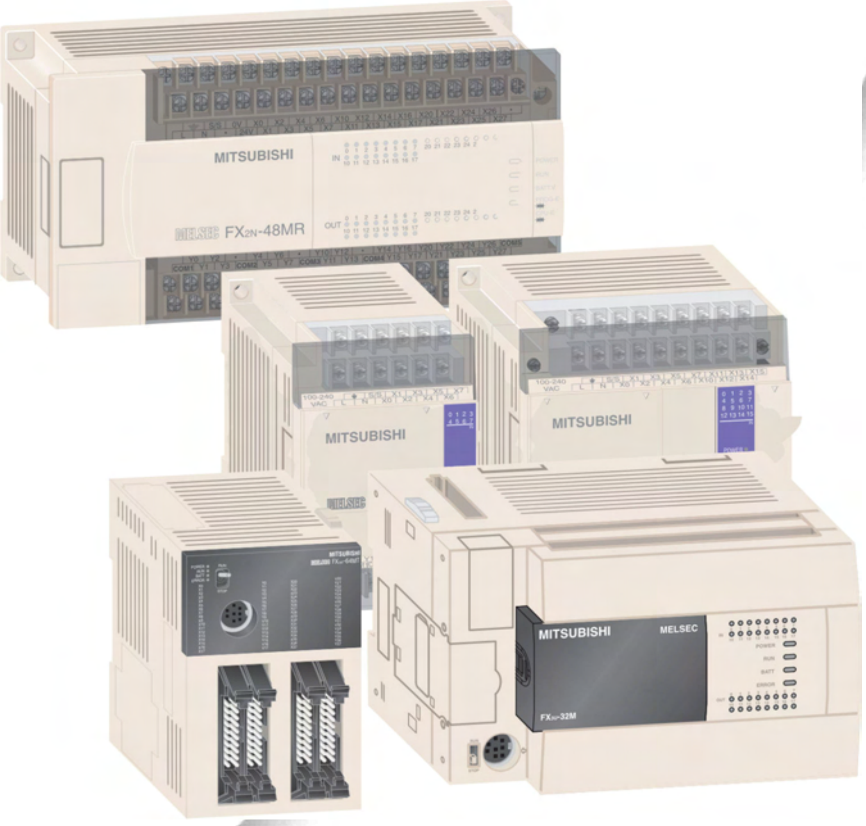
MITSUBISHI
ELECTRIC
FACTORY AUTOMATION

FX Family

Programmable Logic Controllers

Training Manual



GX IEC Developer

About this Manual

The texts, illustrations and examples in this manual only explain the installation, operation and use of the GX IEC Developer programming package.

.

If you have questions about the programming and operation of the programmable logic controllers mentioned in this manual please contact your dealer or one of our distributors (see back cover). Up-to-date information and answers to frequently-asked questions can be found on the Mitsubishi website at www.mitsubishi-automation.com.

MITSUBISHI ELECTRIC EUROPE B.V. reserves the right to make changes to this manual or the technical specifications of its products at any time without notice.

Training Manual
GX IEC Developer Programming Software Package
Art.-no.: 208661

Version			Changes / Additions / Corrections
A	06/2007	pdp	First edition
B	04/2010	pdp-dk	Chapter 2 and appendix: Addition of the base units of the FX3G and FX3UC series and of new modules for the FX3U series (FX3U-2HC, FX3U-4LC and FX3U-3A-ADP). Chapter 8 has been deleted. (The numbering of the following chapters has been shifted accordingly.) Corrections in the sections 3.2.1, 3.2.2, 3.4.1, 4.4.1, and 6.2 Changes in the chapters 5 (Program example) and 18 (Ethernet communications, former chapter 19) and in the sections 12.3.2 and 12.3.3 (Libraries). A new section 12.3.1 has been defined.

Safety Information

For qualified staff only

This manual is only intended for use by properly trained and qualified electrical technicians who are fully acquainted with automation technology safety standards. All work with the hardware described, including system design, installation, setup, maintenance, service and testing, may only be performed by trained electrical technicians with approved qualifications who are fully acquainted with the applicable automation technology safety standards and regulations.

