contact
Device Name

Remote Reset
☐ Allow

Output Mode at STOP to RUN
● Previous State
○ Recalculate(Output is 1 scan later)

Floating Point Arithmetic Processing
☐ Perform internal arithmetic operations in double precision

Intelligent Function Module Setting
Interrupt Pointer Setting

Module Synchronization
☑ Synchronize intelligent module's pulse up

Built-in CC-Link Setting
☑ Use built-in CC-Link

Common Pointer No. P 2048 After (0--4095)

Points Occupied by Empty Slot 16 Points

System Interrupt Setting
Interrupt Counter Start No. C (0--768)
Fixed Scan Interval
I28 100.0 ms (0.5ms--1000ms)
I29 40.0 ms (0.5ms--1000ms)
I30 20.0 ms (0.5ms--1000ms)
I31 10.0 ms (0.5ms--1000ms)
High Speed Interrupt Setting

Interrupt Program / Fixed Scan Program Setting
☐ High Speed Execution

A-PLC Compatibility Setting
☐ Use special relay / special register from SM/SD 1000

Service Processing Setting
● Execute the process as the scan time proceeds 10 %
○ Specify service process time ms (0.2ms--1000ms)
○ Specify service process execution counts Times (1--10 Times)
○ Execute it while waiting for constant scan setting

PLC Module Change Setting
PLC Module Change Setting

Timer limit setting configures the time base for low speed and high speed timers as discussed earlier.

The RUN and PAUSE signals allow any PLC input to be defined to work as a run switch or pause switch.

The latch data backup operation is used to set a contact address to backup latched memory areas to standard ROM before powering down the CPU for extended amounts of time.

Notes

Remote reset must be checked if software reset of PLC is to be allowed. By default, this feature is disabled for safety reasons.

At the bottom of the first column is a check box to enable the built-in CC-Link module on the L26CPU-BT. By default this is on, but if the module is not to be used, it can be turned off here to prevent error detection.

Common pointer setting is used to set a range of pointer addresses to be local to specific programs. By default all pointers are global.

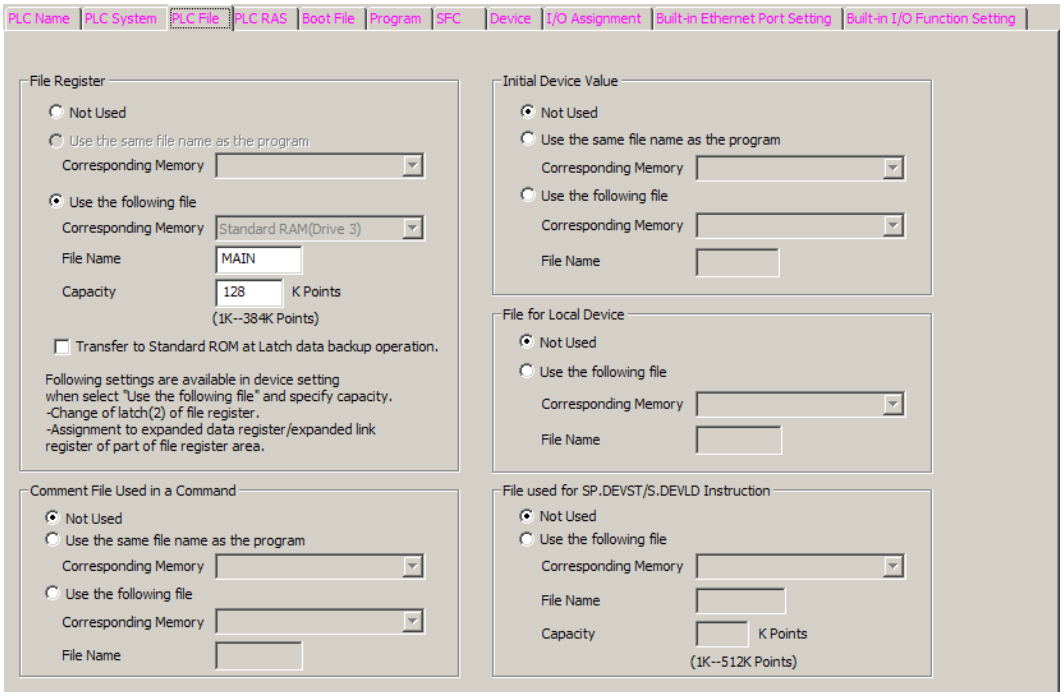
Points occupied by an empty slot sets the default I/O allocation for all empty slots.

System interrupt settings configure the time interval for timed interrupts.

Notes

9.4 PLC File

The PLC file tab allocates storage locations for data which is not automatically allocated in the CPU. Each option is assigned a location and file name to store the data.



File Register
- Not Used
- Use the same file name as the program
  - Corresponding Memory
- [✓ Use the following file]
  - Corresponding Memory: Standard RAM(Drive 3)
  - File Name: MAIN
  - Capacity: 128 K Points (1K--384K Points)
- ☐ Transfer to Standard ROM at Latch data backup operation.

Following settings are available in device setting when select "Use the following file" and specify capacity.
-Change of latch(2) of file register.
-Assignment to expanded data register/expanded link register of part of file register area.

Comment File Used in a Command
- [✓ Not Used]
- Use the same file name as the program
  - Corresponding Memory
- Use the following file
  - Corresponding Memory
  - File Name

Initial Device Value
- [✓ Not Used]
- Use the same file name as the program
  - Corresponding Memory
- Use the following file
  - Corresponding Memory
  - File Name

File for Local Device
- [✓ Not Used]
- Use the following file
  - Corresponding Memory
  - File Name

File used for SP.DEVST/S.DEVLD Instruction
- [✓ Not Used]
- Use the following file
  - Corresponding Memory
  - File Name
  - Capacity K Points (1K--512K Points)

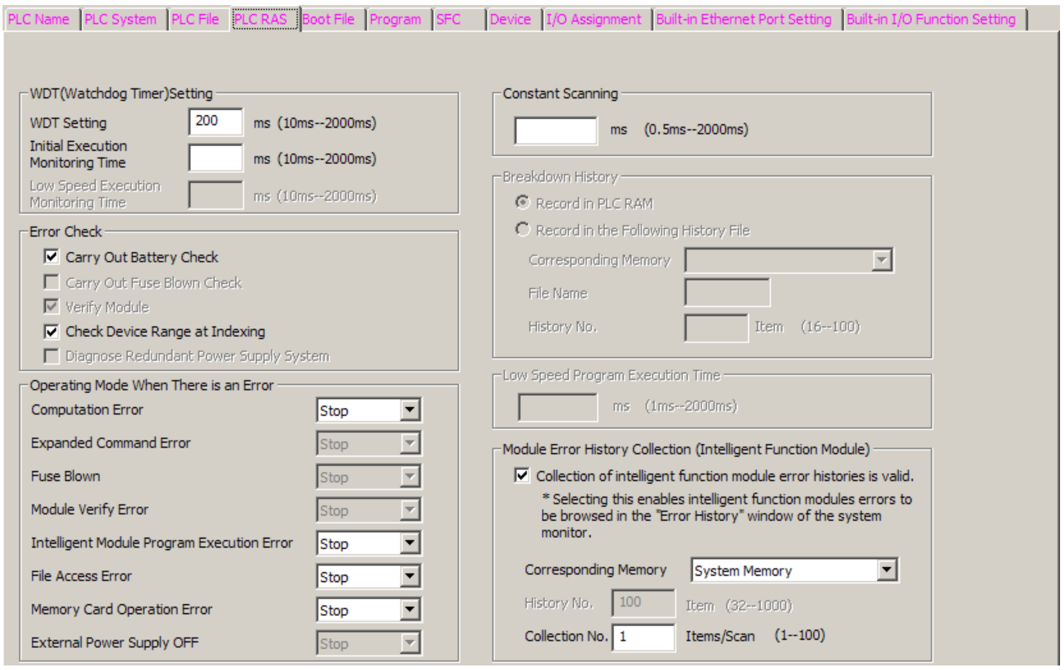
File register allocates memory for storage of file registers either on the internal Standard RAM drive. L Series defaults to 128K of data registers in a file called MAIN. There is a check box to include file registers in the latch data backup discussed on the last tab.

Comments used in command allocates storage location for comments which will be accessed by commands within the CPU. Initial device value sets a location to store initial values to be loaded into data registers at power-up. File for local devices is used when local devices are configured to store the multiple instances of the address. File used for SP.DEVST/S.DEVLD sets a file to be used by these two special commands in the CPU.

Notes

9.5 PLC RAS

RAS is an abbreviation for Reliability, Accessibility, and Serviceability. This tab configures the PLC watchdog timer and error handling.



WDT(Watchdog Timer)Setting
WDT Setting 200 ms (10ms--2000ms)
Initial Execution Monitoring Time ms (10ms--2000ms)
Low Speed Execution Monitoring Time ms (10ms--2000ms)

Error Check
☑ Carry Out Battery Check
☐ Carry Out Fuse Blown Check
☑ Verify Module
☑ Check Device Range at Indexing
☐ Diagnose Redundant Power Supply System

Operating Mode When There is an Error

Computation Error	Stop
Expanded Command Error	Stop
Fuse Blown	Stop
Module Verify Error	Stop
Intelligent Module Program Execution Error	Stop
File Access Error	Stop
Memory Card Operation Error	Stop
External Power Supply OFF	Stop

Constant Scanning
ms (0.5ms--2000ms)

Breakdown History
● Record in PLC RAM
○ Record in the Following History File
Corresponding Memory
File Name
History No. Item (16--100)

Low Speed Program Execution Time
ms (1ms--2000ms)

Module Error History Collection (Intelligent Function Module)
☑ Collection of intelligent function module error histories is valid.
* Selecting this enables intelligent function modules errors to be browsed in the "Error History" window of the system monitor.
Corresponding Memory System Memory
History No. 100 Item (32--1000)
Collection No. 1 Items/Scan (1--100)

Watchdog timers are used to warn if the processing of the program is taking too long. There are different watchdog timers for standard scan and initial scan programs.

Operating mode when there is an error sets whether the CPU will continue or stop in the event of several different types of CPU errors.

Error check will allow certain types of error checking to be turned off completely.

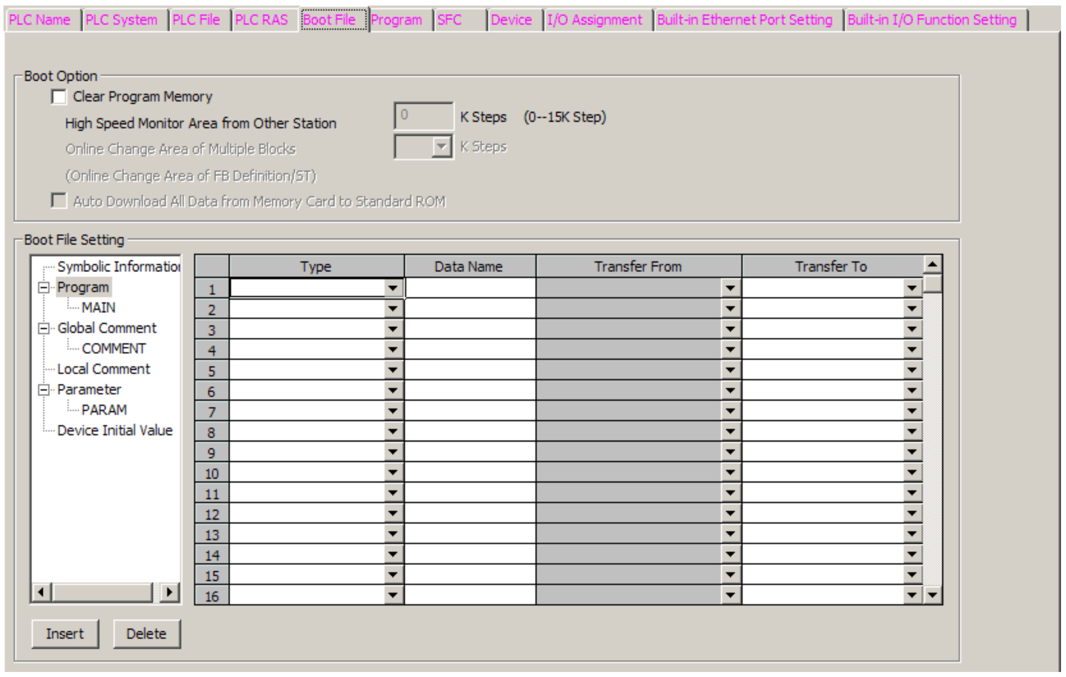
Constant scan sets a fixed execution time for each PLC scan.

Module error history collection will allow the errors detected by the intelligent function modules to be browsed in the PLC's main error history.

Notes

9.6 Boot file

The boot file tab is used to set the locations to copy various components of the CPU configuration and program at startup. This is typically used when the data is stored on a separate memory card or in the Standard ROM drive.

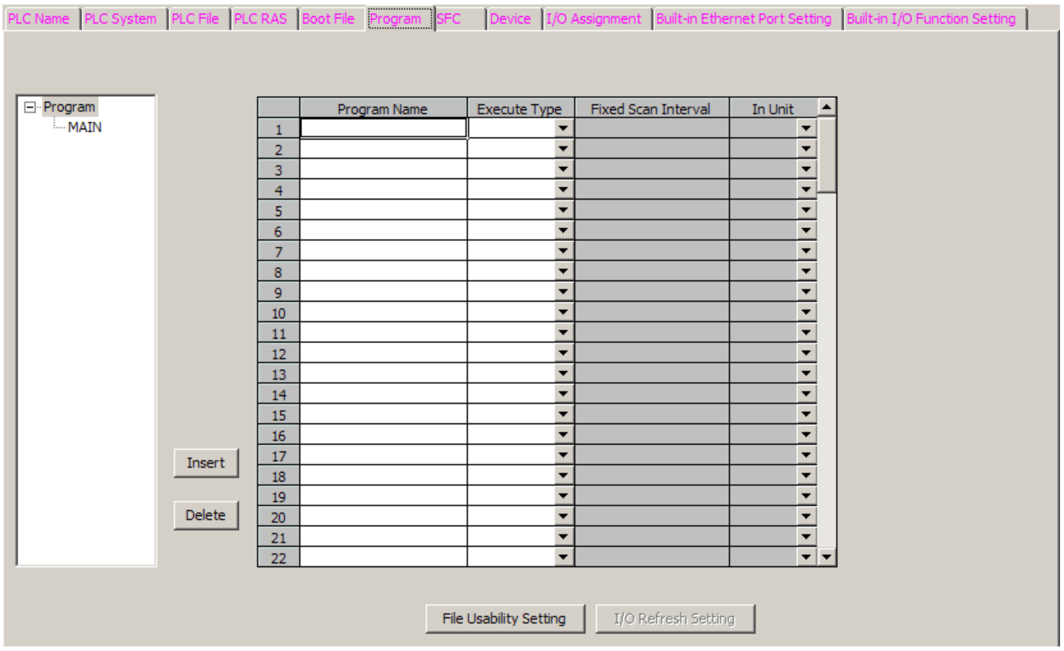


Each item to be copied to the CPU’s memory is added to this list. When the PLC is powered up and a boot file exists on the memory card or Standard ROM drive, the settings made there will be followed prior to starting the CPU into the RUN mode.

Notes

9.7 Program

The program tab sets up the program type and execution order when using multiple programs. With GX Works2, this can also be done by drag and drop of the programs into the Program Setting section of the project navigation window.



	Program Name	Execute Type	Fixed Scan Interval	In Unit
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				
21				
22				

9.8 SFC

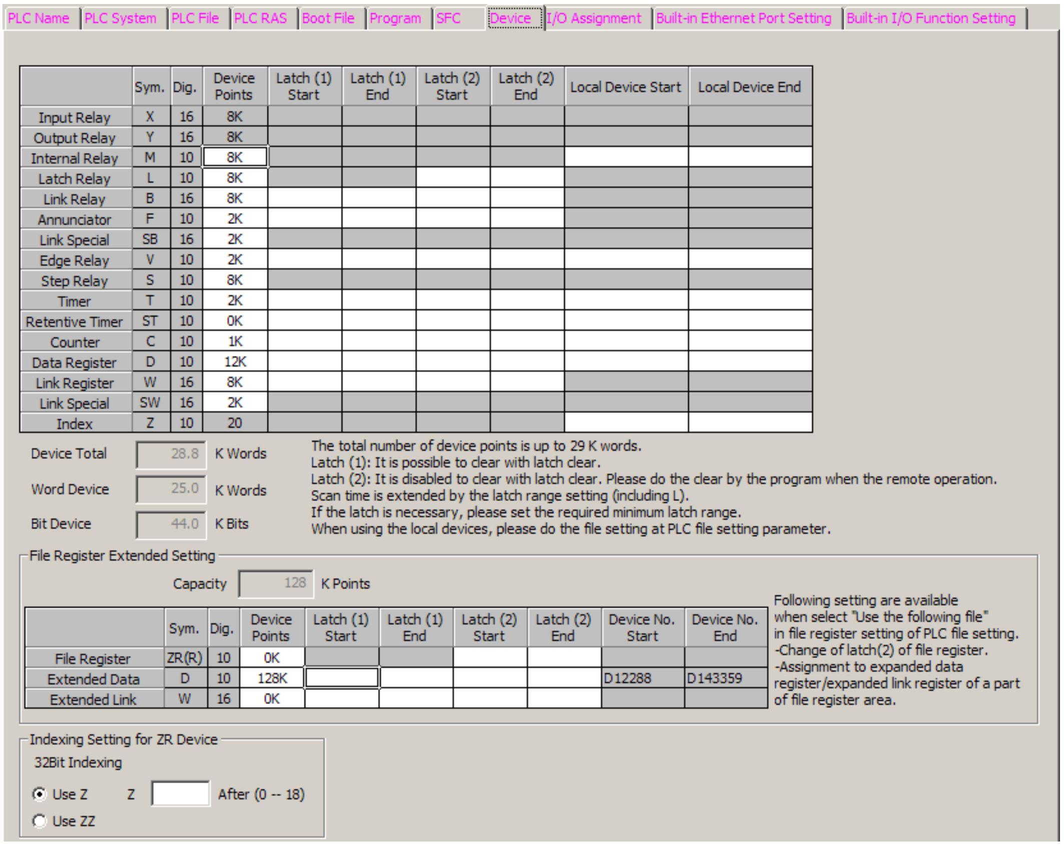
The parameters on the SFC tab are used to configure basic operation of programs written in the Sequential Function Chart programming language.

SFC programming is covered in the iQ Works Structured Programming class. It is outside the scope of this class.

Notes

9.9 Device

The Device parameters tab allocates memory to various data types in the CPU, and sets the battery backed ranges of addresses.



PLC Name | PLC System | PLC File | PLC RAS | Boot File | Program | SFC | Device | I/O Assignment | Built-in Ethernet Port Setting | Built-in I/O Function Setting

	Sym.	Dig.	Device Points	Latch (1) Start	Latch (1) End	Latch (2) Start	Latch (2) End	Local Device Start	Local Device End
Input Relay	X	16	8K						
Output Relay	Y	16	8K						
Internal Relay	M	10	8K						
Latch Relay	L	10	8K						
Link Relay	B	16	8K						
Annunciator	F	10	2K						
Link Special	SB	16	2K						
Edge Relay	V	10	2K						
Step Relay	S	10	8K						
Timer	T	10	2K						
Retentive Timer	ST	10	0K						
Counter	C	10	1K						
Data Register	D	10	12K						
Link Register	W	16	8K						
Link Special	SW	16	2K						
Index	Z	10	20						

Device Total 28.8 K Words

Word Device 25.0 K Words

Bit Device 44.0 K Bits

The total number of device points is up to 29 K words.
Latch (1): It is possible to clear with latch clear.
Latch (2): It is disabled to clear with latch clear. Please do the clear by the program when the remote operation.
Scan time is extended by the latch range setting (including L).
If the latch is necessary, please set the required minimum latch range.
When using the local devices, please do the file setting at PLC file setting parameter.

File Register Extended Setting

Capacity 128 K Points

	Sym.	Dig.	Device Points	Latch (1) Start	Latch (1) End	Latch (2) Start	Latch (2) End	Device No. Start	Device No. End
File Register	ZR(R)	10	0K						
Extended Data	D	10	128K					D12288	D143359
Extended Link	W	16	0K						

Following setting are available when select "Use the following file" in file register setting of PLC file setting.
-Change of latch(2) of file register.
-Assignment to expanded data register/expanded link register of a part of file register area.

Indexing Setting for ZR Device

32Bit Indexing

[✓ Use Z] Z After (0 -- 18)

Use ZZ

The dev point column configures the quantity for each type of data memory. This number is set in single numbers or by K (1024) addresses. The number of addresses configured for each memory type is adjustable. Below the table, the window shows the total memory used. This value must be 29K or less. It may be required to lower one address quantity before raising another to maintain this 29K limit.

Notes

Latch 1 and Latch 2 start and end set the first and last addresses in a range which are backed up by the battery. Latch 1 range can be cleared with the latch clear function, while Latch 2 cannot be cleared.

The local device starting and ending addresses set a range of addresses which will be unique to each program. For example, if 0 and 10 are entered in the D row, D0 through D10 will be unique within each program, so values written to D0 in one program have no effect on D0 in another program. This is useful to set local addresses when multiple programmers will write different sections of the program. It will allow a range of addresses to be specified which have no effect on the other programs.

The bottom half of the Device tab is used to allocate the file registers to various memory addresses. If file registers are allocated and configured on the PLC File tab, this section will allow distribution of that memory across D, R, and W addresses. The default allocation of the 128k of file registers is that file registers become addresses D12288 through D143359. The 128k can be divided into R registers, W registers, and D registers as the programmer sees fit.