Proper use of equipment

The programmable logic controllers are only intended for the specific applications explicitly described in this manual. Please take care to observe all the installation and operating parameters specified in the manual. All products are designed, manufactured, tested and documentated in agreement with the safety regulations. Any modification of the hardware or software or disregarding of the safety warnings given in this manual or printed on the product can cause injury to persons or damage to equipment or other property. Only accessories and peripherals specifically approved by MITSUBISHI ELECTRIC may be used. Any other use or application of the products is deemed to be improper.

Relevant safety regulations

All safety and accident prevention regulations relevant to your specific application must be observed in the system design, installation, setup, maintenance, servicing and testing of these products. The regulations listed below are particularly important. This list does not claim to be complete; however, you are responsible for knowing and applying the regulations applicable to you.

- VDE Standards
  - VDE 0100
    (Regulations for electrical installations with rated voltages up to 1,000V)
  - VDE 0105
    (Operation of electrical installations)
  - VDE 0113
    (Electrical systems with electronic equipment)
  - VDE 0160
    (Configuration of electrical systems and electrical equipment)
  - VDE 0550/0551
    (Regulations for transformers)
  - VDE 0700
    (Safety of electrical appliances for household use and similar applications)
  - VDE 0860
    (Safety regulations for mains-powered electronic appliances and their accessories for household use and similar applications)
- Fire prevention regulations
- Accident prevention regulations
  - VBG No. 4 (Electrical systems and equipment)

Safety warnings in this manual

In this manual special warnings that are important for the proper and safe use of the products are clearly identified as follows:

⚠	**DANGER:** ***Personnel health and injury warnings. Failure to observe the precautions described here can result in serious health and injury hazards.***

⚠	***CAUTION:*** ***Equipment and property damage warnings. Failure to observe the precautions described here can result in serious damage to the equipment or other property.***

General safety information and precautions

The following safety precautions are intended as a general guideline for using the PLC together with other equipment. These precautions must always be observed in the design, installation and operation of all control systems.

CAUTION:

- ***Observe all safety and accident prevention regulations applicable to your specific application. Installation, wiring and opening of the assemblies, components and devices may only be performed with all power supplies disconnected.***
- ***Assemblies, components and devices must always be installed in a shockproof housing fitted with a proper cover and protective equipment.***
- ***Devices with a permanent connection to the mains power supply must be integrated in the building installations with an all-pole disconnection switch and a suitable fuse.***
- ***Check power cables and lines connected to the equipment regularly for breaks and insulation damage. If cable damage is found, immediately disconnect the equipment and the cables from the power supply and replace the defective cabling.***
- ***Before using the equipment for the first time check that the power supply rating matches that of the local mains power.***
- ***Residual current protective devices pursuant to DIN VDE Standard 0641 Parts 1-3 are not adequate on their own as protection against indirect contact for installations with positioning drive systems. Additional and/or other protection facilities are essential for such installations.***
- ***EMERGENCY OFF facilities pursuant to EN 60204/IEC 204 VDE 0113 must remain fully operative at all times and in all control system operating modes. The EMERGENCY OFF facility reset function must be designed so that it cannot cause an uncontrolled or undefined restart.***
- ***You must also implement hardware and software safety precautions to prevent the possibility of undefined control system states caused by signal line cable or core breaks.***
- ***All relevant electrical and physical specifications must be strictly observed and maintained for all the modules in the installation.***

Table of Contents

1 Course Overview and Requirements

2 The Hardware

3 Programming

4 Building a Project

5 Program Example

6 Functions and Function Blocks

7 Advanced Monitoring Functions

8 Device Edit

9 Online Mode

10 Data Unit Types (DUT)

11 Arrays

12 Working with Libraries

13 Security

14 Sequential Function Chart - SFC

15 IEC Instruction List

16 IEC Structured Text

17 PROFIBUS/DP Communication

18 Ethernet Communications

A Appendix

1 Course Overview and Requirements

This course has been specially produced as an introduction to Mitsubishi's FX family utilising the GX IEC Developer software package.

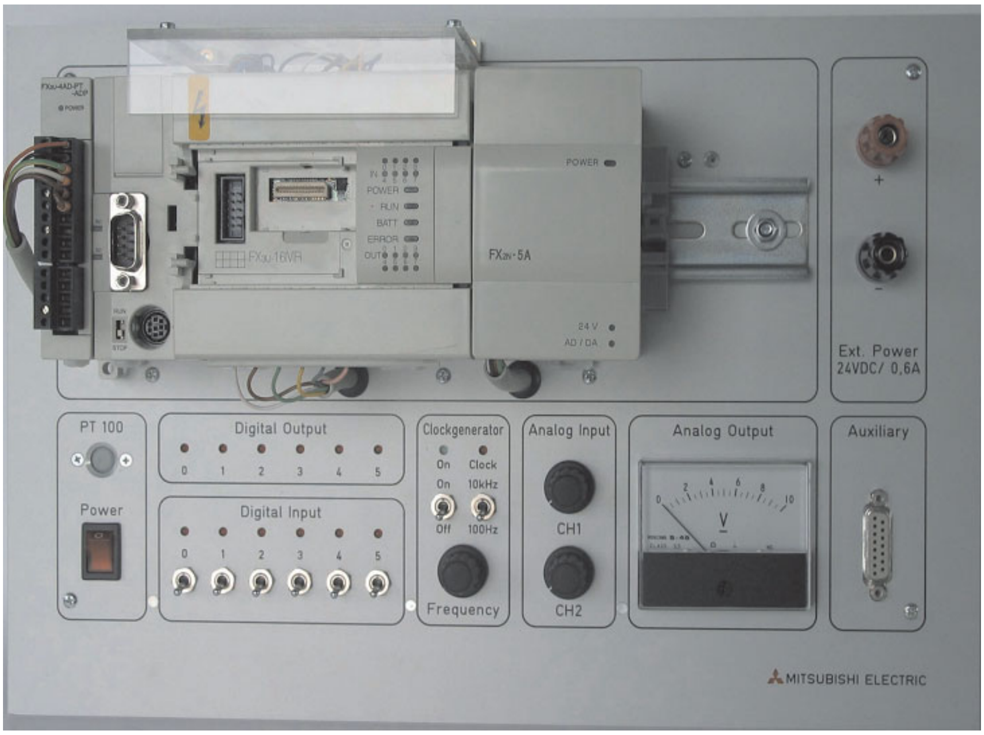
The course content has been selectively produced to provide an introduction into the functionality of the Mitsubishi range of FX PLC's, together with the GX IEC Developer programming system. The second section deals with the PLC hardware configuration and operation, whilst the remainder covers the use of Mitsubishi's IEC61131-3 programming system, which is illustrated using worked examples.

It is assumed that student will have a sound working knowledge of the Microsoft Windows operating environment.

1.1 Modular PLC Training Hardware

There are various models of training rigs for Mitsubishis FX family. Most exercises within this training manual are based around use of the facilities offered in these training systems. The examples used in these course notes assume the following configuration:

- 6 Digital input simulator switches: X0–X5
- Variable clock input (1 to 100 Hz and 0.1 to10 kHz): X7
- 6 Digital output LED indicators: Y0–Y5
- 1 Special function block FX2N-5A with 4 analog inputs and 1 analog output
- 1 Temperature acquisition special adapter FX3U-4AD-PT-ADP



Thus, adjustments according to other training simulators may be accommodated with appropriate address alterations to the example code provided this training document.

2 The Hardware

2.1 General Introduction to PLCs

2.1.1 History & Development

Bedford Associates, founded by Richard Morley introduced the first Programmable Logic Controller in 1968. This PLC was known as the Modular Digital Controller from which the MODICON Company derived its name.