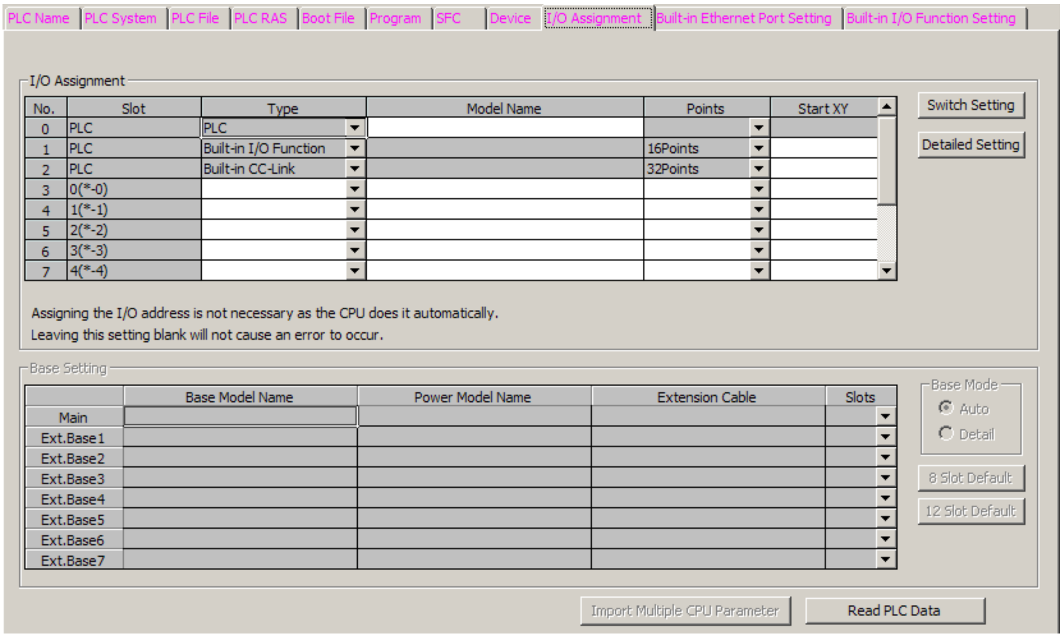
Notes

9.10 I/O Assignment

The I/O assignment tab serves many purposes. On this tab, the allocation of the modules can be configured. Other settings such as input filter, output mode at stop, and software switch settings for intelligent modules can be configured.

Settings are not required by default, as the CPU will automatically read the connected modules and allocate addressing at power-up. In order to configure settings such as input filter or software switch settings for intelligent modules, this information needs to be configured.

There is a button at the bottom to read the currently connected controller's I/O and configure the table to match this information.



I/O Assignment

No.	Slot	Type	Model Name	Points	Start XY
0	PLC	PLC			
1	PLC	Built-in I/O Function		16Points	
2	PLC	Built-in CC-Link		32Points	
3	0(*-0)				
4	1(*-1)				
5	2(*-2)				
6	3(*-3)				
7	4(*-4)				

Assigning the I/O address is not necessary as the CPU does it automatically.
Leaving this setting blank will not cause an error to occur.

Base Setting

	Base Model Name	Power Model Name	Extension Cable	Slots
Main				
Ext.Base1				
Ext.Base2				
Ext.Base3				
Ext.Base4				
Ext.Base5				
Ext.Base6				
Ext.Base7				

Base Mode: [✓ Auto] Detail

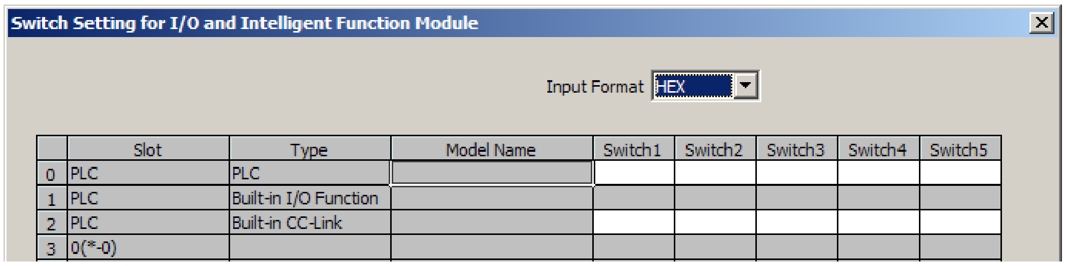
Notice on the L26CPU-BT (shown above) 2 modules have already been allocated. The first module is the built-in I/O points, and the second is the built-in CC-Link module.

Notes

The top half of this window is used to configure the module types and their starting addresses.

- The type column allows choice of module type. Choices are empty, input, output, and intelligent.
- The model name column is for documentation only. Typically the part number of the module would be entered here.
- The Points tab sets how many addresses are allocated to a module.
- The StartXY column sets the head address of each card in the base units. This setting is not required, but does allow the L Series to apply addresses out of order.

Clicking the Switch Setting button will bring up access to the software switch settings page. This information is a software replacement for the various DIP switches which used to exist on the front of many of the modules in the A Series. Instead, software settings are made and stored in the CPU, which makes replacement of a failed module much easier. All switch settings are stored in the parameters which are stored in the CPU.



Input Format HEX

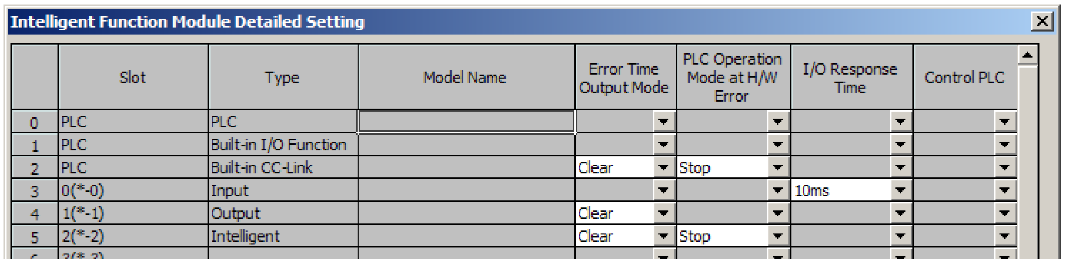
	Slot	Type	Model Name	Switch1	Switch2	Switch3	Switch4	Switch5
0	PLC	PLC						
1	PLC	Built-in I/O Function						
2	PLC	Built-in CC-Link						
3	0(*-0)							

Any module configured as intelligent will have 5 software switches, each of which is a 4-digit hexadecimal value. Modules which are not intelligent will be grayed out. The setting for each switch and their purpose varies by module, so it is important to have the documentation for the intelligent modules available when configuring these switch settings.

These switch settings can also be made in the intelligent module utility.

Notes

The Detailed setting button allows for configuration of the discrete I/O modules.



Intelligent Function Module Detailed Setting

	Slot	Type	Model Name	Error Time Output Mode	PLC Operation Mode at H/W Error	I/O Response Time	Control PLC
0	PLC	PLC					
1	PLC	Built-in I/O Function					
2	PLC	Built-in CC-Link		Clear	Stop		
3	0(*-0)	Input				10ms	
4	1(*-1)	Output		Clear			
5	2(*-2)	Intelligent		Clear	Stop		

For input modules, the input response time can be configured. This value is independently configured for each input card.

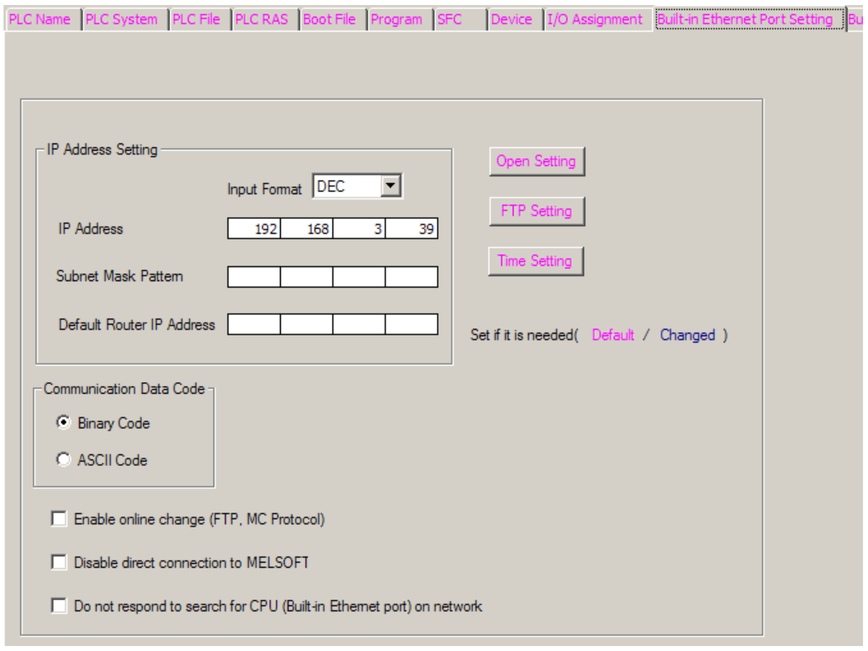
Output modules allow the setting of the status of the outputs, which allows the CPU to leave certain outputs on in the event of a CPU error.

Intelligent modules also offer error time output setting. An additional option for hardware error PLC operation allows the programmer to determine if a hardware error detected in this module will stop the processor or allow it to continue.

Notes

9.11 Built-In Ethernet

With the L Series processors, there is another tab for configuration of the built-in Ethernet port.



IP Address Setting
Input Format DEC
IP Address 192 168 3 39
Subnet Mask Pattern
Default Router IP Address
Set if it is needed(Default / Changed)

Communication Data Code
(●) Binary Code
( ) ASCII Code

☐ Enable online change (FTP, MC Protocol)
☐ Disable direct connection to MELSOFT
☐ Do not respond to search for CPU (Built-in Ethernet port) on network

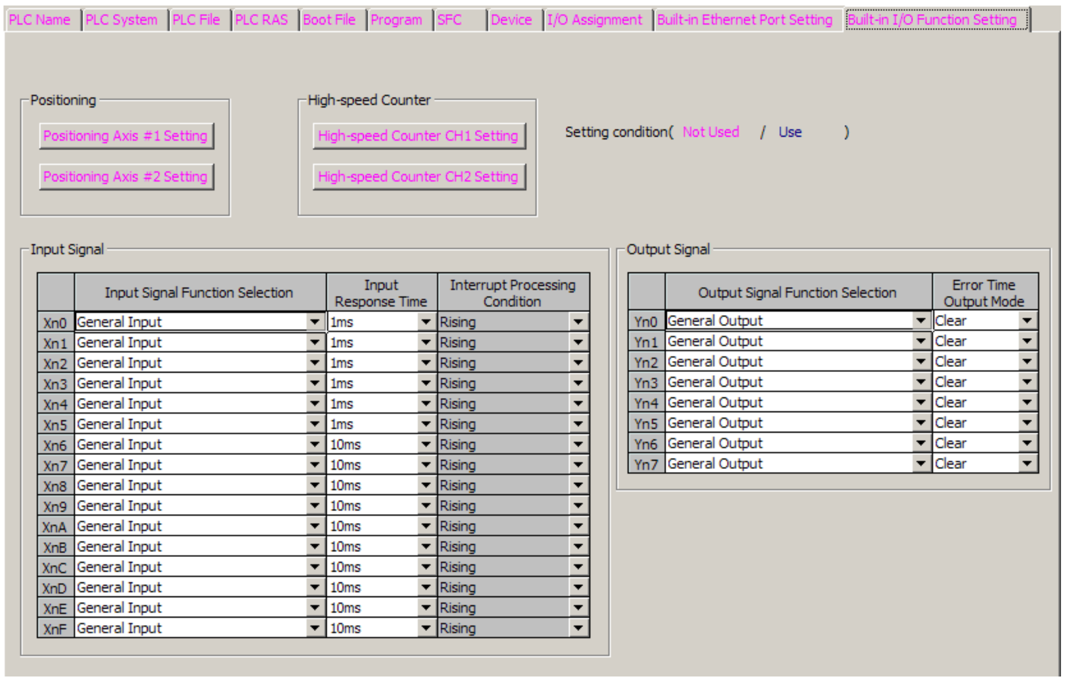
On this tab, the standard Ethernet settings like IP address, subnet mask, and default gateway are configured. There are check boxes to allow online changes, disable the direct connect to MELSOFT, and not respond to processor searches on the network.

There are also buttons for configuring the open port settings, FTP access settings, and SNTP time protocol configuration.

Notes

9.12 Built-In I/O Functions

The L Series controller has 16 inputs and 8 outputs built in. These can be configured for special functions in the Built-In I/O Function tab.


PLC Name
PLC System
PLC File
PLC RAS
Boot File
Program
SFC
Device
I/O Assignment
Built-in Ethernet Port Setting
Built-in I/O Function Setting
Positioning
Positioning Axis #1 Setting
Positioning Axis #2 Setting
High-speed Counter
High-speed Counter CH1 Setting
High-speed Counter CH2 Setting

Setting condition(Not Used / Use)

Input Signal

	Input Signal Function Selection	Input Response Time	Interrupt Processing Condition
Xn0	General Input	1ms	Rising
Xn1	General Input	1ms	Rising
Xn2	General Input	1ms	Rising
Xn3	General Input	1ms	Rising
Xn4	General Input	1ms	Rising
Xn5	General Input	1ms	Rising
Xn6	General Input	10ms	Rising
Xn7	General Input	10ms	Rising
Xn8	General Input	10ms	Rising
Xn9	General Input	10ms	Rising
XnA	General Input	10ms	Rising
XnB	General Input	10ms	Rising
XnC	General Input	10ms	Rising
XnD	General Input	10ms	Rising
XnE	General Input	10ms	Rising
XnF	General Input	10ms	Rising

Output Signal

	Output Signal Function Selection	Error Time Output Mode
Yn0	General Output	Clear
Yn1	General Output	Clear
Yn2	General Output	Clear
Yn3	General Output	Clear
Yn4	General Output	Clear
Yn5	General Output	Clear
Yn6	General Output	Clear
Yn7	General Output	Clear

On the top left are 2 buttons to configure the 2 axes of built-in pulse output positioning control. Next to those are 2 buttons to configure the 2 channels of high speed pulse input.

The bottom half of this window can be used to allocate specific functions to specific inputs or outputs. The choices are limited based on each address, and many require settings to be made using the 4 buttons above to be valid.

Inputs can be configured for general, interrupt, or for signals related to the high speed counter inputs or pulse outputs.

Outputs can be configured for general, or for signals related to the high speed counter inputs or pulse outputs.

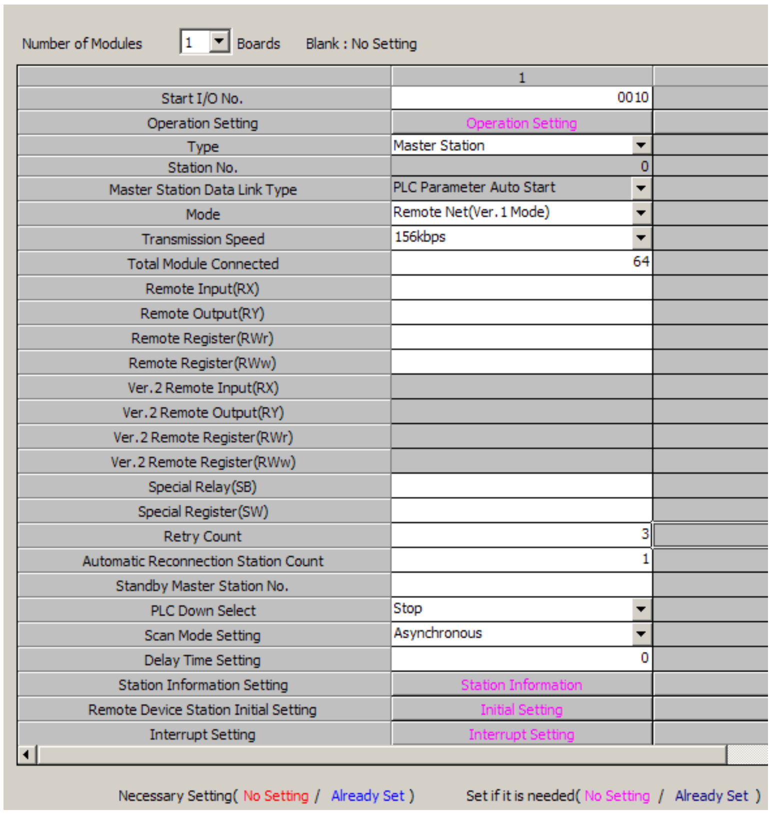
Notes

9.13 Network Parameters

With the L Series, CC-Link networks are configured with parameter settings. This includes the built-in CC-Link module of the L26CPU-BT.

Network parameters are found in the Parameters section of the project tree, directly under the PLC parameters.

The screen shown below is for CC-Link. Each column is used for an additional CC-Link module. Up to 2 cards can be configured on L02CPU, with 3 cards plus the built-in CC-Link available on the L26CPU-BT.



Number of Modules 1 Boards Blank : No Setting

	1
Start I/O No.	0010
Operation Setting	Operation Setting
Type	Master Station
Station No.	0
Master Station Data Link Type	PLC Parameter Auto Start
Mode	Remote Net(Ver.1 Mode)
Transmission Speed	156kbps
Total Module Connected	64
Remote Input(RX)	
Remote Output(RY)	
Remote Register(RWr)	
Remote Register(RWw)	
Ver.2 Remote Input(RX)	
Ver.2 Remote Output(RY)	
Ver.2 Remote Register(RWr)	
Ver.2 Remote Register(RWw)	
Special Relay(SB)	
Special Register(SW)	
Retry Count	3
Automatic Reconnection Station Count	1
Standby Master Station No.	
PLC Down Select	Stop
Scan Mode Setting	Asynchronous
Delay Time Setting	0
Station Information Setting	Station Information
Remote Device Station Initial Setting	Initial Setting
Interrupt Setting	Interrupt Setting

Necessary Setting(No Setting / Already Set) Set if it is needed(No Setting / Already Set)

Configuration and operation of CC-Link are covered in the CC-Link Networking training class.

Notes

LESSON 10 – Label Programming

This lesson will cover the basics of label programming in GX Works2.

Lesson Objectives

At the conclusion of this lesson, you will be able to…

- Understand the difference between global, local, and system labels.
- Assign labels and write programs using labels.

10.1 What are Labels?

Labels are a method of assigning a name to an address or variable in the controller. This label can be used in place of the physical address when writing code. Assignment of labels is required when using the structured text programming language, and is optional in the other languages.

There are several types of labels in GX Works2. The 2 most common types are global and local labels.

- Global labels are registered in the Global Label list in the project workspace, and apply to every program created within the project.
- Local labels are registered under the program in the POU section of the project workspace, and apply only within the specific program.

Labels are also used with functions and function blocks, and for creating structured data types.

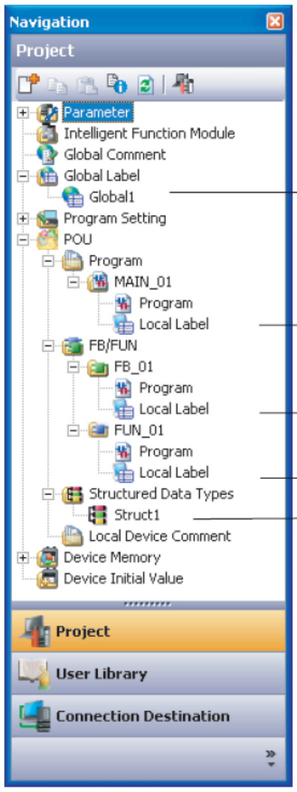
- Function block labels are used inside function blocks, and identify the input and output connections on the function block.
- Data structures, such as arrays and structured data types, can be created by using labels.

More information on labels can be found in Chapter 5 of the GX Works2 Operating Manual (Structured Project).

Labels are also sometimes referred to by other vendors as tags or variables.

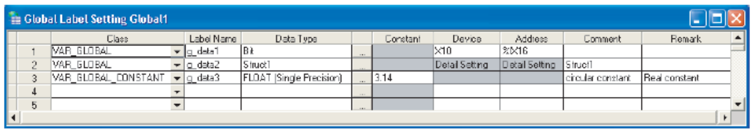
Notes

The labels are configured through the following sections of the navigation window.



Global Label Setting screen

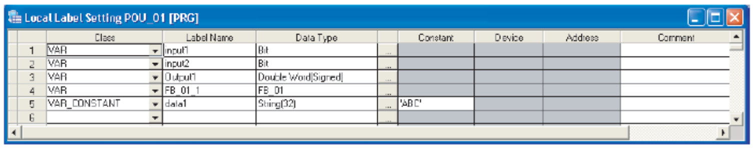
Define labels that can be used for all POUs in the project.


Global Label Setting Global1

	Class	Label Name	Data Type		Constant	Device	Address	Comment	Remark
1	VAR_GLOBAL	g_data1	Bit	...		X10	%IX16		
2	VAR_GLOBAL	g_data2	Struct1	...		Detail Setting	Detail Setting	Struct1	
3	VAR_GLOBAL_CONSTANT	g_data3	FLOAT (Single Precision)	...	3.14			circular constant	Real constant
4				...					
5				...					

Local Label Setting screen

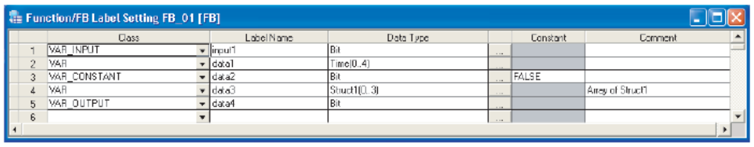
Define labels that can be used only for each POU (program block).


Local Label Setting POU_01 [PRG]

	Class	Label Name	Data Type		Constant	Device	Address	Comment
1	VAR	input1	Bit	...				
2	VAR	input2	Bit	...				
3	VAR	Output1	Double Word[Signed]	...				
4	VAR	FB_01_1	FB_01	...				
5	VAR_CONSTANT	data1	String(32)	...	'ABC'			
6				...				

Function/FB Label Setting screen

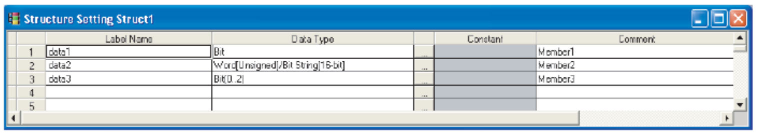
Define labels that can be used only for each POU (function/function block).