Programmable Logic Controllers were developed to provide a replacement for large relay based control panels. These systems were inflexible requiring major rewiring or replacement whenever the control sequence was to be changed.

The development of the Microprocessor from the mid 1970's have allowed Programmable Logic Controllers to take on more complex tasks and larger functions as the speed of the processor increased. It is now common for PLC's to provide the heart of the control functions within a system often integrated with SCADA (Supervisory Control And Data Acquisition), HMI (Human Machine Interfaces), Expert Systems and Graphical User Interfaces (GUI). The requirements of the PLC have expanded to providing control, data processing and management functionality.

2.1.2 The initial specification for the PLC

- Easily programmed and reprogrammed in plant to enable its sequence of operations, to be altered.
- Easily maintained and repaired – preferably using 'plug-in' cards or modules.
- Able to withstand the rigorous Environmental, Mechanical and Electrical conditions, found in plant environments.
- Smaller than its relay and "discrete solid state" equivalents.
- Cost effective in comparison with "discrete solid state" and relay based systems.

2.1.3 Comparison of PLC and Relay Systems

Characteristic	PLC	Relay
Price per function	Low	Low - If equivalent relay program uses more than 10 relays
Physical size	Very compact	Bulky
Operating speed	Fast	Slow
Electrical noise immunity	Good	Excellent
Construction	Easy to program	Wiring - time consuming
Advanced instructions	Yes	No
Changing the control sequence	Very simple	Very difficult – requires changes to wiring
Maintenance	Excellent PLC's rarely fail	Poor - relays require constant maintenance

2.1.4 Programming

Ladder Logic

PLC’s had to be maintainable by technicians and electrical personnel. To support this, the programming language of Ladder Logic was developed. Ladder Logic is based on the relay and contact symbols technicians were used to through wiring diagrams of electrical control panels.

The documentation for early PLC Programs was either non existent or very poor, just providing simple addressing or basic comments, making large programs difficult to follow. This has been greatly improved with the development of PLC Programming packages such as Mitsubishi’s Windows based, **GX Developer**.

Until recently there has been no formal programming standard for PLC’s. The introduction of the **IEC 61131-3** Standard in 1998 provides a more formal approach to coding. Mitsubishi Electric has developed a programming package, “**GX IEC Developer**” (Covered in detail later in this document.). This enables IEC compliant coding to be adopted.

2.1.5 Human Machine Interfaces

The early programmable logic controllers interfaced with the operator in much the same way as the relay control panel, via push-buttons and switches for control and lamps for indication.

The introduction of the Personal Computer (PC) in the 1980’s allowed for the development of a computer based interface to the operator, these where initially via simple Supervisory Control And Data Acquisition (SCADA) systems and more recently via Dedicated Operator Control Panels, known as Human Machine Interfaces (HMI). It is now common place to see PLC’s heavily integrated with these products to form user friendly control system solutions.

Mitsubishi offer a very wide range of HMI and SCADA products to suit a variety of operator Interface applications.

It is now commonplace to find HMI’s integrated into PLC based control systems, providing the operator interface functionality.

2.2 What is a PLC?

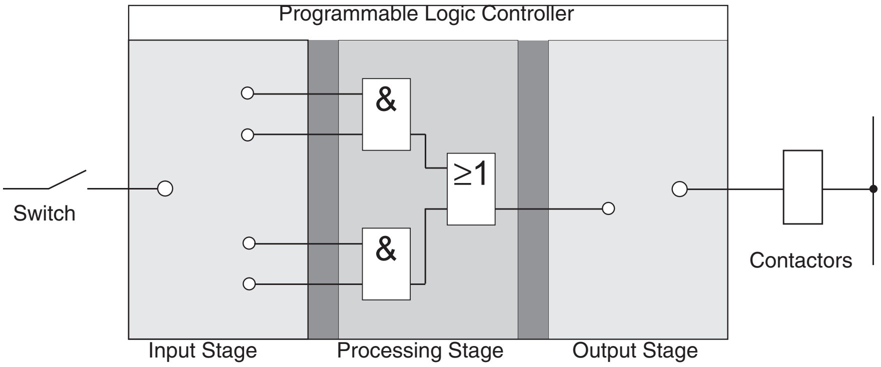
In contrast to conventional controllers with functions determined by their physical wiring the functions of programmable logic controllers or PLCs are defined by a program. PLCs also have to be connected to the outside world with cables, but the contents of their program memory can be changed at any time to adapt their programs to different control tasks.