Function/FB Label Setting FB_01 [FB]

	Class	Label Name	Data Type		Constant	Comment
1	VAR_INPUT	input1	Bit	...		
2	VAR	data1	Time(0..4)	...		
3	VAR_CONSTANT	data2	Bit	...	FALSE	
4	VAR	data3	Struct1(0..3)	...		Array of Struct1
5	VAR_OUTPUT	data4	Bit	...		
6				...		

Structure Setting screen

Define structured data types to be used for labels.


Structure Setting Struct1

	Label Name	Data Type		Constant	Comment
1	data1	Bit	...		Member1
2	data2	Word[Unsigned]/Bit String[16-bit]	...		Member2
3	data3	Bit(0..2)	...		Member3
4			...		
5			...		

Notes

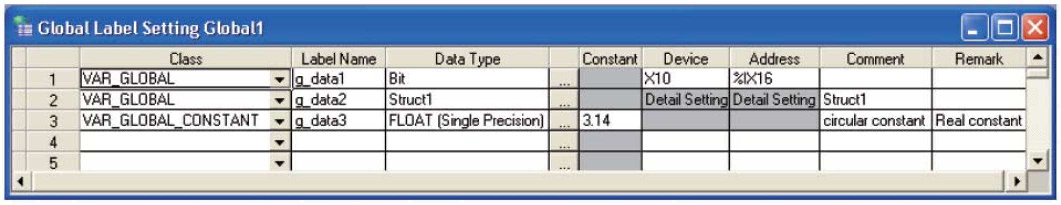
Labels can be up to 32 characters long. Some basic rules to follow are:

- Labels cannot include spaces
- Labels cannot begin with a number
- Labels cannot be the same as device numbers, such as M1
- The following characters are not allowed in labels
  / \ * ? < > | " : [ ] ; , = + % ' ~ @ { } ! # $ &

A complete list of unusable labels can be found in Appendix 9 of the GX Works2 Operating Manual (Common).

10.2 Registering Global Labels

Global labels are registered in the Global Label section of the navigation window.


Global Label Setting Global1

	Class	Label Name	Data Type		Constant	Device	Address	Comment	Remark
1	VAR_GLOBAL	g_data1	Bit	...		X10	%IX16		
2	VAR_GLOBAL	g_data2	Struct1	...		Detail Setting	Detail Setting	Struct1	
3	VAR_GLOBAL_CONSTANT	g_data3	FLOAT (Single Precision)	...	3.14			circular constant	Real constant
4				...					
5				...					

There are 2 classes of label which can be registered.

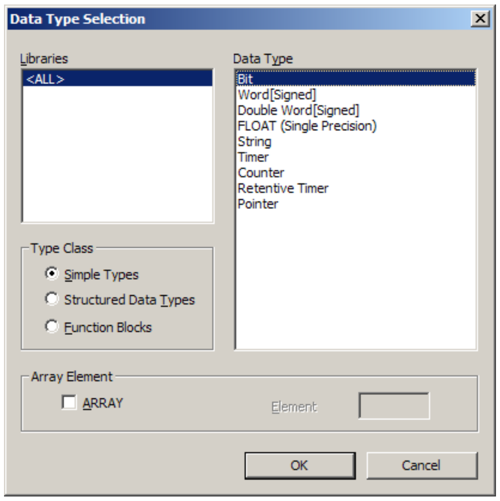
- VAR_GLOBAL items are common labels which can be used in all programs and reference an address in the PLC memory.
- VAR_GLOBAL_CONSTANT is a label specified for a constant number.

After the class is chosen, the label name is entered, following the rules mentioned before.

Using the right click menu or the Edit menu, options 'New Declaration (Before)' or 'New Declaration (After)' allow entries to be added before or after the currently selected label. New Declaration (After) will increment the label and address and set all other settings identical to the selected label. This is handy for creating a series of consecutive label names.

Notes

The data type can be selected from a list by pressing the […] button to the right of the data type column. The dialog window below allows the selection of a simple data type, structured data type, or function block data type.


Data Type Selection

Libraries: <ALL>

Data Type: Bit, Word[Signed], Double Word[Signed], FLOAT (Single Precision), String, Timer, Counter, Retentive Timer, Pointer

Type Class: [✓ Simple Types] Structured Data Types Function Blocks

Array Element: ARRAY Element

Structured data types and function block data types are displayed based on the subjects configured within the open project.

If the VAR_GLOBAL_CONSTANT class was selected, the next column is used to set the constant value for the label.

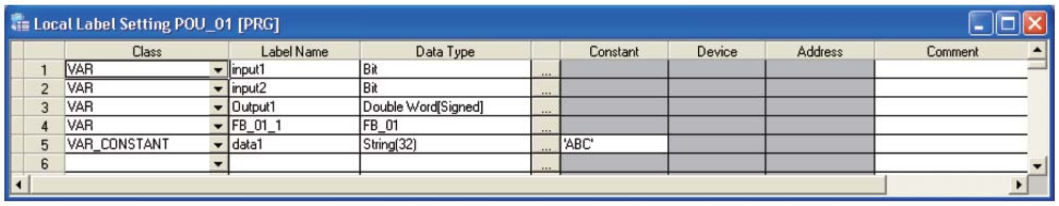
If the VAR_GLOBAL class was selected, the next 2 columns can be used to set the address associated with the label. In the Device column, a Mitsubishi address can be defined. In the Address column, and IEC compliant address can be defined. The Address column only exists in structured projects. Only one of the columns needs to be filled in, the other will be filled automatically. These fields can be left blank and an address will be assigned automatically. If the data type is a structure, these columns will contain buttons labeled ‘Detail Setting’ which will display a configuration window when clicked with the mouse.

The last 2 columns are for documentation. Comments can be displayed with the addresses in the program window, just as they were in GX Developer. Remarks are additional documentation space which can be up to 1024 characters.

Notes

10.3 Registering Local Labels

Local labels will exist only within the program where they are created. They are registered in the Local Label list inside the program in the POU section of the navigation window.



Local Label Setting POU_01 [PRG]

	Class	Label Name	Data Type		Constant	Device	Address	Comment
1	VAR	input1	Bit	...				
2	VAR	input2	Bit	...				
3	VAR	Output1	Double Word[Signed]	...				
4	VAR	FB_01_1	FB_01	...				
5	VAR_CONSTANT	data1	String(32)	...	'ABC'			
6				...				

The columns and options are very similar to the global labels.

The 3 classes available for local labels are:

- VAR for a label to reference a PLC address
- VAR_CONSTANT for a constant numerical value
- VAR_RETAIN for a label which is latched
    - VAR_RETAIN is not supported on the FX controllers.

The label name, data type, and constant fields are the same as discussed with the global labels. Local labels are not defined addresses, so the device and address columns are not available. Local labels support comments, but do not have a remark column.

Notes

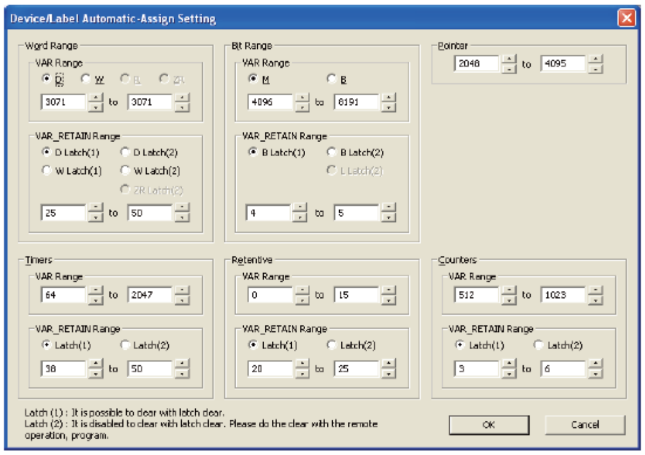
10.4 Automatic Assignment

Labels can be automatically assigned addresses, instead of entering a PLC address on each label. When the label's device and address fields are left blank, they will be automatically assigned a PLC address from a configured pool of reserved addresses.

The reserved addresses for automatic label assignment can be adjusted by the programmer in the project. From the 'Tool' menu, select 'Device/Label Automatic-Assign Setting'.

The window which is displayed will vary based on which family of PLC was selected.

< QCPU (Q mode) >



Device/Label Automatic-Assign Setting

Word Range
- VAR Range: [✓ D] W R ZR; 3071 to 3071
- VAR_RETAIN Range: [✓ D Latch(1)] D Latch(2) W Latch(1) W Latch(2) ZR Latch(2); 25 to 50

Bit Range
- VAR Range: [✓ M] B; 4096 to 8191
- VAR_RETAIN Range: [✓ B Latch(1)] B Latch(2) L Latch(2); 4 to 5

Pointer: 2048 to 4095

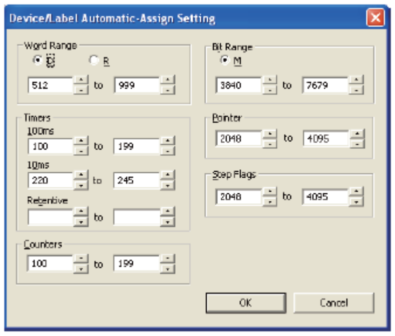
Timers
- VAR Range: 64 to 2047
- VAR_RETAIN Range: [✓ Latch(1)] Latch(2); 38 to 50

Retentive
- VAR Range: 0 to 15
- VAR_RETAIN Range: [✓ Latch(1)] Latch(2); 20 to 25

Counters
- VAR Range: 512 to 1023
- VAR_RETAIN Range: [✓ Latch(1)] Latch(2); 5 to 6

Latch (1) : It is possible to clear with latch clear.
Latch (2) : It is disabled to clear with latch clear. Please do the clear with the remote operation, program.

< FXCPU >



Device/Label Automatic-Assign Setting

Word Range: [✓ D] R; 512 to 999

Bit Range: [✓ M]; 3840 to 7679

Timers
- 100ms: 100 to 199
- 10ms: 220 to 245
- Retentive: to

Pointer: 2048 to 4095

Step Flags: 2048 to 4095

Counters: 100 to 199

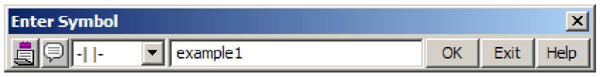
By default, the upper portion of the device memory is reserved for the automatic assignment. These values can be modified for each of the displayed address types.

Notice that in the RETAIN areas, most memory areas are listed twice, with a (1) or (2) after them. This pertains to the Latch(1) and Latch(2) ranges as specified in the PLC parameters on the Device tab.

Notes

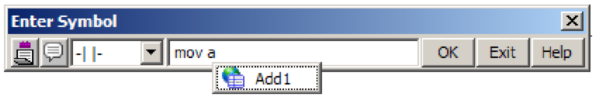
10.5 Using Labels

To write the program with labels, simply code the label name instead of an address when configuring symbols in the ladder logic.

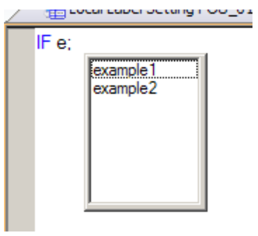


example1

In ladder and structured text programs, labels will have an auto-complete selection window appear as the name is entered, and the label can be selected from this list.

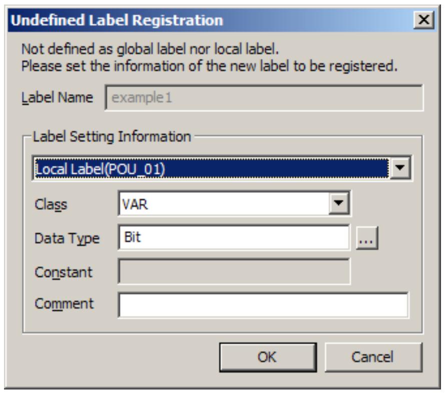


mov a
Add1



IF e;
example1
example2

If a label is entered which is not defined, a dialog window like the one below will appear allowing the programmer to assign that label. It can be assigned as a local label in the current program, or as a global label.



Not defined as global label nor local label.
Please set the information of the new label to be registered.

Label Name: example1

Label Setting Information: Local Label(POU_01)

Class: VAR

Data Type: Bit

Constant:

Comment:

Notes

10.6 EXERCISE – Global Labels

Create a new simple project, and select to use labels.

Add the following entries to the global label list:

- Bits
  - Start_Button (M0)
  - Stop_Button (M1)
  - Good_Part (M2)
  - Bad_Part (M3)
  - Total_Reset (M4)
  - Run_Lamp (Y0)
  - Conveyor (Y1)
  - Reject_Gate (Y2)
- Words
  - Good_Count (D0)
  - Bad_Count (D1)

When the start button is pressed, the system will begin to operate, and the run lamp and conveyor will turn on. The stop button will shut the system off.

Sensors Good_Part and Bad_Part are used to indicate a good or bad part on the conveyor.

When a good part is detected, increment the good product count.

When a bad part is detected, increment the bad part count, and turn on the reject gate for 3 seconds.

When the reset totals switch is pressed, reset the good and bad product counts.

NOTE: Addresses are only specified so the GOT screens will operate properly.

Notes

LESSON 11 – Structured Projects

This lesson will introduce the concepts involved with structured projects.

Lesson Objectives

At the conclusion of this lesson, you will be able to…

- Understand what IEC 61131-3 does and does not define.
- Identify the different components of the structured project.

11.1 IEC 61131-3

IEC 61131-3 is an international standard related to PLC programming. It was defined by the International Electromechanical Commission (IEC). It defines the programming languages and structured elements used for writing PLC programs.

One of the basic goals of IEC 61131-3 is to create a uniform program structure and naming convention for PLC programming. In this way, it is designed to allow a programmer who is familiar with one brand of controls to program another brand of controls with a very limited learning curve. A programming software which is compliant should have the same basic look and feel, regardless of which vendor's equipment the software is for.

It does not state that all software packages have to be identical, nor does it state that code written for one vendor’s product should be able to be copied and pasted into another vendor’s product. It simply lays out groundwork for a programming method which would be similar among all vendors who comply.

Structured projects offer a different way of managing the programs in the PLC than the simple projects.

Compliance with the standard is certified by PLCOpen. PLCOpen is an independent third party who tests compliance with the standards and certifies products as compliant to the standard. They also work to promote the use of IEC 61131-3 products.

Notes

The IEC 61131-3 standard defines 5 programming languages for users to create their programs.

- LD (Ladder Diagram)
- FBD (Function Block Diagram)
- SFC (Sequential Function Chart)
- ST (Structured Text)
- IL (Instruction List)

GX Works2 currently supports the LD, SFC, and ST languages.

The standards state that different components of a project can be written in any of the various languages, and that a program is not locked into a single programming language. The hierarchical structure of an IEC compliant project allows for the programmer to organize the components of an application in an intuitive format. It also makes the re-use of code easier.

The standards also allow for the assignment of names (labels or variables) to identify devices such as I/O and data registers, allowing those names to be used when writing code in place of the specific addresses.

A project is composed of several types of objects. These objects are referred to as programs, tasks, and POUs (program organizational units).

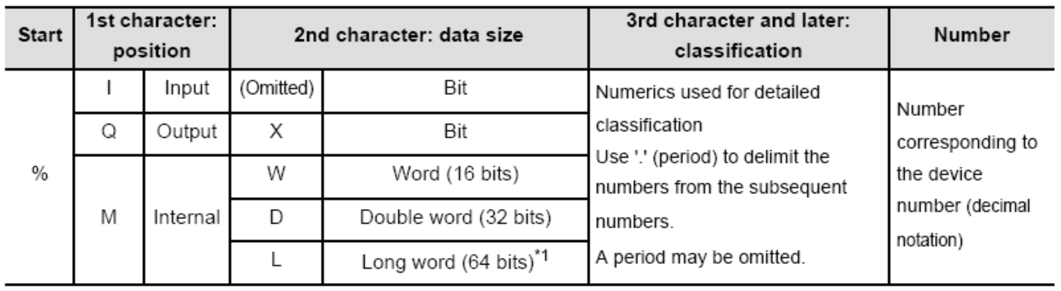
Each project can contain multiple POUs. These POUs will be contained in multiple tasks. Multiple tasks can be created and inserted into programs.

Notes

11.2 IEC Addresses

The IEC standard has a specified method for addressing of the PLC's various memory areas. The tables on the next 2 pages show the addresses in the standard Mitsubishi format and the IEC compliant format. GX Works2 will allow either type to be used, but use of the standard Mitsubishi style address makes a program not compliant with the IEC 61131-3 standard

The basic explanation of the IEC addresses is shown here.



Start	1st character: position		2nd character: data size		3rd character and later: classification	Number
%	I	Input	(Omitted)	Bit	Numerics used for detailed classification Use '.' (period) to delimit the numbers from the subsequent numbers. A period may be omitted.	Number corresponding to the device number (decimal notation)
	Q	Output	X	Bit		
	M	Internal	W	Word (16 bits)		
			D	Double word (32 bits)		
			L	Long word (64 bits)*1		

*1: Not supported with the FXCPU.

IEC addresses always begin with %. The next digit represents the memory area. %I indicates input addresses, while %Q means output addresses. %M indicates internal memory addresses.

The third digit indicates the size of the data. An X here means a bit address, a W means a word address, a D indicates 32-bit double words. L indicates 64-bit addresses, which are only available in the Universal model Q Series and L Series CPUs.

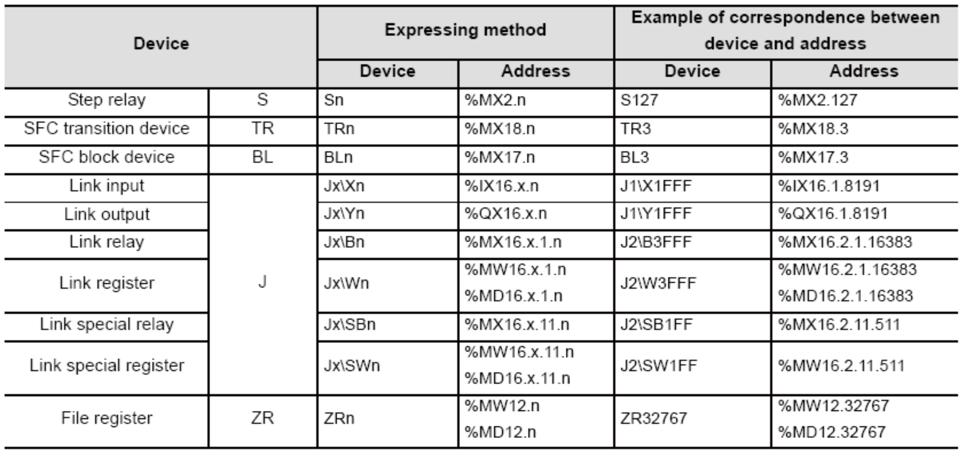
Note that inputs and outputs are both referenced with an X. This X does not indicate input like it does in standard Mitsubishi addresses. In IEC format, the X is indicating a bit level address.

Notes

The tables below shows the L Series addresses.

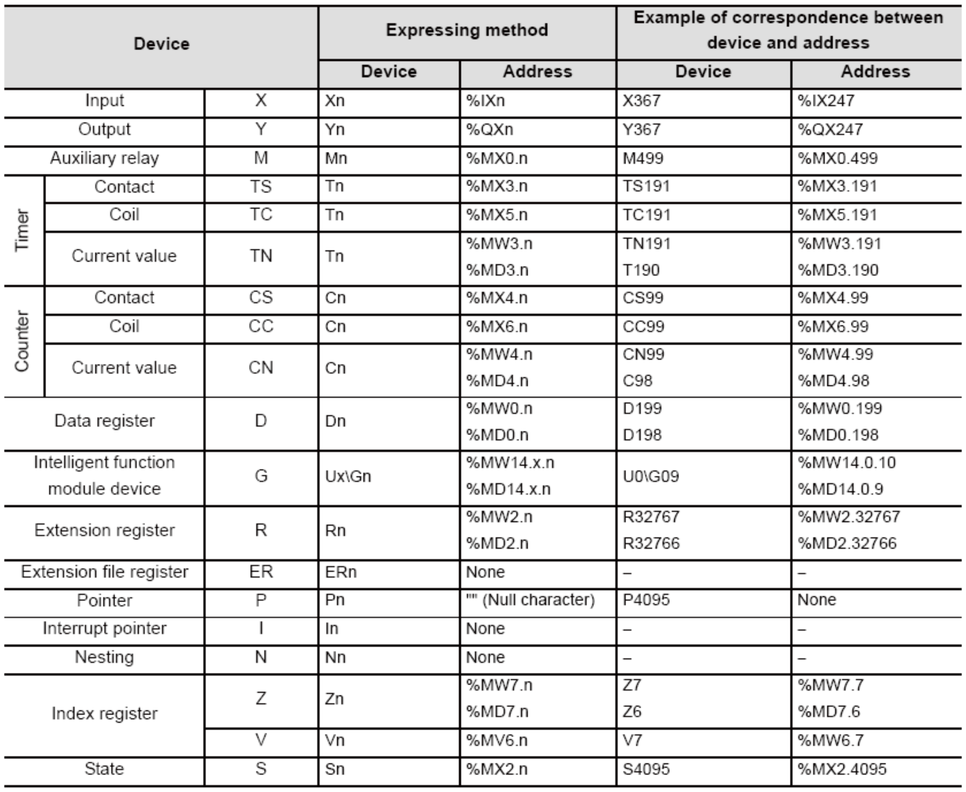
Device			Expressing method		Example of correspondence between device and address	
			Device	Address	Device	Address
Input		X	Xn	%IXn	X7FF	%IX2047
Output		Y	Yn	%QXn	Y7FF	%QX2047
Internal relay		M	Mn	%MX0.n	M2047	%MX0.2047
Latch relay		L	Ln	%MX8.n	L2047	%MX8.2047
Annunciator		F	Fn	%MX7.n	F1023	%MX7.1023
Special relay		SM	SMn	%MX10.n	SM1023	%MX10.1023
Function input		FX	FXn	None	FX10	None
Function output		FY	FYn	None	FY10	None
Edge relay		V	Vn	%MX9.n	V1023	%MX9.1023
Direct access input		DX	DXn	%IX1.n	DX7FF	%IX1.2047
Direct access output		DY	DYn	%QX1.n	DY7FF	%QX1.2047
Timer	Contact	TS	Tn	%MX3.n	TS511	%MX3.511
	Coil	TC	Tn	%MX5.n	TC511	%MX5.511
	Current value	TN	Tn	%MW3.n %MD3.n	TN511 T511	%MW3.511 %MD3.511
Counter	Contact	CS	Cn	%MX4.n	CS511	%MX4.511
	Coil	CC	Cn	%MX6.n	CC511	%MX6.511
	Current value	CN	Cn	%MW4.n %MD4.n	CN511 C511	%MW4.511 %MD4.511
Retentive timer	Contact	STS	STn	%MX13.n	STS511	%MX13.511
	Coil	STC	STn	%MX15.n	STC511	%MX15.511
	Current value	STN	STn	%MW13.n %MD13.n	STN511 ST511	%MW13.511 %MD13.511
Data register		D	Dn	%MW0.n %MD0.n	D11135	%MW0.11135 %MD0.11135
Special register		SD	SDn	%MW10.n %MD10.n	SD1023	%MW10.1023 %MD10.1023
Function register		FD	FDn	None	FD0	None
Link relay		B	Bn	%MX1.n	B7FF	%MX1.2047
Link special relay		SB	SBn	%MX11.n	SB3FF	%MX11.1023
Link register		W	Wn	%MW1.n %MD1.n	W7FF	%MW1.2047 %MD1.2047
Link special register		SW	SWn	%MW11.n %MD11.n	SW3FF	%MW11.1023 %MD11.1023
Intelligent function module device		G	Ux\Gn	%MW14.x.n %MD14.x.n	U0\G65535	%MW14.0.65535 %MD14.0.65535
File register		R	Rn	%MW2.n %MD2.n	R32767	%MW2.32767 %MD2.32767
Pointer		P	Pn	"" (Null character)	P299	None
Interrupt pointer		I	In	None	-	-
Nesting		N	Nn	None	-	-
Index register		Z	Zn	%MW7.n %MD7.n	Z9	%MW7.9 %MD7.9

Notes



Device		Expressing method		Example of correspondence between device and address	
		Device	Address	Device	Address
Step relay	S	Sn	%MX2.n	S127	%MX2.127
SFC transition device	TR	TRn	%MX18.n	TR3	%MX18.3
SFC block device	BL	BLn	%MX17.n	BL3	%MX17.3
Link input	J	Jx\Xn	%IX16.x.n	J1\X1FFF	%IX16.1.8191
Link output		Jx\Yn	%QX16.x.n	J1\Y1FFF	%QX16.1.8191
Link relay		Jx\Bn	%MX16.x.1.n	J2\B3FFF	%MX16.2.1.16383
Link register		Jx\Wn	%MW16.x.1.n %MD16.x.1.n	J2\W3FFF	%MW16.2.1.16383 %MD16.2.1.16383
Link special relay		Jx\SBn	%MX16.x.11.n	J2\SB1FF	%MX16.2.11.511
Link special register		Jx\SWn	%MW16.x.11.n %MD16.x.11.n	J2\SW1FF	%MW16.2.11.511
File register	ZR	ZRn	%MW12.n %MD12.n	ZR32767	%MW12.32767 %MD12.32767

The table below shows FX Series addresses.

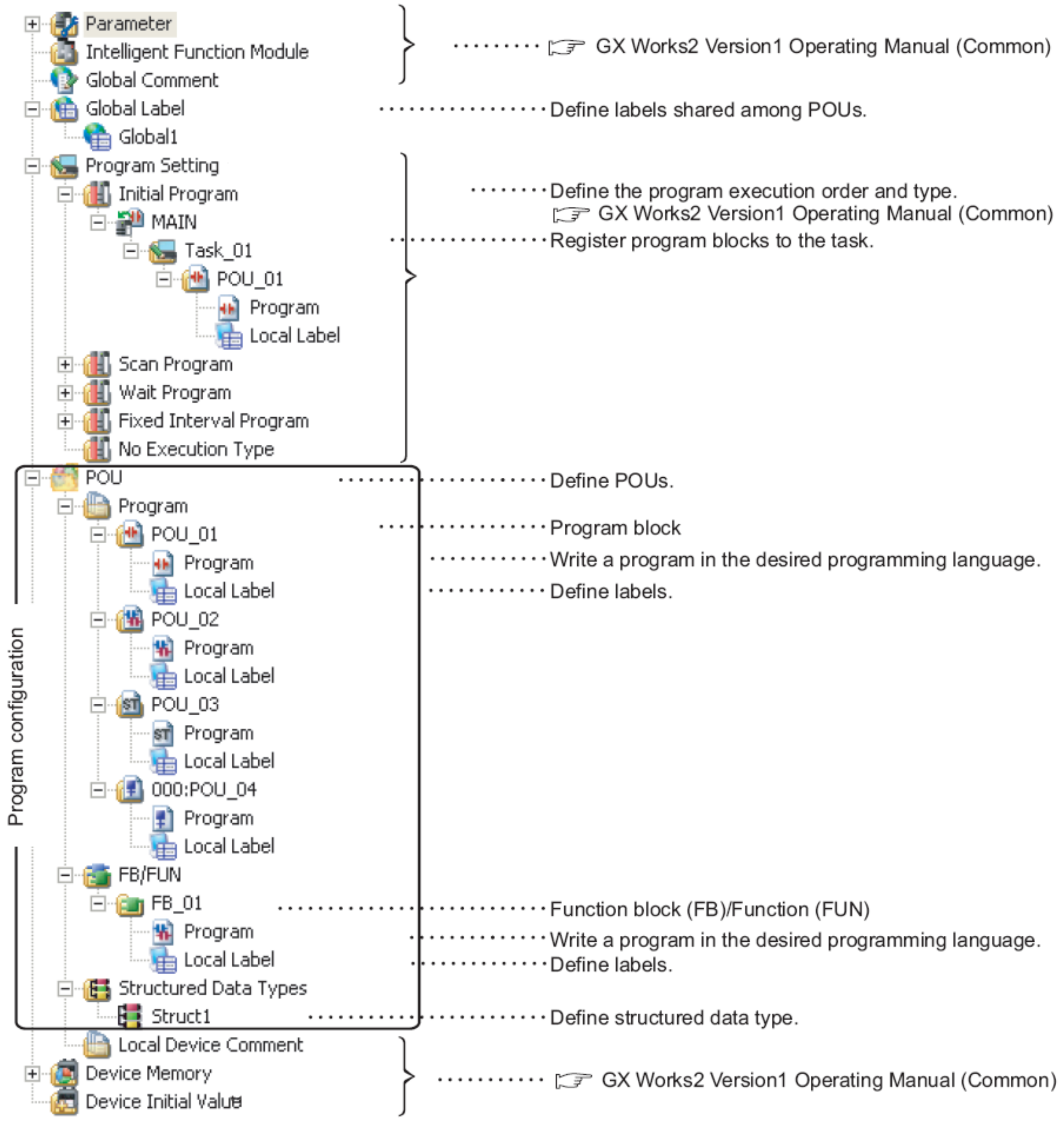


Device			Expressing method		Example of correspondence between device and address	
			Device	Address	Device	Address
Input		X	Xn	%IXn	X367	%IX247
Output		Y	Yn	%QXn	Y367	%QX247
Auxiliary relay		M	Mn	%MX0.n	M499	%MX0.499
Timer	Contact	TS	Tn	%MX3.n	TS191	%MX3.191
	Coil	TC	Tn	%MX5.n	TC191	%MX5.191
	Current value	TN	Tn	%MW3.n %MD3.n	TN191 T190	%MW3.191 %MD3.190
Counter	Contact	CS	Cn	%MX4.n	CS99	%MX4.99
	Coil	CC	Cn	%MX6.n	CC99	%MX6.99
	Current value	CN	Cn	%MW4.n %MD4.n	CN99 C98	%MW4.99 %MD4.98
Data register		D	Dn	%MW0.n %MD0.n	D199 D198	%MW0.199 %MD0.198
Intelligent function module device		G	Ux\Gn	%MW14.x.n %MD14.x.n	U0\G09	%MW14.0.10 %MD14.0.9
Extension register		R	Rn	%MW2.n %MD2.n	R32767 R32766	%MW2.32767 %MD2.32766
Extension file register		ER	ERn	None	–	–
Pointer		P	Pn	"" (Null character)	P4095	None
Interrupt pointer		I	In	None	–	–
Nesting		N	Nn	None	–	–
Index register		Z	Zn	%MW7.n %MD7.n	Z7 Z6	%MW7.7 %MD7.6
		V	Vn	%MV6.n	V7	%MW6.7
State		S	Sn	%MX2.n	S4095	%MX2.4095

Notes

11.3 Structured Project

The components of a structured project are shown below.



Notes

11.4 Program Organization Unit

The lowest level object in the program structure of an IEC compliant program is the program organization unit, or POU. A POU is a segment of the program code for the application. POUs can be written in any of the available programming languages.

Instead of one large program like previous programming methods, a structured project should be broken into smaller, more manageable POUs. This can make troubleshooting much easier, as each POU can contain one small piece of the complete program. A typical example is to segment the code for the program based on the various sections of a machine. This way, only that section of code needs to be reviewed when there is a problem with the machine.

There are 3 different types of POUs.

- Program Blocks
- Functions
- Function Blocks

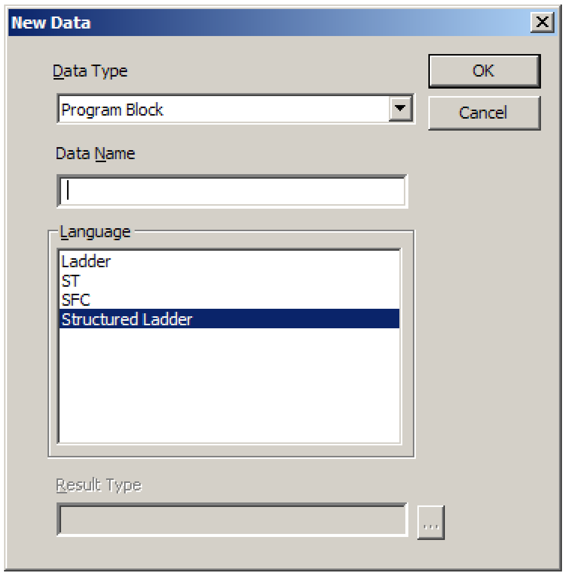
Most POUs are typically program blocks, which are executed based on the configuration of the task which the program block is included in. Functions are user-defined commands for use within other program blocks, functions, or function blocks. Functions will only have one output type.

Function blocks are sections of code which can be called from within another POU, and can be thought of as similar to subroutines. Function blocks can have multiple outputs of various types or no outputs at all. Function blocks must be assigned an instance name, since they have internal memory capable of storing data between executions.

Functions and Function Blocks will be discussed in more detail later in this class.

Notes

POUs are created inside the Program Pool in the navigation window. Right click on the Program folder under POU and select ‘Add New Data’ to create a new POU. Or select Project, then Object, then New in the menus. The new POU window below will be shown.



In this window, each POU is assigned a data type; Program Block, Function, or Function Block.

Next the POU is assigned a name.

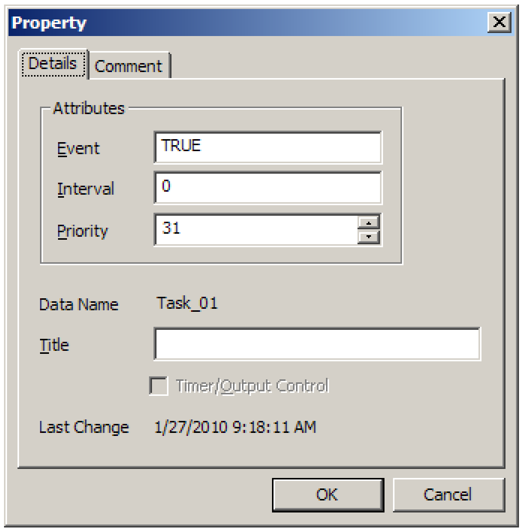
The last selection is the programming language to be used for this POU. Each POU can be programmed in only one language, but multiple POUs can be created using various programming languages within the same project.

Notes

11.5 Tasks

Tasks are used to execute the POUs created as program blocks. Multiple POUs can be included within a single task, depending on the programming language used.

The execution of a task can be based on PLC scan, a bit being on, or at a timed interval. This is configured in the task properties screen, which is shown below.



Event: TRUE
Interval: 0
Priority: 31
Data Name: Task_01
Title:
Timer/Output Control
Last Change: 1/27/2010 9:18:11 AM

To run this task as a constant scan program, the Event section should be set to **TRUE**.

To use the interval function, set Event to **FALSE**, and set a time variable in the Interval field.

Event can also be set to the on status of a bit. This bit can be a direct address (like M0) or a label.

The time period for an interval task is set in the standard IEC time format, such as **t#1m** for 1 minute, or **t#15s** for 15 seconds. This value should not be less than the PLC scan time or it will not operate properly.

Notes

The Priority setting determines the order in which the tasks are processed by the CPU. This is important in the event that more than one task is assigned the same execution criteria, such as TRUE.

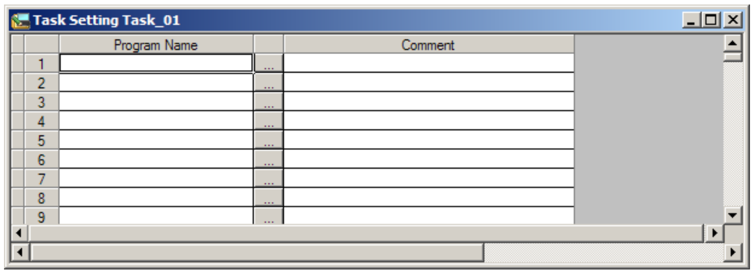
The lower priority numbers are executed first. When multiple tasks have the same Event and Priority, they are executed in alphabetical order based on the task name.

The title field is used to set a description for the task. A comment explaining the purpose of the task can be added on the second tab of the new task window.

Some important rules for tasks are shown below.

- A POU can only be registered into one task. Up to 320 POUs can be registered to a single task. Up to 124 tasks can be created in a single project.
- Only one POU of basic ladder can exist in a single task. So each POU coded in ladder (not structured ladder) must have its own task. Each task of basic ladder must also be placed in its own program.
- POUs written in SFC must exist in their own task. Multiple SFC blocks can be placed in the same task, but that task must not contain any other type of programs.

To attach POUs to a task, and to set the execution order of the POUs within a task, right click on the task in the project tree and select 'Open Task Setting'. Register the POUs into this list in the order which they should be executed.


Task Setting Task_01

	Program Name		Comment
1		...	
2		...	
3		...	
4		...	
5		...	
6		...	
7		...	
8		...	
9		...	

POUs can also be added to the task by clicking and dragging them into the task in the navigation window.

Notes

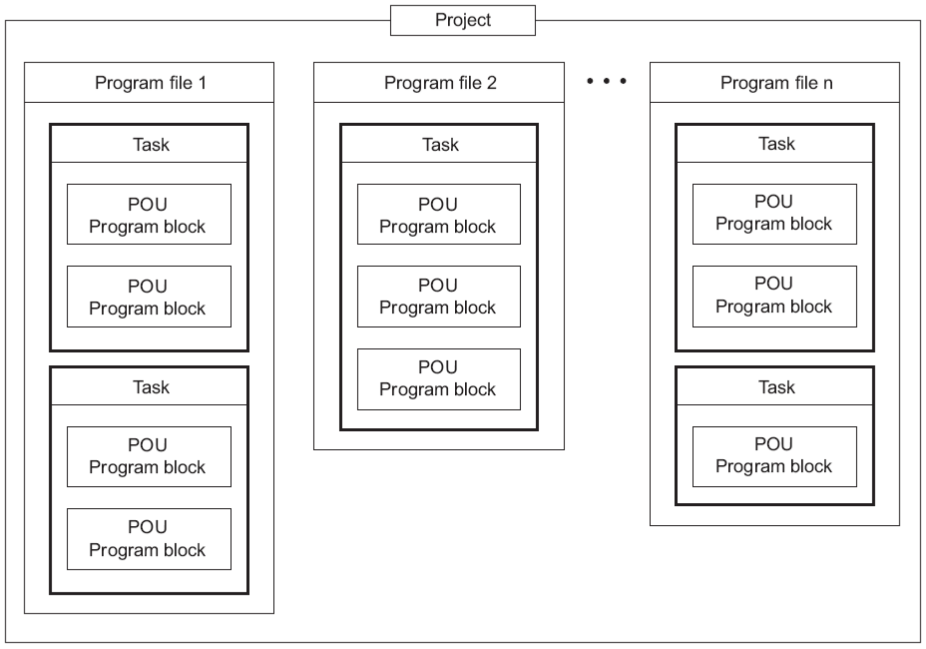
11.6 Programs

Programs are used to group tasks. Programs are assigned an execution type.

Some of the reasons to create more than one program include:

- Grouping of various tasks for a section of a system.
- Security level settings can be made differently for each program.
- Different program execution types, like initial scan or fixed scan, each will require a separate program.

The end result of a properly configured structured program will resemble the following layout.



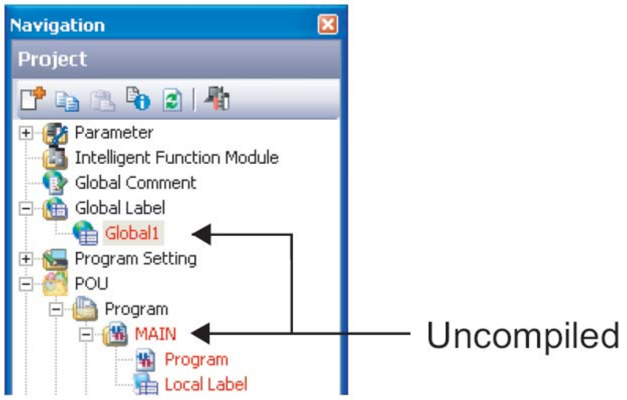
Programs are then assigned to an execution type. This can be done in the PLC Parameters in the Program tab. It can also be accomplished by dragging and dropping the programs to a specific execution type within the Program Setting section of the navigation window.

Notes

11.7 Compiling the Program

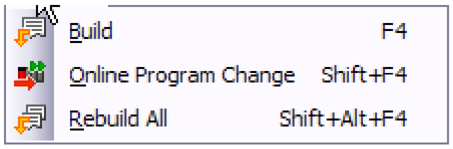
IEC 61131-3 programs must be compiled before being sent to the controller. The compile process will convert the IEC commands and programming languages back into the native instruction set of the PLC.

When programs exist which are not compiled, they will be indicated in RED in the project tree. Once compiled, the names will turn black.



It is important to note that only programs which are included in the active tasks will be compiled. Any POUs which are not included in tasks, or any programs not assigned an execution type will not be compiled.

There are 3 options in the Compile menu in GX Works2. Those items each compile a part of the program.



- Build can be used to build only uncompiled items. By only building the components which are not already compiled, the time required can be reduced.
- Online Program Change will compile a program, check for errors, and if there are no errors, update the PLC while it is running.
- Rebuild All will recompile all programs used in the current project.

The results of the compile will be displayed as errors and warnings in the Output window of GX Works2. Double clicking on one of these errors will navigate the program window to the location of the error.

By default, the compiler will abort if the number of errors reaches 25 or the number of warning reaches 100. These settings can be adjusted in the software options, and the limit can range from 1 to 9999.

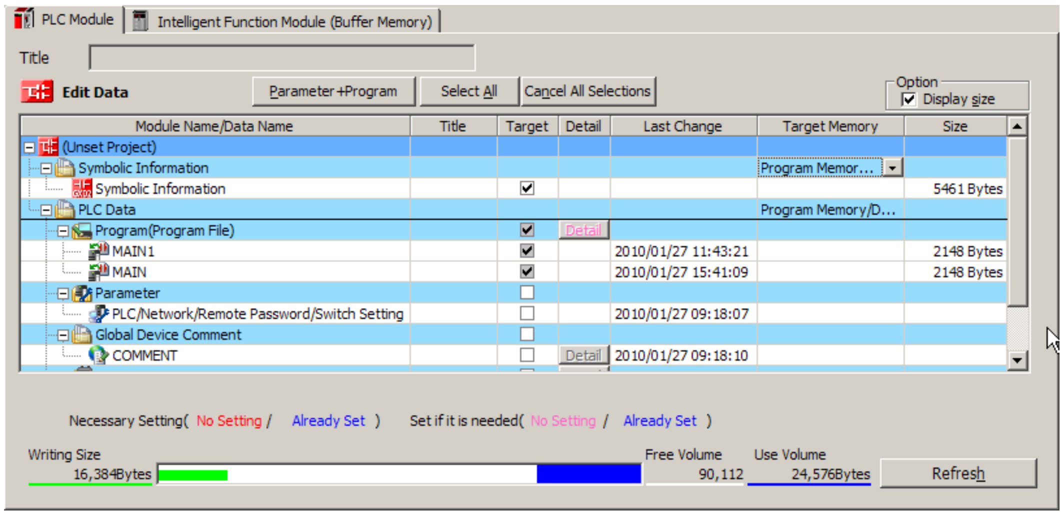
Notes

11.8 Symbolic Information

IEC programs are compiled into instruction list language which is used inside the PLC. The PLC only requires this compiled version in order to run. Since that code was compiled, the original program structure complete with labels and program types does not exist inside the CPU.

For this reason, there is an option when writing to the PLC to include the symbolic information. Symbolic information is the raw, uncompiled version of the program. If the symbolic information was downloaded to the PLC, it can be uploaded and the program languages and labels will be restored in the uploaded project. Symbolic information is checked by default as part of the 'Parameters + Program' button.

Symbolic information is typically the same size or slightly larger than the compiled PLC programs, so this will require additional memory in the PLC. In the Q Series, the symbolic information can be saved to the program memory or a SRAM or ATA Memory Card. In the L Series, symbolic information can be stored to the program memory, SD memory card, or Standard ROM. After checking the symbolic information box, there is a selection to the right for the target memory area. This does not need to be the same memory area which the program and parameters are being downloaded to.