Programmable logic controllers input data, process it and then output the results. This process is performed in three stages:

- an input stage,
- a processing stage

and

- an output stage



The input stage

The input stage passes control signals from switches, buttons or sensors on to the processing stage.

The signals from these components are generated as part of the control process and are fed to the inputs as logical states. The input stage passes them on to the processing stage in a pre-processed format.

The processing stage

In the processing stage the pre-processed signals from the input stage are processed and combined with the help of logical operations and other functions. The program memory of the processing stage is fully programmable. The processing sequence can be changed at any time by modifying or replacing the stored program.

The output stage

The results of the processing of the input signals by the program are fed to the output stage where they control connected switchable elements such as contactors, signal lamps, solenoid valves and so on.

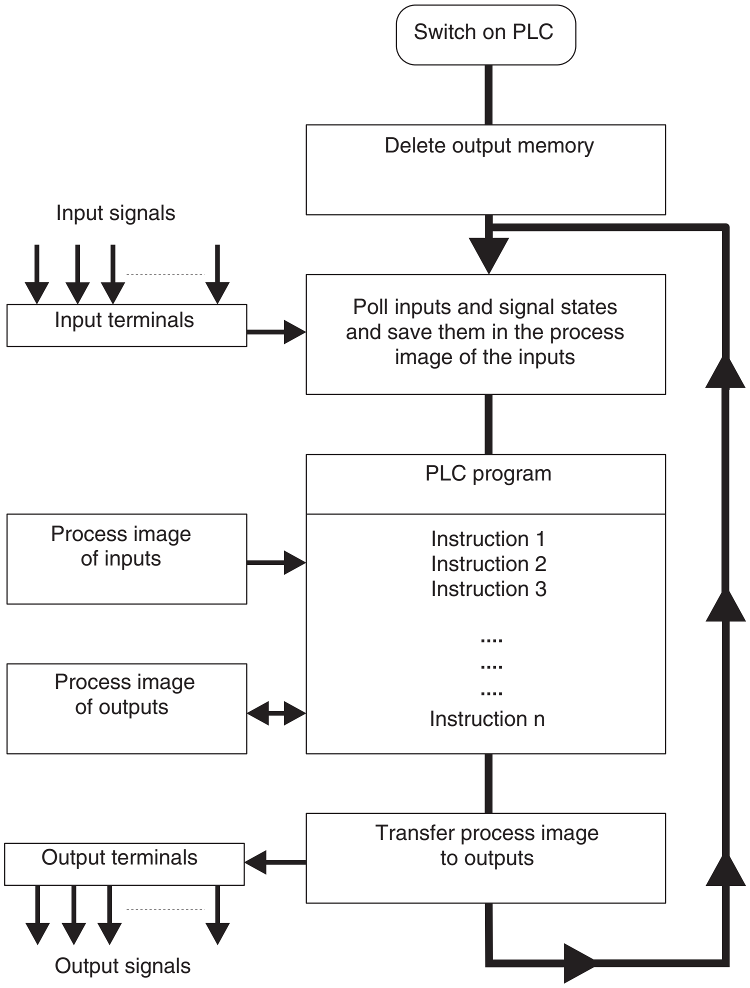
2.3 How PLCs Process Programs

A PLC performs its tasks by executing a program that is usually developed outside the controller and then transferred to the controller's program memory. Before you start programming it is useful to have a basic understanding of how PLCs process these programs.

A PLC program consists of a sequence of instructions that control the functions of the controller. The PLC executes these control instructions sequentially, i.e. one after another. The entire program sequence is cyclical, which means that it is repeated in a continuous loop. The time required for one program repetition is referred to as the program cycle time or period.

Process image processing

The program in the PLC is not executed directly on the inputs and outputs, but on a "process image" of the inputs and outputs:



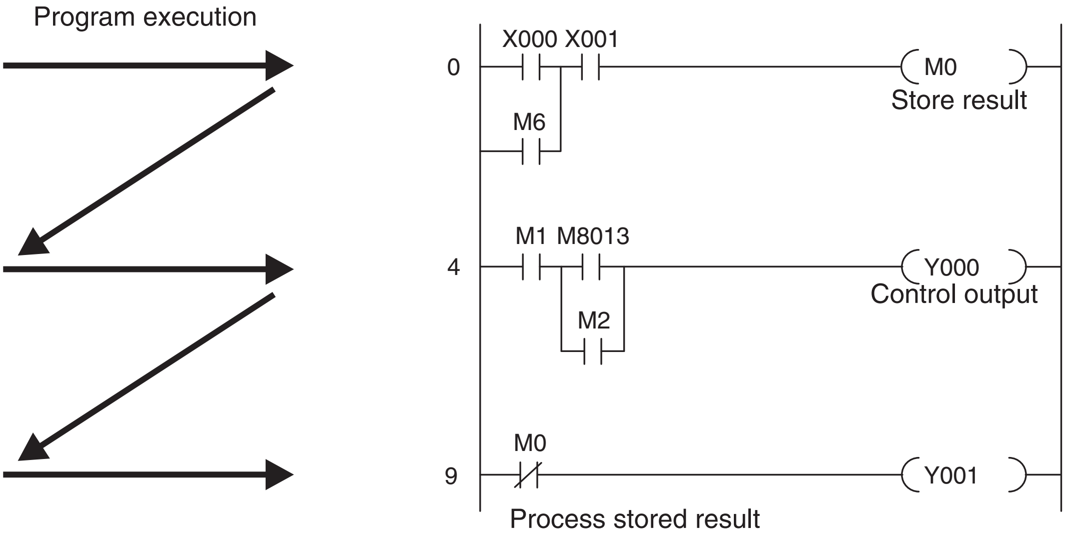
Input process image

At the beginning of each program cycle the system polls the signal states of the inputs and stores them in a buffer, creating a "process image" of the inputs.

Program execution

After this the program is executed, during which the PLC accesses the stored states of the inputs in the process image. This means that any subsequent changes in the input states will not be registered until the **next** program cycle!

The program is executed from top to bottom, in the order in which the instructions were programmed. Results of individual programming steps are stored and can be used during the current program cycle.



Output process image

Results of logical operations that are relevant for the outputs are stored in an output buffer – the output process image. The output process image is stored in the output buffer until the buffer is rewritten. After the values have been written to the outputs the program cycle is repeated.

Differences between signal processing in the PLC and in hard-wired controllers

In hard-wired controllers the program is defined by the functional elements and their connections (the wiring). All control operations are performed simultaneously (parallel execution). Every change in an input signal state causes an instantaneous change in the corresponding output signal state.

In a PLC it is not possible to respond to changes in input signal states until the next program cycle after the change. Nowadays this disadvantage is largely compensated by very short program cycle periods. The duration of the program cycle period depends on the number and type of instructions executed.

2.4 The MELSEC FX Family

MELSEC means **M**ITSUBISHI **EL**ECTRIC **SE**QUEN**C**ER. The compact micro-controllers of the MELSEC FX series provide the foundation for building economical solutions for small to medium-sized control and positioning tasks requiring 10 to 256 integrated inputs and outputs in applications in industry and building services.

With the exception of the FX1S all the controllers of the FX series can be expanded to keep pace with the changes in the application and the user's growing requirements.

Network connections are also supported. This makes it possible for the controllers of the FX family to communicate with other PLCs and controller systems and HMIs (Human-Machine Interfaces and control panels). The PLC systems can be integrated both in MITSUBISHI networks as local stations and as slave stations in open networks like PROFIBUS/DP.