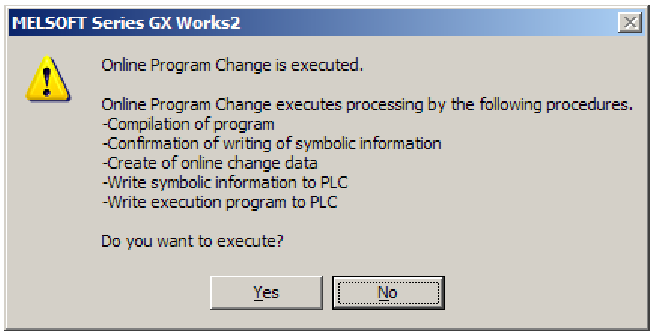
Module Name/Data Name	Title	Target	Detail	Last Change	Target Memory	Size
(Unset Project)						
Symbolic Information					Program Memor...	
Symbolic Information		✓				5461 Bytes
PLC Data					Program Memory/D...	
Program(Program File)		✓	Detail			
MAIN1		✓		2010/01/27 11:43:21		2148 Bytes
MAIN		✓		2010/01/27 15:41:09		2148 Bytes
Parameter		☐				
PLC/Network/Remote Password/Switch Setting		☐		2010/01/27 09:18:07		
Global Device Comment		☐				
COMMENT		☐	Detail	2010/01/27 09:18:10		

Necessary Setting(No Setting / Already Set) Set if it is needed(No Setting / Already Set)

Writing Size 16,384Bytes Free Volume 90,112 Use Volume 24,576Bytes

Notes

As previously discussed, when performing an online program change, a dialog will ask to confirm the write while the PLC is running.

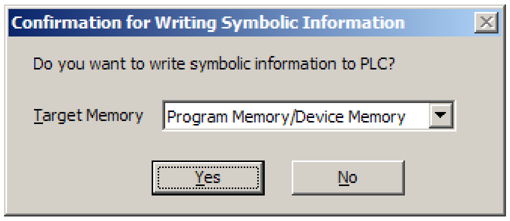

MELSOFT Series GX Works2

Online Program Change is executed.

Online Program Change executes processing by the following procedures.
-Compilation of program
-Confirmation of writing of symbolic information
-Create of online change data
-Write symbolic information to PLC
-Write execution program to PLC

Do you want to execute?

GX Works2 will check to see if symbolic information exists in the connected controller when it performs and online write. If the symbolic information exists in the controller, a prompt will be displayed asking if the symbolic information is to be updated in the controller. If the symbolic information is not updated, the executing program and the source code will not match, so it is advised that this option always be used.


Confirmation for Writing Symbolic Information

Do you want to write symbolic information to PLC?

Target Memory: Program Memory/Device Memory

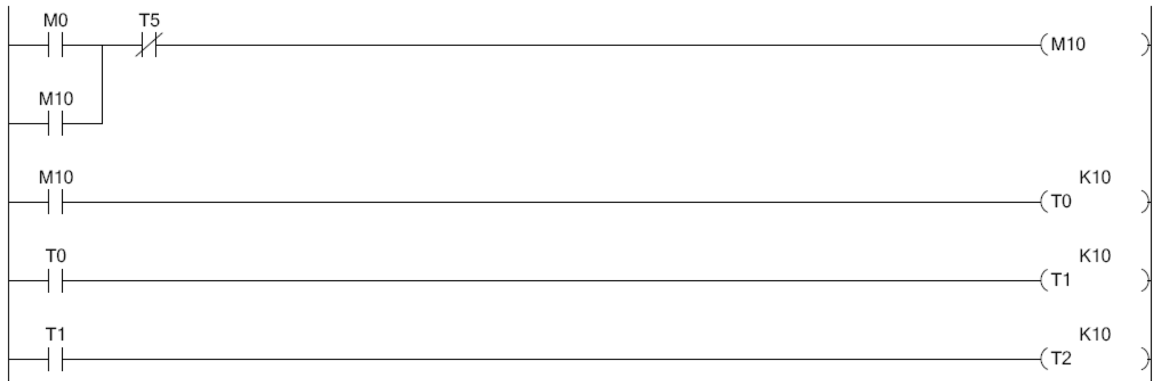
Notes

11.9 EXERCISE – Structured Project

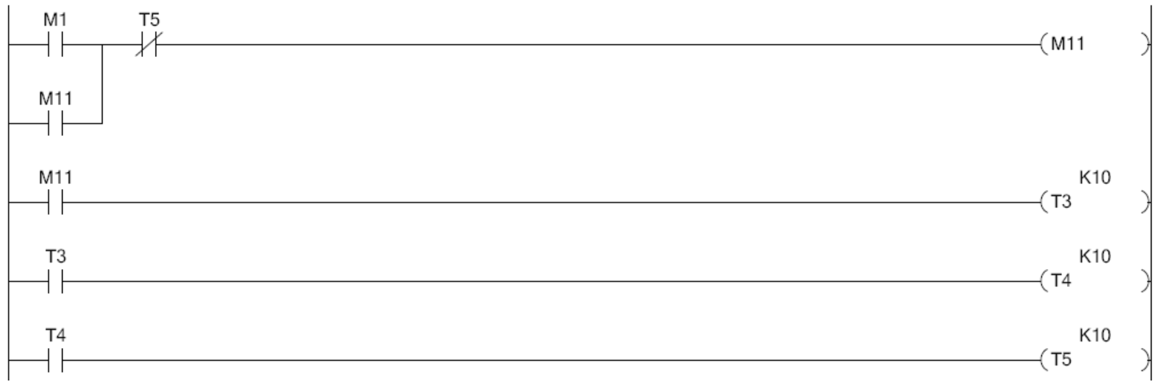
In this exercise, you will create a new structured project, add programs and tasks, and download to the CPU.

Create the first program in ladder (not structured ladder). Notice that a POU called POU_01 is already created and added to as task, called Task_01. The task is added to a program called MAIN. This all shows up under the 'No Execution Type' folder in Program Settings. To make this a scan program, drag and drop MAIN into the Scan Program folder.

Rename POU_01 to StartSequence. Enter the following code.

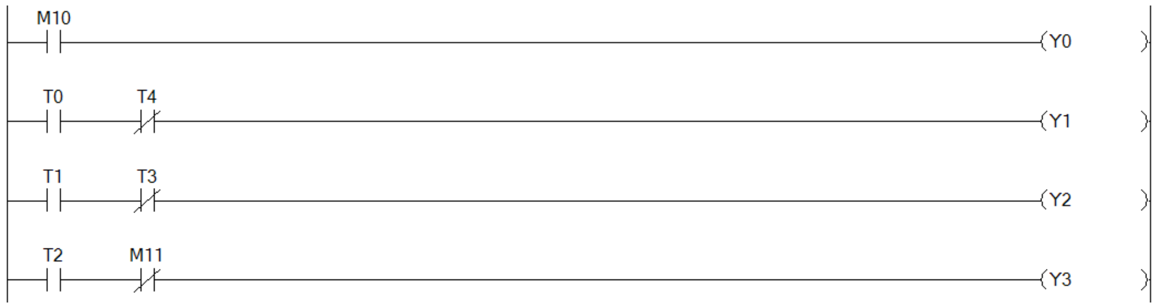


Create a new POU, called StopSequence. Enter the following code.



Notes

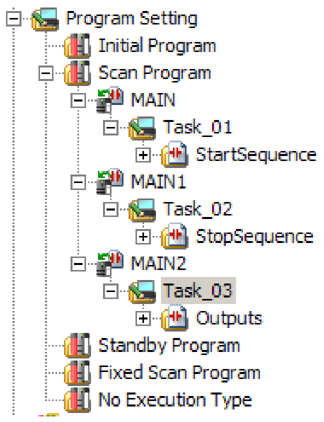
Create a third POU called Outputs. Enter the code below.



Since these POUs were all created in simple ladder, each one will need to be assigned to a separate task, and each task must be assigned to a separate program file.

Create 2 additional tasks and 2 additional programs and place one POU in each task then place each task into the new programs. Try to compile the program, and ensure that each program, POU, and task turns black and there are no errors or warnings in the compile.

End result should be similar to this:



```
Program Setting
  Initial Program
  Scan Program
    MAIN
      Task_01
        StartSequence
    MAIN1
      Task_02
        StopSequence
    MAIN2
      Task_03
        Outputs
  Standby Program
  Fixed Scan Program
  No Execution Type
```

Download this project to the CPU, including the symbolic information.

Test the operation of the conveyor sequence.

Notes

LESSON 12 – Structured Ladder

This lesson demonstrates the structured ladder programming language.

Lesson Objectives

At the conclusion of this lesson, you will be able to…

- Identify the components of the structured ladder programming language.
- Write programs in the structured ladder programming language.
- Understand how timers and counters are used in structured ladder.

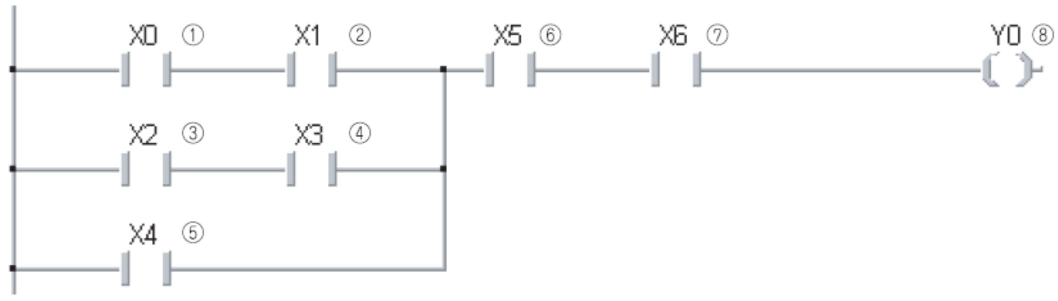
12.1 Introduction

Structured ladder is the IEC compliant ladder logic programming language. It is only available in structured projects.

The standard IEC libraries and their functions are available to the programmer in the structured ladder editor. There are also manufacturer-specific libraries, which would contain comparable commands to the Mitsubishi-standard commands in the ladder language.

Just like standard ladder, structured ladder relies on a power rail on the left hand side. There is no displayed right hand power rail in structured ladder, it is implied. Structured ladder is made up of contact and coils. Functions and function blocks can also be called from within the structured ladder editor.

Ladder processes one block at a time, left to right, top to bottom, as shown below.



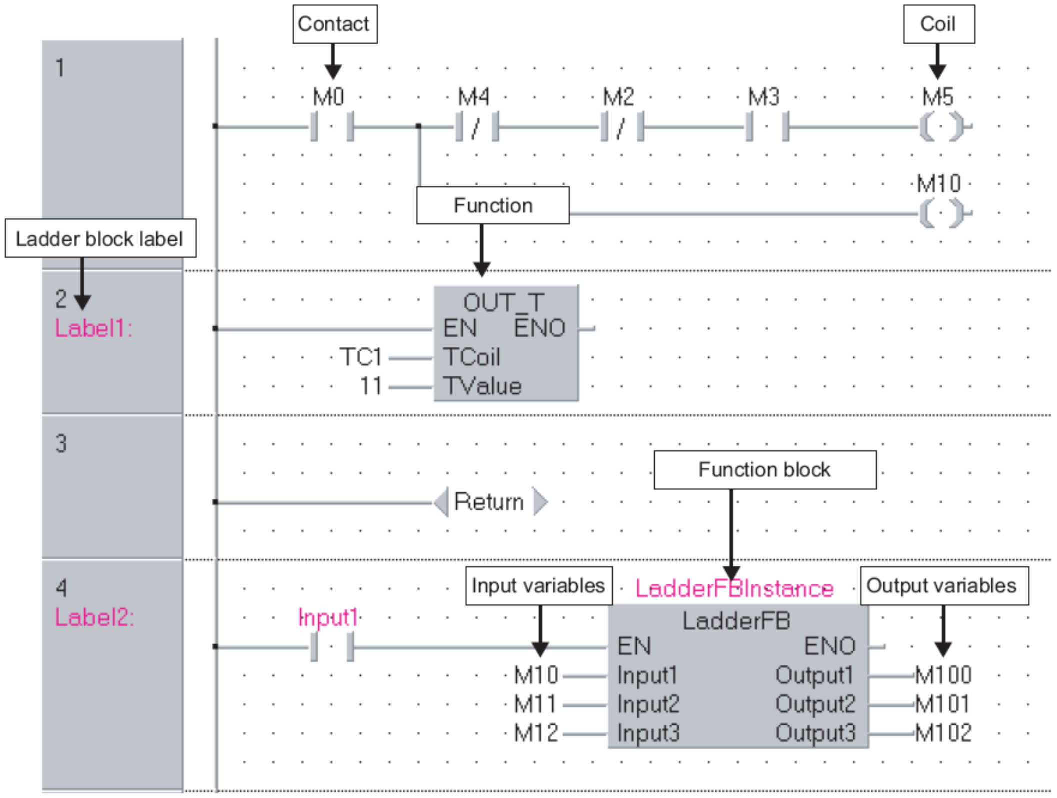
Notes

12.2 Editor Basics

The structured ladder editor resembles the standard ladder editor in many ways. It has a grid structure, and components can be placed anywhere along the grid.

Structured ladder is broken into ladder blocks. Only one rung of ladder logic can be placed in each ladder block. To insert a ladder block, right click anywhere in a ladder block, and select ‘Add Ladder Block Before’ or ‘Add Ladder Block After’.

The example below shows multiple ladder blocks, as well as contacts, coils, functions, and function blocks.



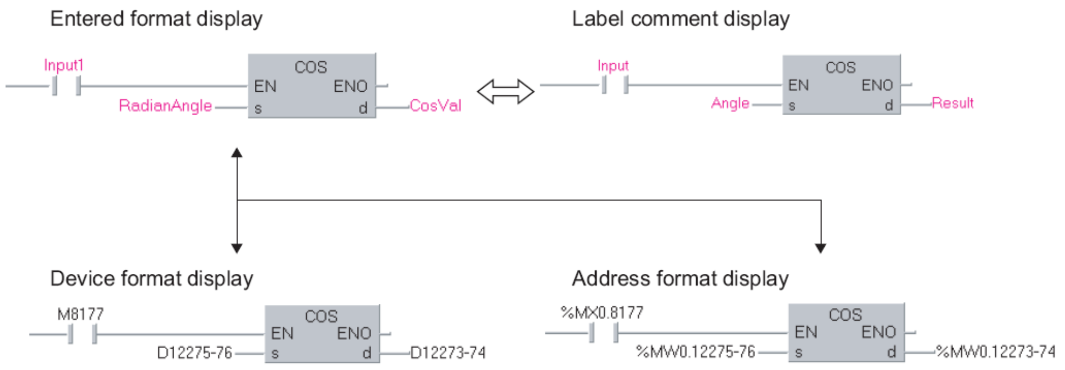
Ladder blocks can be assigned labels (used for jump and subroutines) and a title by double clicking on the gray block at the left side. Labels can be up to 8 characters, and titles can be 20 characters.

Notes

In the Edit menu, there is a Ladder Block List, which will allow display of the labels and titles of all ladder blocks. This is useful for quick navigation of the POU contents, and offers insert, delete, cut, copy, and paste functions.

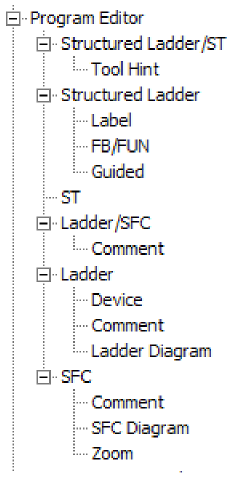
There are several options for the display format of the variables. By default, the label name will be displayed. If a label is not defined, the device address will be displayed. There is also a display format which will show the label comments instead of the labels. Another display format shows the Mitsubishi addresses assigned, and one more shows the IEC style address names.

These views can be changed from the View menu under ‘View Mode’. They can also be cycled by using the Ctrl-Shift-M keyboard shortcut.



Numerous settings exist in the software options which will affect the way the editor works. These can be customized to personal preference, and settings are stored in the PC with the software, not in the projects.

These settings can be found in the Tools menu under ‘Options. The tree to the right shows all of the pages related to program editor operation. There are 3 pages related to the structured ladder editor. Some of the settings deal with text wrapping, default sizes, and label connectors for function blocks.



Notes

12.3 Editing Modes

There are 3 basic editor modes in the structured ladder editor. These modes are:

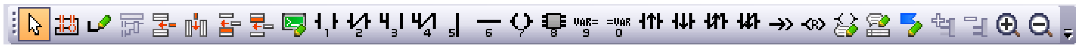
- Select mode, where the objects on the screen can be selected and moved
- Interconnect mode, which is used to draw the connecting lines between the contacts and coils
- Guided mode, which is similar to the old GX Developer ladder editor

In select or interconnect mode, there is a setting for auto-connect. This auto-connect setting is used to make connecting lines to the connection points on the symbols easier.

Each mode is indicated by the style of cursor displayed.

Mode type	Menu and toolbar	Auto connect	Mouse cursor
Select mode	[Edit] ⇒ [Select Mode] ↖	OFF	↖
		ON	↖
Interconnect mode	[Edit] ⇒ [Interconnect Mode] ✎	OFF	✎
		ON	✎
Guided mode	[Edit] ⇒ [Guided Mode] ⇒ [Guided Editing]	OFF	↖

The structured ladder toolbar is shown below. The first 3 symbols are to select the various modes. This toolbar also contains the ladder symbols, editing tools, and zoom in and zoom out buttons.



Notes

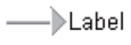
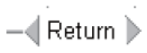
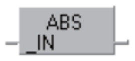
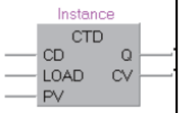
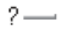
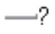
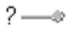
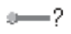
12.4 Ladder Symbols

The following table shows the ladder symbols which can be used in structured ladder mode.

Element	Ladder symbol	Description
Normal *1	-| |-	Turns ON when a specified device or label is ON.
Negation *1	-|/|-	Turns OFF when a specified device or label is OFF.
Rising edge *1,*2	-|↑|-	Turns ON at the rising edge (OFF to ON) of a specified device or label.
Falling edge *1,*2	-|↓|-	Turns ON at the falling edge (ON to OFF) of a specified device or label.
Negated rising edge *1,*2	-|↑̸|-	Turns ON when a specified device or label is OFF or ON, or at the falling edge (ON to OFF) of a specified device or label.
Negated falling edge *1,*2	-|↓̸|-	Turns ON when a specified device or label is OFF or ON, or at the rising edge (OFF to ON) of a specified device or label.
Normal	——( )-	Outputs the operation result to a specified device or label.
Negation	——(/)-	A specified device or label turns ON when the operation result turns OFF.
Set	——(S)-	A specified device or label turns ON when the operation result turns ON. Once the device or label turns ON, it remains ON even when the operation result turns OFF.
Reset	——(R)-	A specified device or label turns OFF when the operation result turns ON. If the operation result is OFF, the status of the device or label does not change.

*1 AND/OR operation based on connections.
*2 GX Works2 version 1.15R or later.

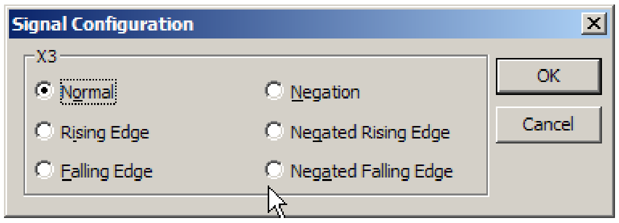
Notes

Element	Network element	Description
Jump		Pointer branch instruction Unconditionally executes the program at the specified pointer number in the same program file.
Return		Indicates the end of a subroutine program. Returns the step to the next step after the instruction which called the subroutine program.
Function		Executes a function.
Function block		Executes a function block.
Function argument input		Inputs an argument to a function or function block.
Function return value output		Outputs the return value from a function or function block.
Function inverted argument input		Inverts and inputs an argument to a function or function block.
Function inverted return value output		Inverts the return value from a function or function block and outputs it.

Ladder symbols can be created by keyboard shortcut keys or by clicking a button on the toolbar and then the desired location in the ladder block. Pressing the keyboard shortcut key will turn on the tool, and then the symbol will be drawn where the mouse is clicked.

Notes

Contacts and coils can have their function changed after drawing by double clicking on the symbol. A dialog will appear offering the choices for the symbol.


Signal Configuration

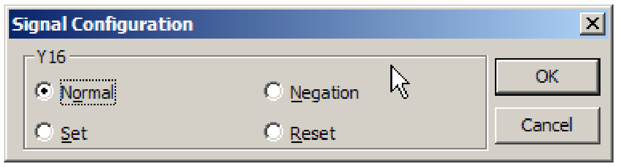
X3

[✓ Normal] Negation

Rising Edge Negated Rising Edge

Falling Edge Negated Falling Edge

OK Cancel


Signal Configuration

Y16

[✓ Normal] Negation

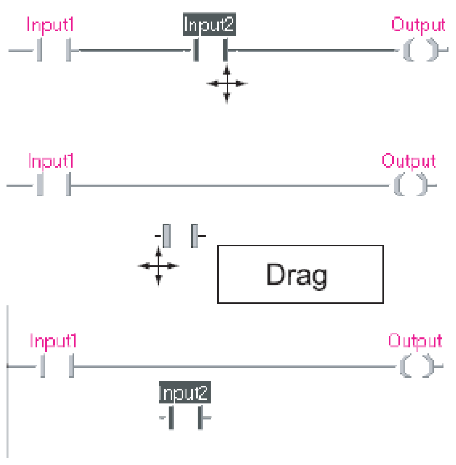
Set Reset

OK Cancel

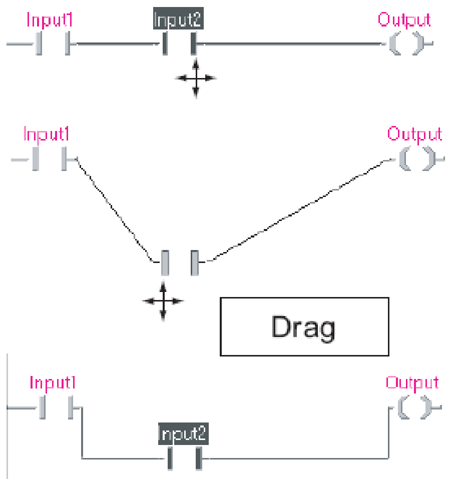
12.5 Connecting Lines

When in the select mode with auto connect off, moving a ladder symbol or function will move that object, but will not move the lines or variables which are connected to it. Before moving the object, switch to auto connect mode so that the lines will be moved with the object.