In addition to this you can also build multi-drop and peer-to-peer networks with the controllers of the MELSEC FX family.

The FX1N, FX2N, FX3G, FX3UC and FX3U have modular expansion capabilities, making them the right choice for complex applications and tasks requiring special functions like analog-digital and digital-analog conversion and network capabilities.

All the controllers in the series are part of the larger MELSEC FX family and are fully compatible with one another.

Specifications	FX1S	FX1N	FX2N	FX2NC	FX3G	FX3U	FX3UC
Max integrated I/O points	30	60	128	96	60	128	96
Expansion capability (max. possible I/Os)	34	132	256	256	256	384	384
Program memory (steps)	2000	8000	16000	16000	32000	64000	64000
Cycle time per logical instruction (μs)	0.55–0.7	0.55–0.7	0.08	0.08	0.21/0.42	0.065	0.065
No. of instructions (standard / step ladder / special function)	27 / 2 / 85	27 / 2 / 89	27 / 2 / 107	27 / 2 / 107	29 / 2 / 123	27 / 2 / 209	29 / 2 / 209
Max. special function modules connectable	—	2	8	4	4[1] / 8[2]	10[1] / 8[2]	6[1] / 8[2]

[1] Connectable to the left side

[2] Connectable to the right side

2.5 Selecting the Right Controller

The base units of the MELSEC FX family are available in a number of different versions with different power supply options and output technologies. You can choose between units designed for power supplies of 100–240 V AC, 24 V DC or 12–24 V DC, and between relay and transistor outputs.

Series	I/Os	Type	No. of inputs	No. of outputs	Power supply	Output type
FX1S	10	FX1S-10 M□-□□	6	4	24 V DC or 100–240 V AC	Transistor or relay
	14	FX1S-14 M□-□□	8	6		
	20	FX1S-20 M□-□□	12	8		
	30	FX1S-30 M□-□□	16	14		
FX1N	14	FX1N-14 M□-□□	8	6	12–24 V DC or 100–240 V AC	Transistor or relay
	24	FX1N-24 M□-□□	14	10		
	40	FX1N-40 M□-□□	24	16		
	60	FX1N-60 M□-□□	36	24		
FX2N	16	FX2N-16 M□-□□	8	8	24 V DC or 100–240 V AC	Transistor or relay
	32	FX2N-32 M□-□□	16	16		
	48	FX2N-48 M□-□□	24	24		
	64	FX2N-64 M□-□□	32	32		
	80	FX2N-80 M□-□□	40	40		
	128	FX2N-128 M□-□□	64	64		
FX2NC	16	FX2NC-16 M□-□□	8	8	24 V DC	Transistor or relay
	32	FX2NC-32 M□-□□	16	16		
	64	FX2NC-64 M□-□□	32	32		
	96	FX2NC-96 M□-□□	48	48		
FX3G	14	FX3G-14M□/□□□	8	6	24 V DC or 100–240 V AC	Transistor or relay
	24	FX3G-24M□/□□□	14	10		
	40	FX3G-40M□/□□□	24	16		
	60	FX3G-60M□/□□□	36	24		
FX3U	16	FX3U-16 M□-□□	8	8	24 V DC or 100–240 V AC	Transistor or relay
	32	FX3U-32 M□-□□	16	16		
	48	FX3U-48 M□-□□	24	24		
	64	FX3U-64 M□-□□	32	32		
	80	FX3U-80 M□-□□	40	40		
	128	FX3U-128 M□-□□	64	64	100–240 V AC	Transistor or relay
FX3UC	16	FX3UC-16M□/□□□	8	8	24 V DC	Transistor
	32	FX3UC-32M□/□□□	16	16		
	64	FX3UC-64M□/□□□	32	32		
	96	FX3UC-96M□/□□□	48	48		

Here are some considerations that should be taken into account when configuring a system:

- Power supply requirements

  Supply voltage: 24 V DC or 100–240 V AC

- Input/Output requirements
  - How many signals (external switch contacts, buttons and sensors) do you need to input?
  - What types of functions do you need to switch, and how many of them are there?
  - How high are the loads that the outputs need to switch?