Without Auto-Connect



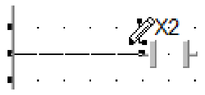
With Auto-Connect



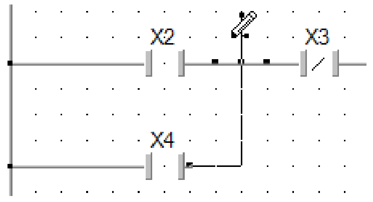
Notes

In select mode, the objects are not automatically connected with lines. The lines will need to be added later in the interconnect mode.

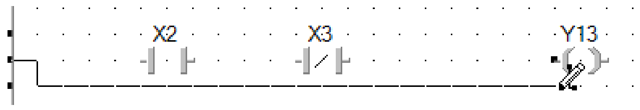
In interconnect mode, the lines are connected to the symbols. Click once on the power rail, and then move the mouse towards where the other end of the line is to be connected. When the black dot appears on the connection point, click again and the line will be drawn.



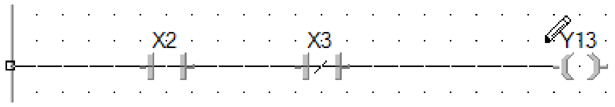
Branches can be added in this mode, as the black connector dots will appear anywhere along the line of the rung above, as shown below.

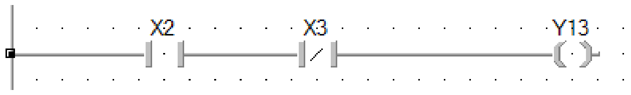


When auto-connect mode is on, dragging will draw a line only between the click and release points, and will draw around other symbols in the ladder block.



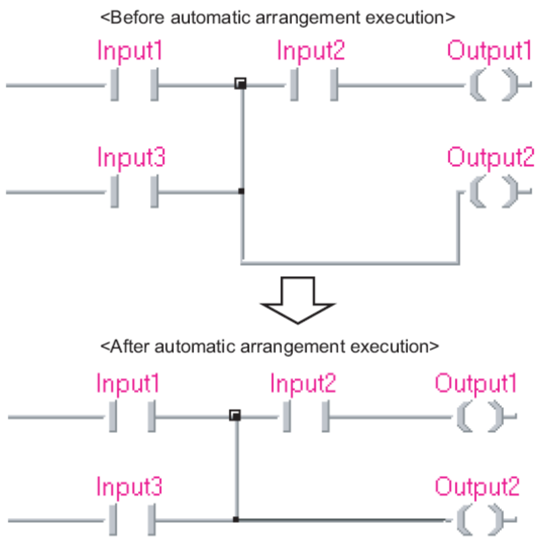
When auto-connect is off in interconnect mode, a line can be dragged directly across a rung including multiple contacts, and each contact will be inserted into the line.



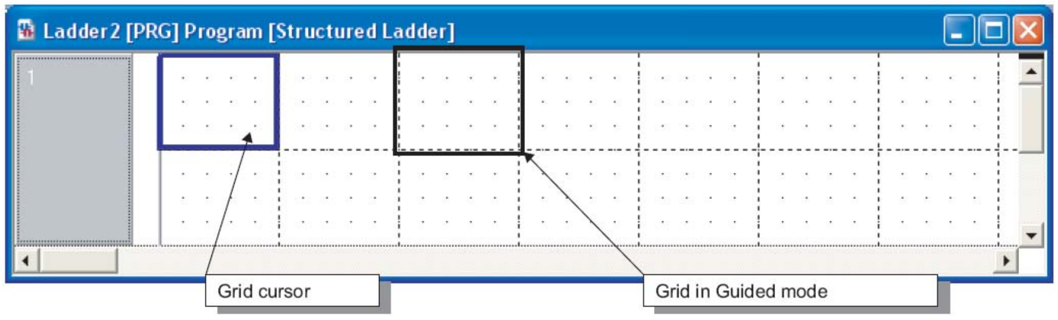


Notes

Sometimes is auto-connect mode lines are drawn which are not direct or straight. There is a tool called ‘Recalculate Line’ which will align the lines. It can be found in the Edit menu.

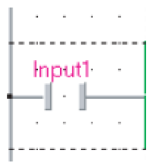

<Before automatic arrangement execution>
Input1
Input2
Output1
Input3
Output2
<After automatic arrangement execution>
Input1
Input2
Output1
Input3
Output2

In guided mode, the screen is broken into a grid format. Symbols such as contacts and coils will already have connecting lines as part of their grid. This is more similar to the standard ladder editor and the GX Developer ladder editor. This mode is useful when coding primarily from the keyboard.



Notes

In guided mode, drawing a contact will result in the symbol as shown below. Notice the horizontal line already connected to the edge of the block.

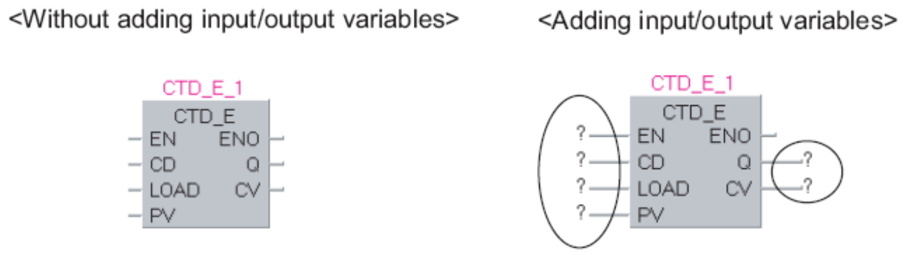


Another feature of guided mode is automatic comments on ladder blocks. This can be enabled or disabled by a toolbar button or from the Guided Mode section of the Edit menu. When this is enabled, a comment box is automatically created across the top of each new ladder block inserted into the program.

12.6 Functions and Function Blocks

The simplest way to use functions or function blocks in the structured ladder editor is to select the command from the selection window and drag it into the program.

Function blocks will require inputs and output attached for connection of variables. There is a setting in the software options which will insert these automatically when the function is dropped into the structured ladder diagram. This option is called ‘Automatic Input/Output Labels’. This setting only works when in the auto-connect mode.


<Without adding input/output variables>
CTD_E_1
CTD_E
EN
ENO
CD
Q
LOAD
CV
PV
<Adding input/output variables>
CTD_E_1
CTD_E
EN
ENO
CD
Q
LOAD
CV
PV
?
?
?
?
?
?

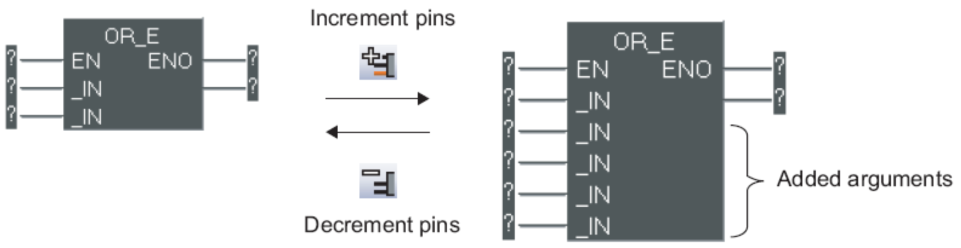
If the labels were not inserted automatically, there is a tool in the toolbar used to add these labels manually. Addresses or labels used for numeric inputs and outputs must be entered into these variables. Another setting in software options allows the connector lines to remove the variable automatically, for example when connecting a contact instead of a label.

Notes

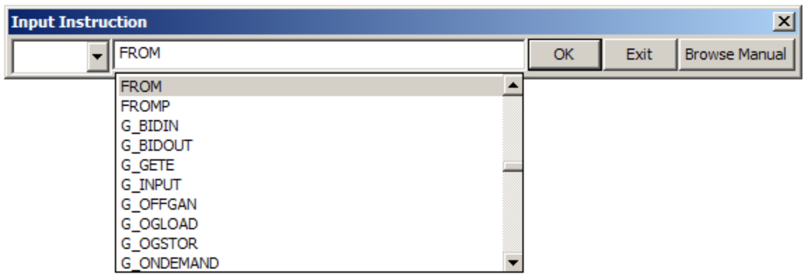
As previously mentioned, function blocks will require an instance name be defined. This instance name is displayed directly above the function block.

Some functions and function blocks allow the number of inputs to be changed. An example of this function is the OR_E instruction. This is an OR condition. The _E on the end of the name indicates the function has an enable input, which must be turned on for the command to execute. Another example of resizable functions is the add (ADD_E) and subtract (SUB_E) instructions.

To add input pins onto the block, once it is selected, use the toolbar buttons to add or remove pins. Pins can also be added and removed from the Edit menu under Number of Pins. Another method is to click and drag the bottom edge of the function until the desired number of pins is shown.



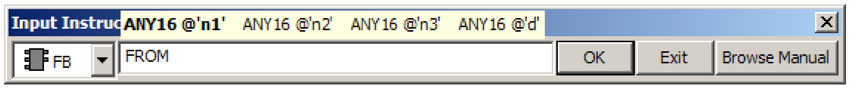
Another tool for inserting function blocks can be found in the toolbar. It is called 'Input Instruction'. This pop-up dialog lets you type a command name (or part of command name with lookup) and then enter the required parameters as was previously done in GX Developer.



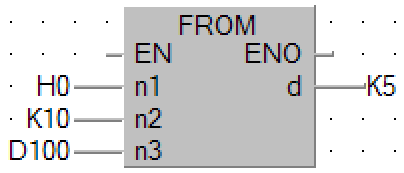
There is a 'Browse Manual' button at the right, which will display the instruction help for the selected instruction from the programming manual.

Notes

Once the command name is selected and a space is pressed, help will appear indicating what format the data has to be in for each parameter.



Once the OK button or Enter is pressed, the mouse pointer will show a symbol of a function block. Click in the ladder block to draw the instruction as configured.



As mentioned at the beginning of the lesson, structured ladder offers the entire standard Mitsubishi command set, as well as new instructions from the standard IEC libraries.

When referencing the Structured Programming manuals, several new data types will be shown. The most commonly used ones are indicated below.

- ANY_BIT means any type of bit address
- ANY_NUM means any numeric values (word, double, floating point)
- ANY_SIMPLE means any simple data type (no structures)
- ANY includes arrays and structures

An example of a command which uses the new data types is ADD. Add uses data type ANY_NUM. So this one command can be used to add 16-bit words, 32-bit words, as well as single or double precision floating point numbers. All inputs to one function must be of the same data type.

So the same ADD instruction can be used in place of +, E+, D+, or ED+ instructions.

A table in Appendix 1 of the MELSEC-Q/L/F Structured Programming Manual (Fundamentals) shows the memory areas usable for each data type.

Notes

12.7 Constants

The method for specifying constants to a structured program is different than in basic ladder. The table below shows the different methods which may be used to enter a constant value into a function or function block.

Constant type	Expressing method	Example
Bool	Input FALSE or TRUE, or input 0 or 1.	TRUE, FALSE
Binary	Append '2#' in front of a binary number.	2#0010, 2#01101010
Octal	Append '8#' in front of an octal number.	8#0, 8#337
Decimal	Directly input a decimal number, or append 'K' in front of a decimal number.	123, K123
Hexadecimal	Append '16#' or 'H' in front of a hexadecimal number. When a lowercase letter 'h' is appended, it is converted to uppercase automatically.	16#FF, HFF
Real number	Directly input a real number or append 'E' in front of a real number.	2.34, E2.34
Character string	Enclose a character string with single quotations (') or double quotations (").	'ABC', "ABC"
Time	Append 'T#' in front.	T#1h, T#1d2h3m4s5ms

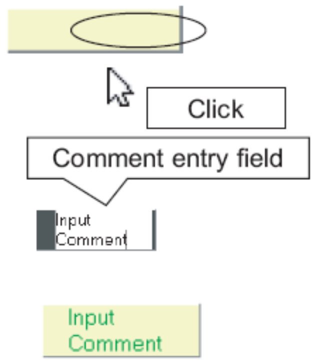
Floating point data constants must have a decimal place in them. Some instruction types will require a specific type of data. Be sure to use the expression methods above to ensure the program compiles properly.

Notes

12.8 Program Documentation

Comments can be placed anywhere within a ladder block. To insert a comment, select the comment tool on the toolbar and then click in the ladder block. A yellow box will be placed at the location of the mouse click.

Text can be entered by clicking once in the comment window body and typing. It can be resized by clicking and dragging the corners. It can be moved by clicking and dragging on the header on the left side of the comment box.



Multiple comment boxes can be placed in a single ladder block. Comment blocks are not allowed to overlap each other or any symbols in the ladder block.

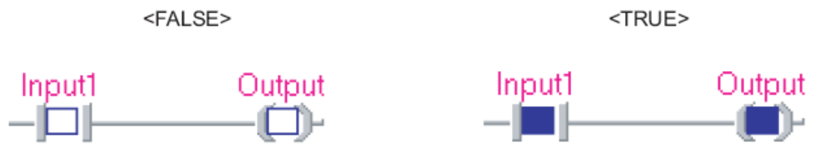
There is an option setting in guided mode to automatically add a comment across the top of each ladder block. When this option is selected, a comment box which is one line of text high is created across the top of a number of cells in a new ladder block. The number of cells can be defined in software options. This would most closely resemble the statements used to comment rungs in simple ladder.

Notes

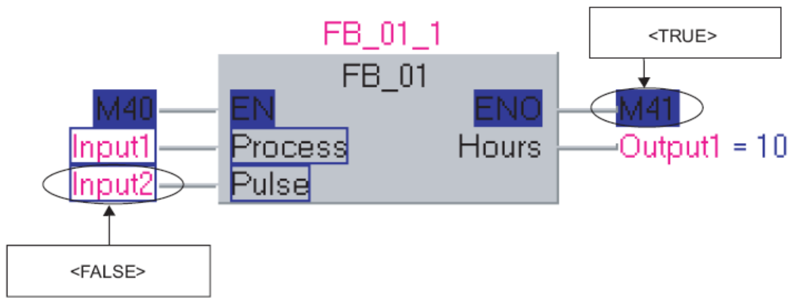
12.9 Monitoring

Monitoring can be stopped and started individually in any open window. Once monitoring is started in a structured ladder window, the status of the devices is shown as indicated below.

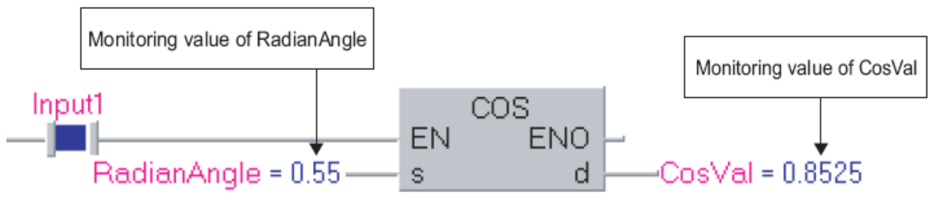
Contacts and coils will fill with a solid blue box when the contact or coil is true.



Bit devices used in functions and function blocks will highlight with blue boxes around the address or label. If the bit is true, the box will be filled with dark blue. If the bit is false, the box will be clear.



Numeric labels and addresses will display their value next to the label or address. The format of the displayed numeric data will be based automatically on the type of data used in the instruction. By default, U\G addresses do not monitor due to the increased communications on the CPU bus.



Notes

12.10 EXERCISE – Structured Ladder

Create a new structured project. Using the concepts explained in this lesson, create the following program in the structured ladder language.

Write the program for a carton filling machine. The machine operator will enter a number via the operator interface which indicates the number of products to place in a carton. Once the start button is pressed, the conveyor will turn on to move products into the carton. Products are counted by the product sensor. When the number of products in the carton equals the number entered via the operator interface, the conveyor will turn off. Turn on the indicator lamp when the carton is filled. Turn it off when the system starts again.

DO NOT ALLOW THE CARTON SIZE TO BE CHANGED WHILE THE SYSTEM IS RUNNING!

DO NOT USE A COUNTER FOR THIS EXERCISE.

Use the following addresses (to ensure the GOT program communicates properly). Any addresses not listed below can be assigned anywhere in the PLC memory, including labels without address definitions.

- Bits
  - Start M0
  - Part Sensor M1
  - Conveyor Y0
  - Carton Full Lamp Y1
- Words
  - Carton Fill Size D0
  - Current Number in Carton D1

Notes

12.11 Timers and Counters

Timers and counters are not coded the same way in the structured ladder program which they are coded in the standard ladder logic. Timers and counters are coded as functions. The commands most comparable to the standard timer and counter coils in ladder logic are:

- OUT_T for a timer
- OUTH_T for a high speed timer
- OUT_C for a counter

Timers and counters also have a different method of distinguishing the timer contact than in standard ladder. Each timer or counter has 3 addresses, as indicated below.

- TC is the timer coil, used on the OUT_T or OUTH_T function
- TS is the completion indicator, used when coding a contact on the timer
- TN (or just T) is the current value, used in word commands

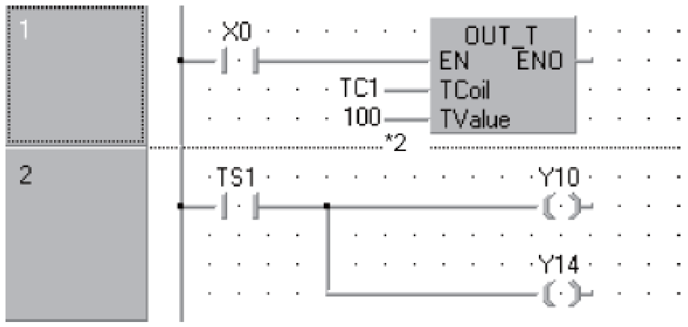
- STC, STS, and STN or ST are used for retentive timers

- CC is the counter coil, used on the OUT_C function
- CS is the completion indicator for use on contacts
- CN (or just C) is the current value, used in word commands

These commands are covered in the Q/L Structured Programming Manual (Common Instructions).

Notes

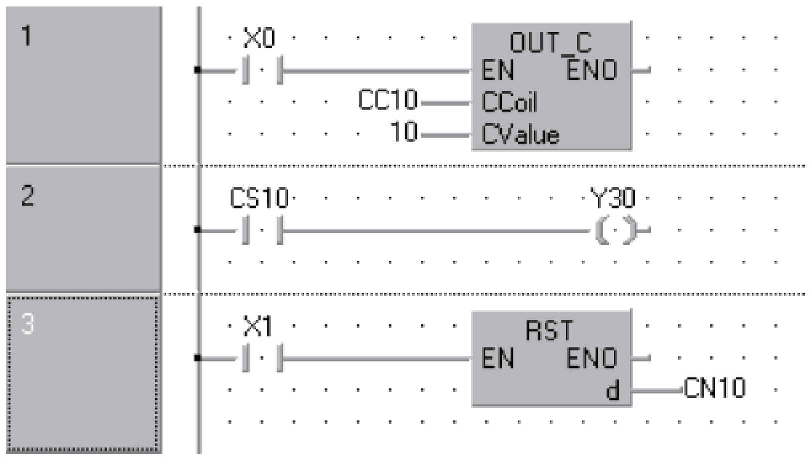
An example of the OUT_T command is shown below.



The EN (enable) input starts the timer timing. The timer number is specified at the TCoil input. The preset is specified at the TValue input. This is a word value, and can be an integer or a variable name or a direct address.

Retentive timers use the same commands, but specify a retentive address in the function's input connections.

The OUT_C instruction is shown below.



Notes

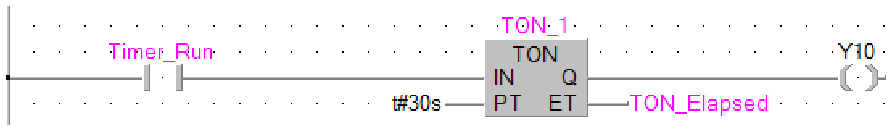
The IEC command set also includes standard timer and counter function blocks, which are listed below.

- TON for an on-delay timer
- TOF for an off-delay timer
- TP for a pulse timer
- CTU for an up counter
- CTD for a down counter
- CTUD for an up/down counter

Adding an _E to the function block names above creates a function block with enable input and enable output. These timer instructions take a preset value in the time format standard, and output their current value in the time format.

These commands are covered in the Q/L Structured Programming Manual (Application Instructions).

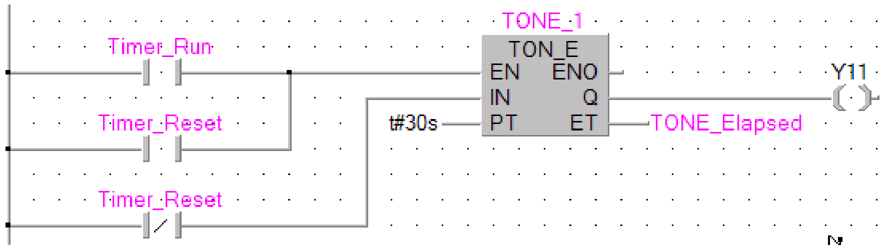
An example of the TON instruction is shown below. While Timer_Run is on, the timer will time to a preset of 30 seconds, and then turn on output Y10. The current elapsed time is stored in TON_Elapsed. The timer will reset automatically when Timer_Run is turned off.



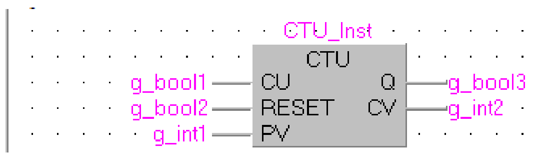
A retentive timer function can be accomplished with the TON_E command. The EN input will allow the timer to time forward as long as IN is also true. When EN turns off and IN remains on, the value in the timer is retained. When EN turns back on, the timer will continue from its previous value. To reset the timer, EN must be on while IN turns off.

Notes

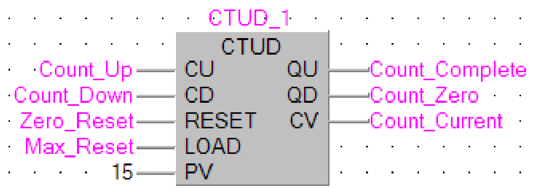
An example of the TON_E instruction is shown below. While Timer_Reset is off and Timer_Run is on, the timer runs to a preset of 30 seconds, and then Y11 turns on. TONE_Elapsed will display the current elapsed time. In order to reset the timer, EN must be on while IN is off, so Timer_Reset is used on both EN and IN connections.



The example below is the CTU instruction. When g_bool1 turns on, count value in g_int2 increments by one, and when g_bool2 turns on, count value is reset to 0. Counter will count until the preset set in g_int1, and then output g_bool3 turns on.



The CTUD instruction can count up and down. Input CU causes a count up. Input CD causes a count down. Inputs RESET and LOAD are the reset signals. RESET will reset the counter to zero. LOAD will load the preset value into the counter. The PV input is a number or register which stores the counter preset. Output QU indicates when the counter has counted up to the preset value. Output QD indicates when the counter has counted down to zero. Output CV stores the current value of the counter.



Notes

12.12 EXERCISE – Timers and Counters

This exercise will demonstrate the use of the timer and counter instructions in the structured ladder language. Create a new structured project for this exercise.

PART 1

Write a program to control lead/lag status of 2 pumps. Each time a request for water reaches the PLC, the lead pump should turn on. If that request is not turned back off within 10 seconds, the second pump should turn on. When the request turns off, all pumps should stop.

The pumps should alternate after each request to balance the run time on the two pumps.

PART 2

Track total usage (in seconds) of each pump in data registers. Provide a switch for each total to allow it to be reset.

Use the following addresses (to ensure the GOT program communicates properly). Any addresses not listed below can be assigned anywhere in the PLC memory, including labels without address definitions.

- Bits

o Water Request	M0
o P1 Hours Reset	M15
o P2 Hours Reset	M16
o Pump 1	Y0
o Pump 2	Y1

The word addresses used for the hours counting can be assigned anywhere, or left unallocated. Monitor these values in the software.

Notes

Notes

LESSON 13 – Structured Text

This lesson will introduce the structured text programming language.

Lesson Objectives

At the conclusion of this lesson, you will be able to…

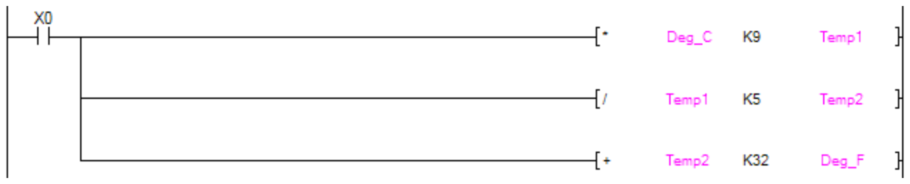
- Explain the syntax of the structured text programming language.
- Write programs in structured text.

13.1 Introduction

The structured text language is a text-based programming language similar in syntax to the C programming language on a PC. Grammatical layout and command names are very similar to C programming. By using the structured text language, a person familiar with PC programming can easily write code for a PLC.

Structured text can be used for complex programming applications, such as conditional judgment, mathematical formulas, and repeating processes, which may not be easily written in the structured ladder language.

By using simple assignment syntax, mathematical formulas can be calculated without creating a series of ladder functions as previously used. As an example, Celsius to Fahrenheit conversion would typically be accomplished by multiplying the incoming Celsius number by 9, dividing that result by 5, and then adding 32 to that result. This is 3 separate math commands in ladder logic, as shown below.



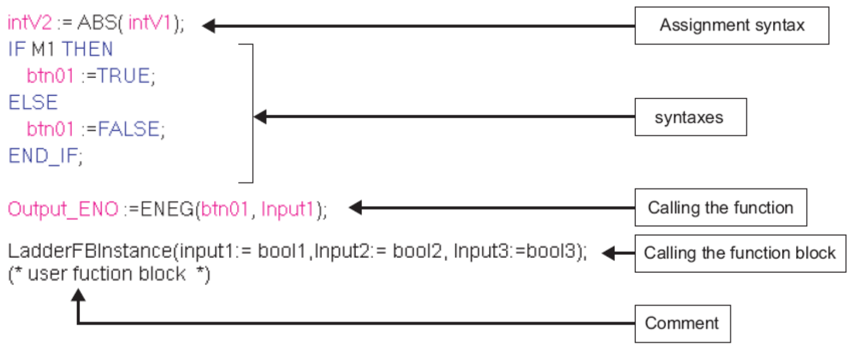
In structured text, it can be written as shown below.

```
Deg_F := (9 * Deg_C) / 5 +32;
```

Notes

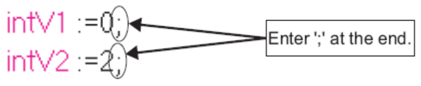
13.2 Editor Basics

Lines of code in structured text are called syntaxes. Each syntax must end with a semicolon. Functions and function blocks can be called from within a structured text program as shown below.

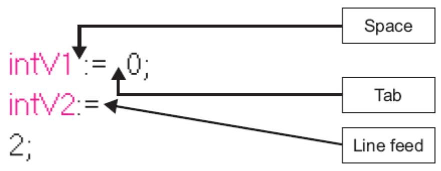


```
intV2 := ABS( intV1);
IF M1 THEN
   btn01 :=TRUE;
ELSE
   btn01 :=FALSE;
END_IF;

Output_ENO :=ENEG(btn01, Input1);

LadderFBInstance(input1:= bool1,Input2:= bool2, Input3:=bool3);
(* user fuction block  *)
```

Spaces, tabs, and line breaks can be inserted anywhere between keywords and identifiers. The examples shown below are called assignment syntaxes. Each is assigning a numeric value to a variable. Both examples are valid code, and perform the same function.

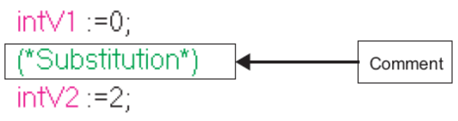


```
intV1 :=0;
intV2 :=2;
```



```
intV1 :=  0;
intV2:=
2;
```

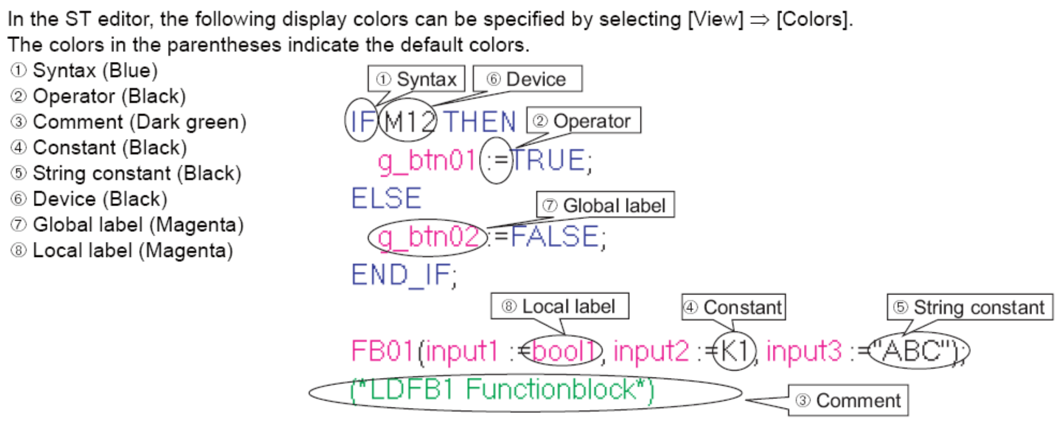
Comments can be entered at any location in the program by enclosing the comment text as shown below.



```
intV1 :=0;
(*Substitution*)
intV2 :=2;
```

Notes

GX Works2 will automatically apply color codes to the text in a structured text program to make the program easier to read. The color codes are adjustable in the software options. The default settings are shown below.



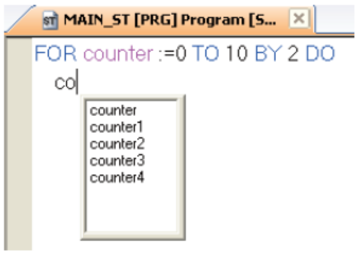
In the ST editor, the following display colors can be specified by selecting [View] ⇒ [Colors].
The colors in the parentheses indicate the default colors.

① Syntax (Blue)
② Operator (Black)
③ Comment (Dark green)
④ Constant (Black)
⑤ String constant (Black)
⑥ Device (Black)
⑦ Global label (Magenta)
⑧ Local label (Magenta)

```
IF M12 THEN
  g_btn01 :=TRUE;
ELSE
  g_btn02 :=FALSE;
END_IF;

FB01(input1 :=bool1, input2 :=K1, input3 :="ABC");
(*LDFB1 Functionblock*)
```

GX Works2 will also automatically tab indent consecutive lines of the same command, as demonstrated above. Lines are indented until the semicolon indicates the end of that syntax.

GX Works2 will perform automatic completion of the label names used while programming. When entering a label, all labels beginning with the typed characters are displayed and can be selected from the selection list. If only one label matches, it will be automatically filled in. The auto-complete function uses both the global label list and the local labels of the edited program. The project should be saved with the labels entered for this function to work.



```
FOR counter :=0 TO 10 BY 2 DO
  co
```

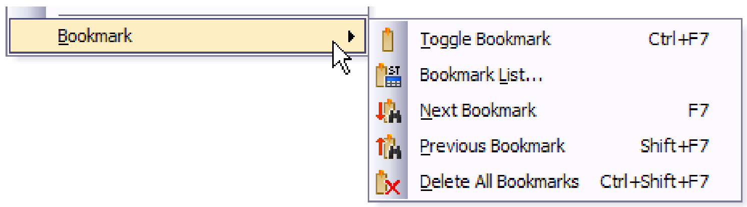
```
counter
counter1
counter2
counter3
counter4
```

Labels can be added while editing program by pressing F2 or using List Operands from the Edit menu.

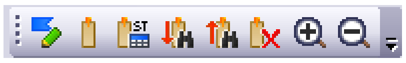
Notes

The structured text editor has a bookmark tool for quick navigation of a program. The programmer can set bookmarks on various lines in the program, and use keyboard shortcuts or a list window to navigate the bookmarks.

The bookmark tools are found at the bottom of the Find/Replace menu.

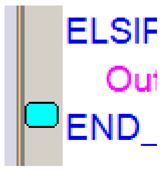


They are also available from the structured text toolbar.

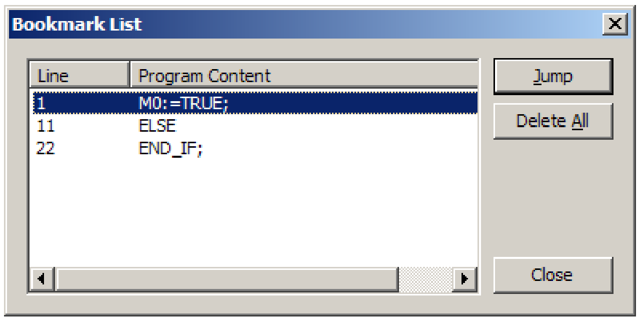


Bookmarks are registered with Ctrl-F7, or the tool in the toolbar, and apply to the line in the structured text. Clicking the tool or pressing Ctrl-F7 again on the same rung will remove the bookmark.

Bookmarks are shown in the software as a light blue box in the gray margin on the left hand side of the structured text program.



There is a list window for browsing all of the bookmarks set in the program. From this window the programmer can jump to any bookmark, or can clear all bookmarks.



Line	Program Content
1	M0:=TRUE;
11	ELSE
22	END_IF;

Notes

13.3 Operators

The table below shows the available operators in the ST language and their execution priority. This priority sets the order of execution of the operators within a single syntax.

Operator	Description	Example	Priority
( )	Parenthesized expression	(1+2)*(3+4)	Highest
Function ( )	Function (Parameter list)	ADD_E(bo01, in01, in02, in03)	↑
**	Exponentiation	re01:= 2.0 ** 4.4	
NOT	Inverted bit value	NOT bo01	
* / MOD	Multiplication Division Modulus operation	3 * 4 12 / 3 13 MOD 3	
+ -	Addition Subtraction	in01 + in02 in01 - in02	
<, >, <=, =>	Comparison	in01 < in02	
= <>	Equality Inequality	in01 = in02 in01 <> in02	
AND, &	Logical AND	bo01 & bo02	
XOR	Exclusive OR	bo01 XOR bo02	
OR	Logical OR	bo01 OR bo02	Lowest

If multiple operators in a syntax have the same priority, they are processed left to right.

13.4 Syntaxes

There are 4 basic syntax types in structured text.

- Assignment syntax
- Conditional syntax
- Iteration syntax
- Control syntax

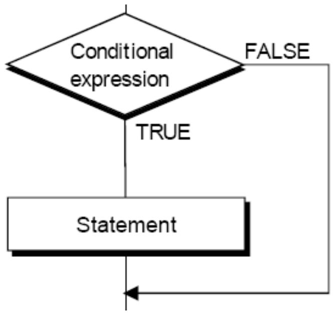
Notes

Assignment syntax is used to assign a value to a label or address. The labels or addresses must represent the same data format, such as an integer or real number. Assignment syntax uses a colon followed by an equal sign, as shown below.

```
M0:=TRUE;
D0:=D1*(D2+3)/100;
Output1:=Input1 + Input2;
```

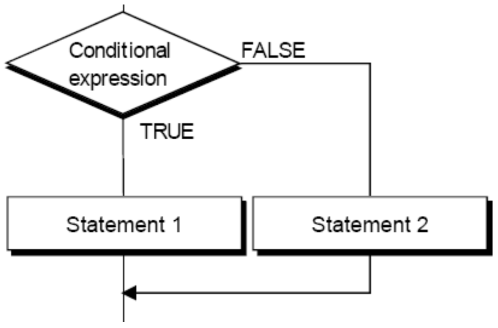
Conditional syntaxes are used to perform an action when a Boolean expression is true. Samples of conditional syntax include IF THEN, IF ELSE, IF ELSIF and CASE. The result of the IF is always a Boolean TRUE/FALSE condition. Some examples of conditional syntax are shown below. Conditional syntaxes can be nested to 16 levels.

The IF instruction will run the included statements only if the expression is true. If it is false, nothing is executed.



```
IF Bit1 THEN
   Output1:=Output1+1;
END_IF;
```

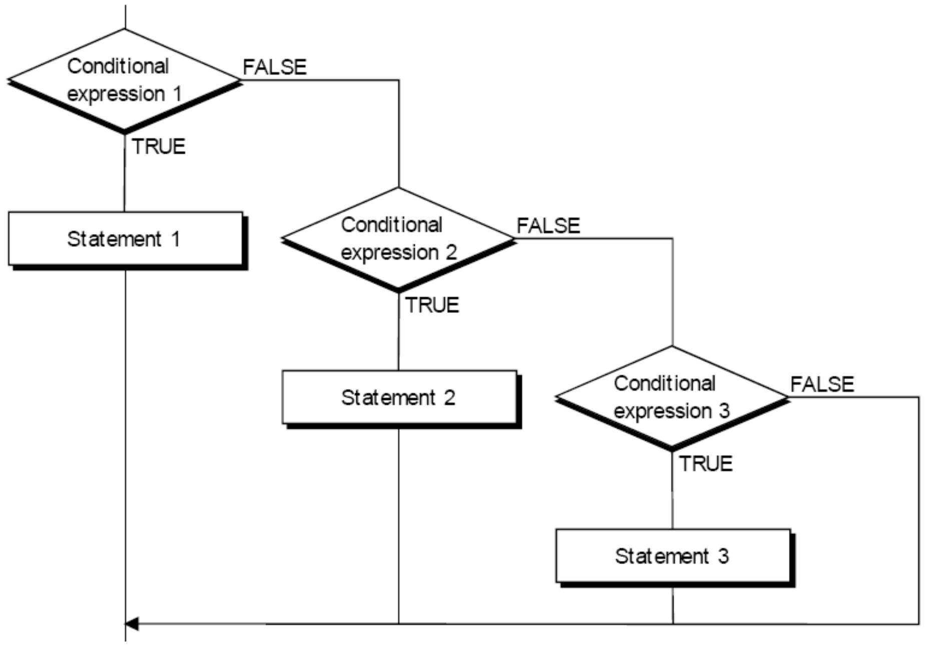
The IF ELSE instruction will run one of two sets of statements, depending on whether the condition is true or false.



```
IF Bit1 AND Bit2 THEN
   Input2:=Input1*5;
ELSE
   Input2:=Input1*4;
END_IF;
```

Notes

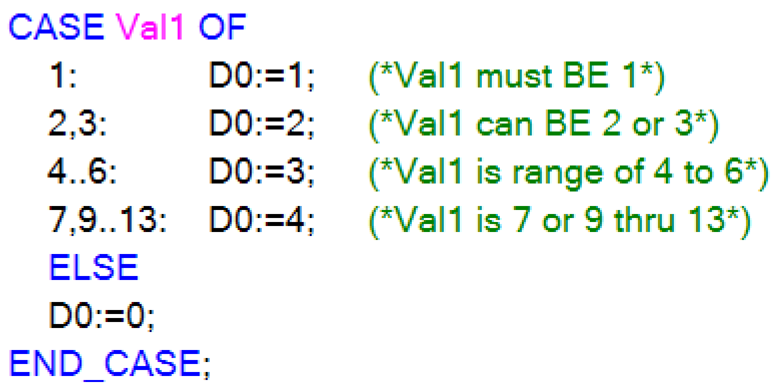
The IF ELSIF command will process one of several sets of statements depending on the status of multiple comparisons. If the first expression is false, the ELSIF instruction is performed. If that expression is false, the next ELSIF is performed. If none of the conditions is true, nothing is performed.



```
IF Bit1 OR Bit2 THEN
   Output2:=Input1+Output1;
ELSIF  Bit1 AND Bit2 THEN
   Output2:=50;
   M1:=TRUE;
ELSIF  Bit3 THEN
   Output1:=Output2;
END_IF;
```

Notes

The CASE instruction is considered a conditional syntax, and the result is based on which case in the instruction is true. The value evaluated by the case instruction is an integer value. If none of the cases are true, the ELSE condition is executed, if present. Cases can be established based on a single value, multiple values separated by commas, or a range of numbers defined by a double period in between the lower and upper values. Multiple formats can be combined within a single case. If more than one case applies, the first one which applies is executed, and the other cases are not executed.



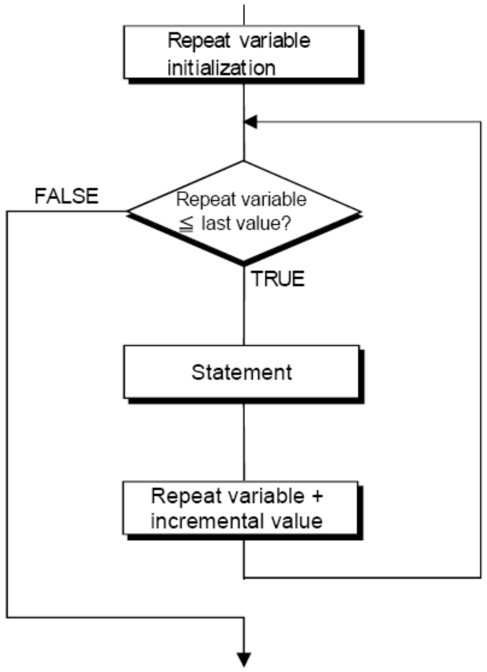
```
CASE Val1 OF
   1:          D0:=1;    (*Val1 must BE 1*)
   2,3:        D0:=2;    (*Val1 can BE 2 or 3*)
   4..6:       D0:=3;    (*Val1 is range of 4 to 6*)
   7,9..13:    D0:=4;    (*Val1 is 7 or 9 thru 13*)
   ELSE
   D0:=0;
END_CASE;
```

Iteration syntaxes are used to repeat an assignment or function. The FOR DO syntax repeats a command a specified number of times. The WHILE DO syntax repeats as long as the Boolean condition is true. The REPEAT UNTIL syntax continues to repeat until a boolean condition becomes true.

Notes

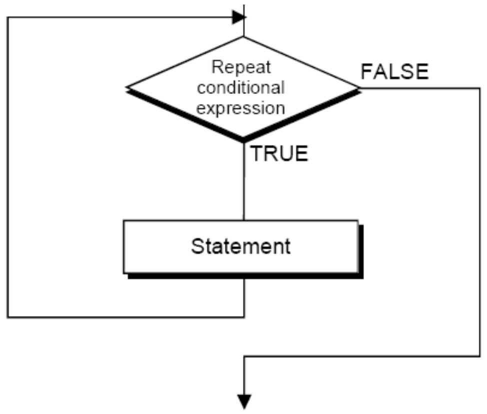
The FOR loop will repeat a specified number of times, and can increment a value at each execution. The FOR value is assigned a starting value. The BY portion sets up the amount to increment the value, and is optional. The DO portion of the command sets the actions to repeat.

```
FOR Output1:=0
   TO 10 BY 1 DO
   Output2:=Output1*2;
END_FOR;
```


Repeat variable initialization
FALSE
Repeat variable ≤ last value?
TRUE
Statement
Repeat variable + incremental value

The WHILE command will continue to repeat as long as the condition expression is true. Once the condition is false, execution will pass from the loop to the next instructions.

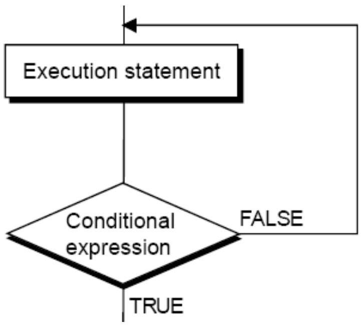
```
WHILE Val1 < 20 DO
   Output1:=Output2+Val1;
END_WHILE;
```



Notes

The REPEAT UNTIL command will continue to run as long as the condition set by UNTIL is true.

```
REPEAT
   D1:=D1+1;
   UNTIL D1 > 100
END_REPEAT;
```



The control syntaxes are used to modify the execution of a program. The RETURN instruction will jump to the end of the structured text program, ignoring the remaining program steps. The EXIT instruction is used to break out of an iteration syntax before the end of its process.

```
IF Bit1 THEN
   RETURN;
END_IF;

FOR Val1 := 0
   TO 100 BY 1 DO
   Val2:= Val1 + D0;
   IF Val2 > 100 THEN
      EXIT;
   END_IF;
END_FOR;
```

Notes

13.5 Functions and Function Blocks

When coding a function in structured text, the variables for the function are enclosed in parentheses after the name of the function. If there is more than one variable required, they are separated by commas.

The result of the function is stored to the variable as designated by the assignment syntax. The examples below show functions with one variable and three variables.

Calling a function with one input variable (Example: ABS)

```
        Output1 := ABS(Input1);
```

Calling a function with three input variables (Example: MAX)

```
        Output1 := MAX(Input1, Input2, Input3);
```

Function blocks can require more than one input, and can have more than one output. The function block must have an instance declared, so be sure to add a local variable of the data type to match the desired command.

The function block is called by its instance name, and then both the inputs and outputs can be defined inside parentheses. The name of the signal and the value must be specified with an assignment syntax.

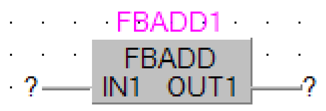
Inputs and outputs can also be defined using assignment syntax. The inputs will be defined prior to the function block call, and the outputs will be defined after the function block is called.

Notes

The example below shows a function block with one input and one output, with the output shown in a separate assignment syntax. Notice the name of the function block is not used in the command, only the name of the instance. An example of what this function block would look like in structured ladder is provided only as a reference, as it would not be coded in the program.

Calling a function block with one input variable and one output variable

FB definition
FB Name: FBADD
FB instance name: FBADD1
Input variable1: IN1
Output variable1: OUT1



Variables are assigned with the typical assignment syntax inside the parentheses by assigning the input or output name to the variable. In the example below, outputs are handled outside the function block in an assignment syntax.

```
FBADD1(IN1:=Input1);
Output1:=FBADD1.OUT1;
```

Another method of writing this code, with input and output parameters included in the one syntax, is shown below.

```
FBADD1(IN1:=Input1,OUT1:=Output1);
```

The third method to write this function block is using assignment syntax for all inputs and outputs.

```
Input1:=FBADD1.IN1;
FBADD1();
Output1:=FBADD1.OUT1;
```

Notes

Below is an example of a function block with 3 inputs and two outputs.

Calling a function block with three input variables and two output variables

FB definition

FB Name: FBADD

FB instance name: FBADD1

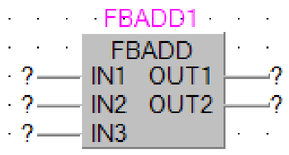
Input variable1: IN1

Input variable2: IN2

Input variable3: IN3

Output variable1: OUT1

Output variable2: OUT2



The three different ways to call this instruction are shown below.

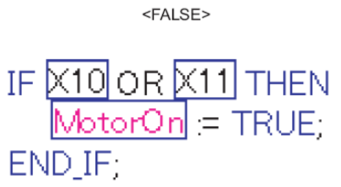
```
FBADD1(IN1:=Input1,IN2:=Input2,IN3:=Input3);
Output1:=FBADD1.OUT1;
Output2:=FBADD1.OUT2;

FBADD1(IN1:=Input1,IN2:=Input2,IN3:=Input3,
OUT1:=Output1,OUT2:=Output2);

Input1:=FBADD1.IN1;
Input2:=FBADD1.IN2;
Input3:=FBADD1.IN3;
FBADD1();
Output1:=FBADD1.OUT1;
Output2:=FBADD1.OUT2;
```

Notes

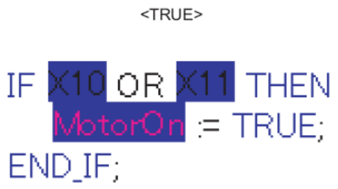
13.6 Monitoring

Monitoring can be started on a structured text program. When monitoring is started, the structured text program window is split into 2 columns. On the left side will be the structured text code, and on the right side will be a list of the numeric variables used in the program.

Bits can be monitored on the left side of the screen. Boxes around the bit labels or addresses will display the on or off status of that variable. If the box is white, the contact is not true. If the box is solid, the contact is on.


<FALSE>

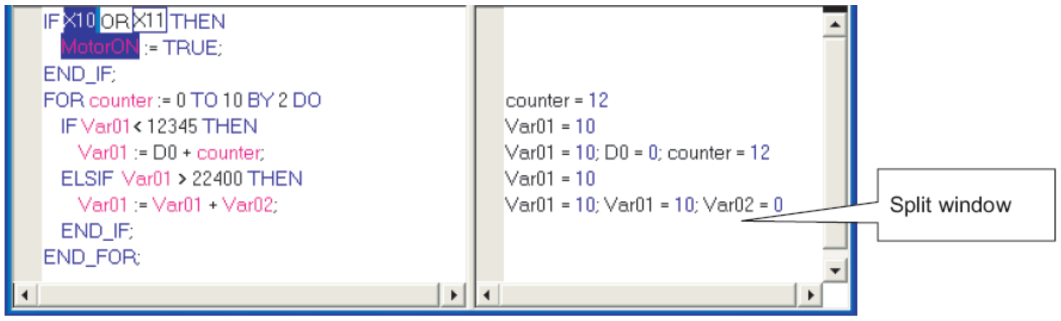
```
IF X10 OR X11 THEN
  MotorOn := TRUE;
END_IF;
```


<TRUE>

```
IF X10 OR X11 THEN
  MotorOn := TRUE;
END_IF;
```

Numeric values can be monitored in the right half of the window. All numeric variables in the program will be displayed in the right window automatically, and the value will be listed next to the variable name or address. Values are displayed next to each line where the label was used in the structured text program, so many labels may be listed more than once in the right window.

The picture below demonstrates monitoring in a structured text program.



```
IF X10 OR X11 THEN
  MotorON := TRUE;
END_IF;
FOR counter := 0 TO 10 BY 2 DO                     counter = 12
  IF Var01 < 12345 THEN                            Var01 = 10
    Var01 := D0 + counter;                         Var01 = 10; D0 = 0; counter = 12
  ELSIF Var01 > 22400 THEN                         Var01 = 10
    Var01 := Var01 + Var02;                        Var01 = 10; Var01 = 10; Var02 = 0
  END_IF;
END_FOR;
```

Notes

13.7 EXERCISE – Structured Text

Write a structured text program to perform the following tasks.

Create a new structured project for this exercise.

1. Write a data table storing a number from D0 on the operator interface each time the trigger (M0) is pressed. Use the FIFW command to build this table, and start the table at D100.

2. Only allow this table to store 10 samples. When 10 samples exist and the data store trigger is pressed, use the FIFR command to remove the oldest entry in the table before storing the new value.

3. Enable the operator to reset all of the data in the table with the reset button (M1) by writing an FMOV instruction.

4. Calculate the average of all of the entries in the table. The MEAN instruction can be used to calculate the average. Store the result of this calculation in D120.

5. Search the minimum value in the table, using the MIN instruction, and store it to D122.

6. Search the maximum value in the table, using the MAX instruction, and store it to D125.

Notes

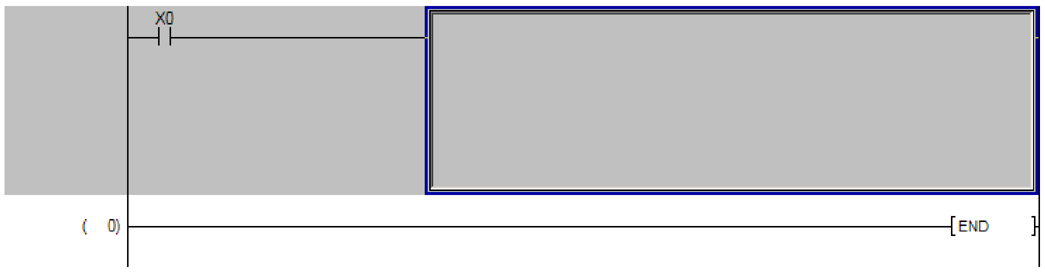
13.8 Inline Structured Text Box

It is possible with GX Works2 to place a block of structured text code at the output end of a rung of simple ladder, in either simple or structured projects. This function is convenient for embedding small amounts of structured text in the ladder program, such as mathematical functions or character string processing.

In order to use this function, a setting must be made in the software options. In the Compile section, under Basic Setting, be sure that the 'Enable function block call from ladder to ST and from ST to ladder' option is checked. This option is checked by default.

The inline structured text block is an output, so it justifies to the right side of the ladder window. It must have a contact in front of it in order to be a valid rung of code. There cannot be more than one structured text block per rung, and no vertical lines can be drawn in front of the inline structured text box.

Select the Edit menu, then Inline Structured Text, then Insert Inline Structured Text Box to insert an ST block into the ladder. It can also be accomplished from the keyboard by pressing Ctrl-B, or by typing STB in the enter symbol dialog without choosing a ladder symbol.



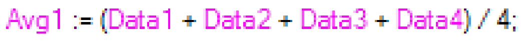
Programs can be edited in the structured text box by double clicking on the box. Editing programs in the structured text box is done the same as editing a structured text program.

When finished editing the inline structured text box, click any area outside the box, or press the ESC key. This will end the editing of the block.

Notes

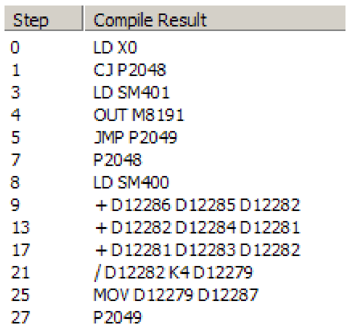
To see the results of the compile of an inline structured text block, select ‘Display Compile Result’ from the View menu. This will show the instruction list code generated to perform the actions in the structured text box.

As an example, here is the structured text to average 4 values.



```
Avg1 := (Data1 + Data2 + Data3 + Data4) / 4;
```

Here is the result of the compile in instruction list.



Step	Compile Result
0	LD X0
1	CJ P2048
3	LD SM401
4	OUT M8191
5	JMP P2049
7	P2048
8	LD SM400
9	+ D12286 D12285 D12282
13	+ D12282 D12284 D12281
17	+ D12281 D12283 D12282
21	/ D12282 K4 D12279
25	MOV D12279 D12287
27	P2049

There are some basic precautions when working with the inline structured text box. They are shown below.

- One box can be created per ladder rung
- Function blocks and inline ST boxes cannot be used in the same ladder rung
- Ladder cannot be edited while uncompiled ST boxes exist
- Up to 2048 characters can be entered into an inline ST box
- Counters, timers, pointers, structures, arrays, and function blocks cannot be used in the inline ST box
- To delete an inline ST box, select the entire rung, including the left base line
- Up to 100 structured text boxes can be inserted into a program, with a maximum of 400 per project for Q Series or L Series

Notes

13.9 EXERCISE – Inline Structured Text Box

Add a simple ladder program to the project from the previous exercise.

In this program, perform conversion from Celsius to Fahrenheit on a value stored via the operator interface screen. The formula was discussed at the beginning of this lesson.

To verify proper operation, test with the following values:

_____°C = ___________°F

_____°C = ___________°F

To ensure the operator interface communication, use the following addresses.

- Celsius temperature input to D200
- Fahrenheit temperature to be stored to D210

Notes

APPENDIX

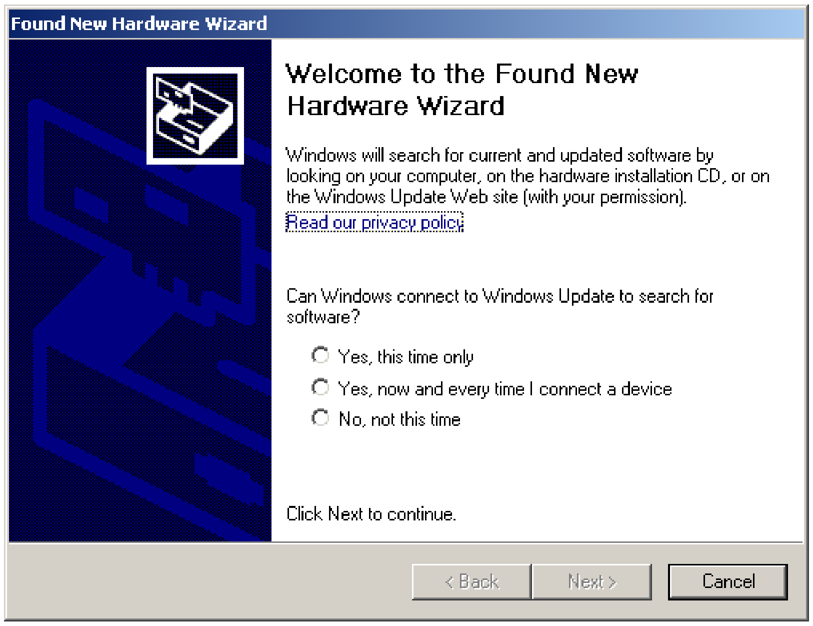
Notes

Notes

APPENDIX 1 – L Series USB Driver Installation

The following steps must be followed to install the L Series USB driver. The files required are copied to the hard drive during the installation of GX Works2, so no CDs or diskettes are required.

When the cable is first connected, Windows will show the ‘Found New Hardware’ wizard. The following screen will be displayed.



Welcome to the Found New Hardware Wizard

Windows will search for current and updated software by looking on your computer, on the hardware installation CD, or on the Windows Update Web site (with your permission).

Read our privacy policy

Can Windows connect to Windows Update to search for software?

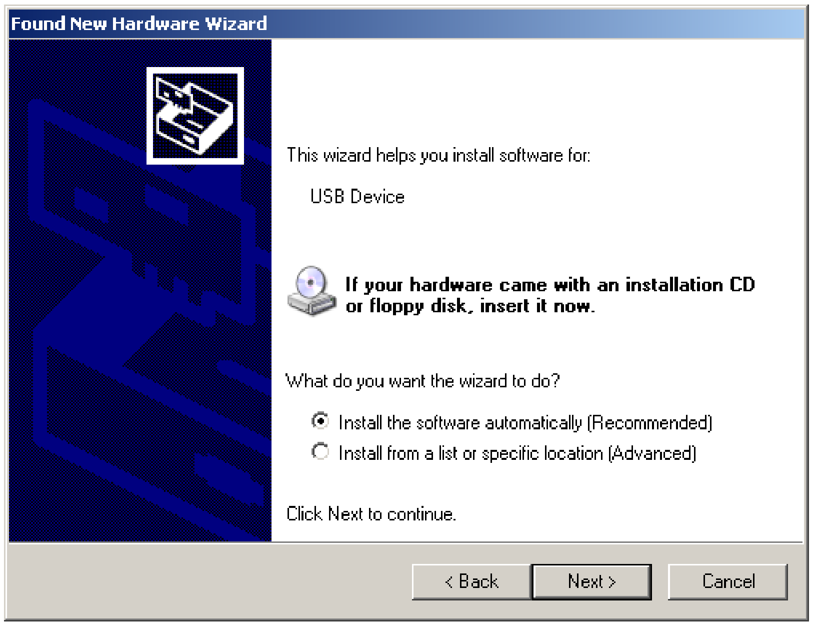
- Yes, this time only
- Yes, now and every time I connect a device
- No, not this time

Click Next to continue.

Select the ‘No, not this time’ option and click Next.

Notes

The next screen asks how to find the driver to be installed.



This wizard helps you install software for:

USB Device

If your hardware came with an installation CD or floppy disk, insert it now.

What do you want the wizard to do?

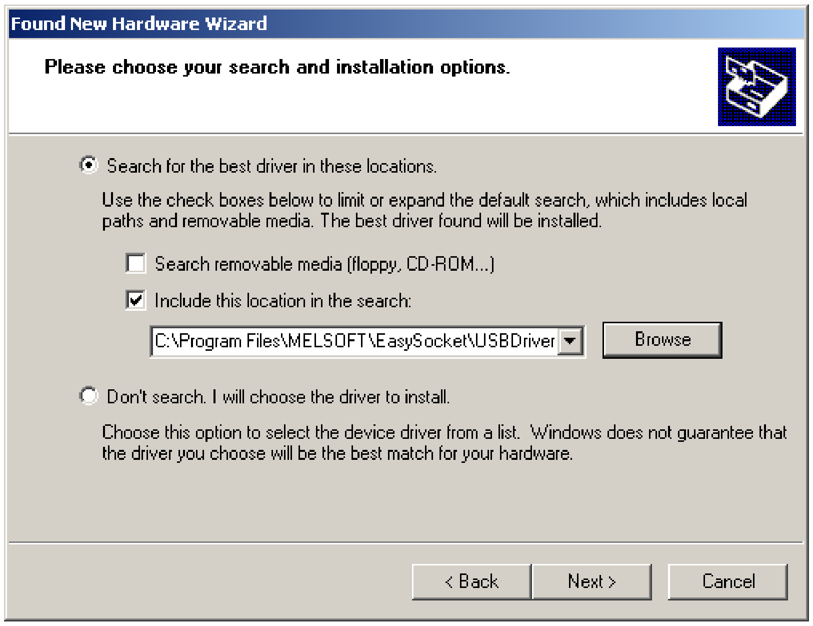
[✓ Install the software automatically (Recommended)]
Install from a list or specific location (Advanced)

Click Next to continue.

Choose ‘Install from a list or specific location (Advanced) and click Next.

Notes

The next screen will ask for the location of the driver to install.


Found New Hardware Wizard

Please choose your search and installation options.

(●) Search for the best driver in these locations.

Use the check boxes below to limit or expand the default search, which includes local paths and removable media. The best driver found will be installed.

☐ Search removable media (floppy, CD-ROM...)

☑ Include this location in the search:

C:\Program Files\MELSOFT\EasySocket\USBDriver

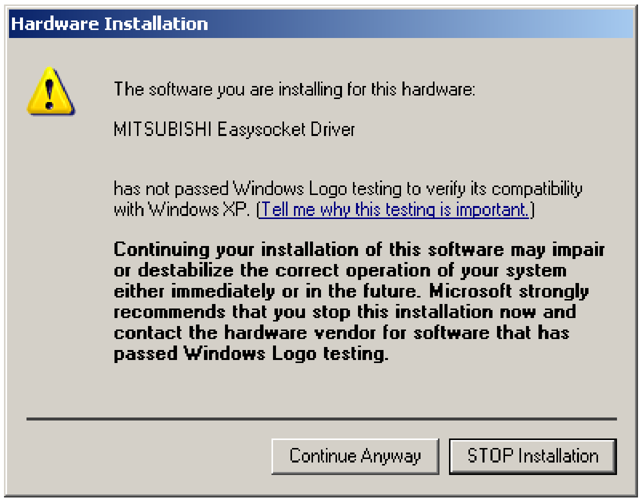
( ) Don't search. I will choose the driver to install.

Choose this option to select the device driver from a list. Windows does not guarantee that the driver you choose will be the best match for your hardware.

Ensure the 'Include this location in the search' box is checked, and click Browse. Find the directory **C:\Program Files\MELSEC\EasySocket\USBDrivers** and click OK. This will return you to the screen shown above. Click Next to move on.

Notes

The next window will state that this driver has not passed Windows XP Logo testing.



The software you are installing for this hardware:

MITSUBISHI Easysocket Driver

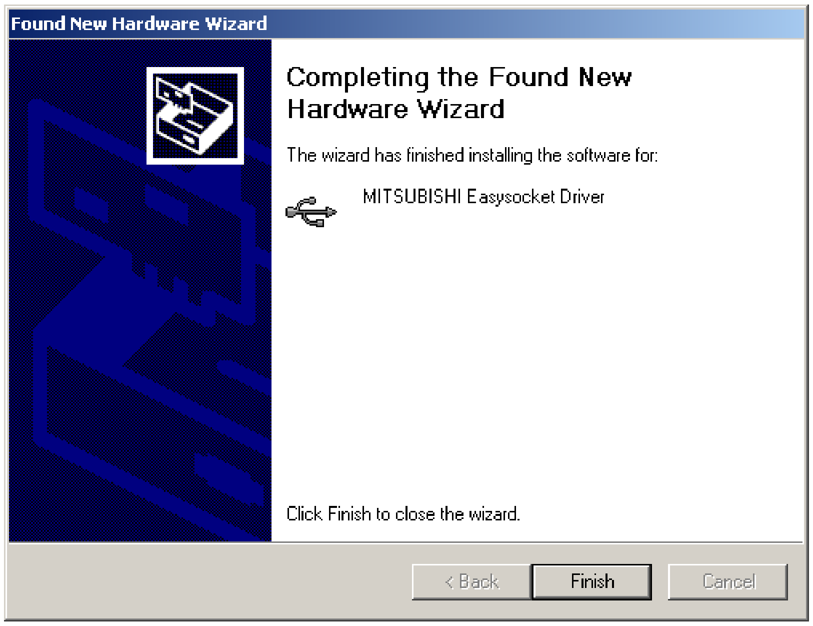
has not passed Windows Logo testing to verify its compatibility with Windows XP. (Tell me why this testing is important.)

Continuing your installation of this software may impair or destabilize the correct operation of your system either immediately or in the future. Microsoft strongly recommends that you stop this installation now and contact the hardware vendor for software that has passed Windows Logo testing.

Click ‘Continue Anyway’ to proceed with the installation.

Notes

The final screen indicates successful installation of the USB drivers.


Found New Hardware Wizard

Completing the Found New Hardware Wizard

The wizard has finished installing the software for:

MITSUBISHI Easysocket Driver

Click Finish to close the wizard.

Notes