    Choose relay outputs for switching high loads and transistor outputs for switching fast, trigger-free switching operations.

- **Special Function Modules**
  - Number of modules in system
  - External power supply requirements

2.6 Controller Design

All the controllers in the series have the same basic design. The main functional elements and assemblies are described in the glossary in the appendix.

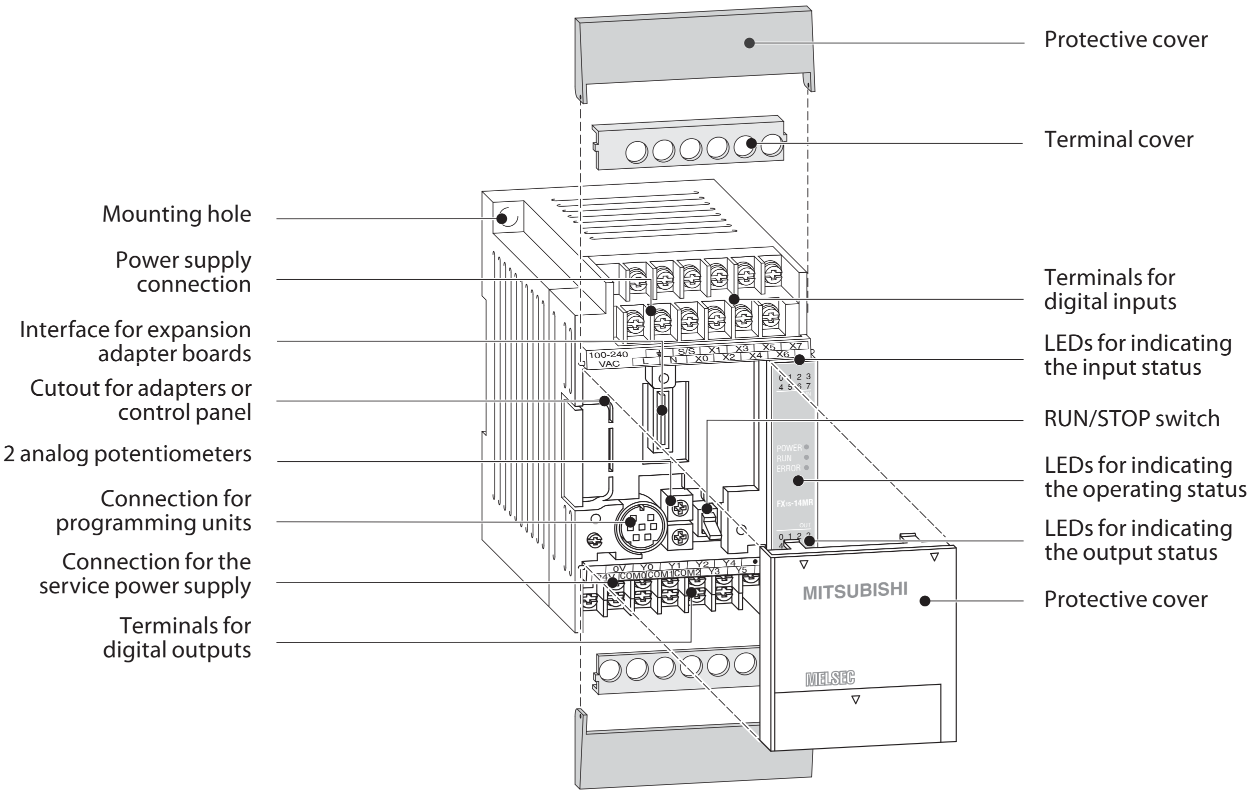
2.6.1 Input and output circuits

The **input circuits** use floating inputs. They are electrically isolated from the other circuits of the PLC with optical couplers. The **output circuits** use either relay or transistor output technology. The transistor outputs are also electrically isolated from the other PLC circuits with optical couplers.

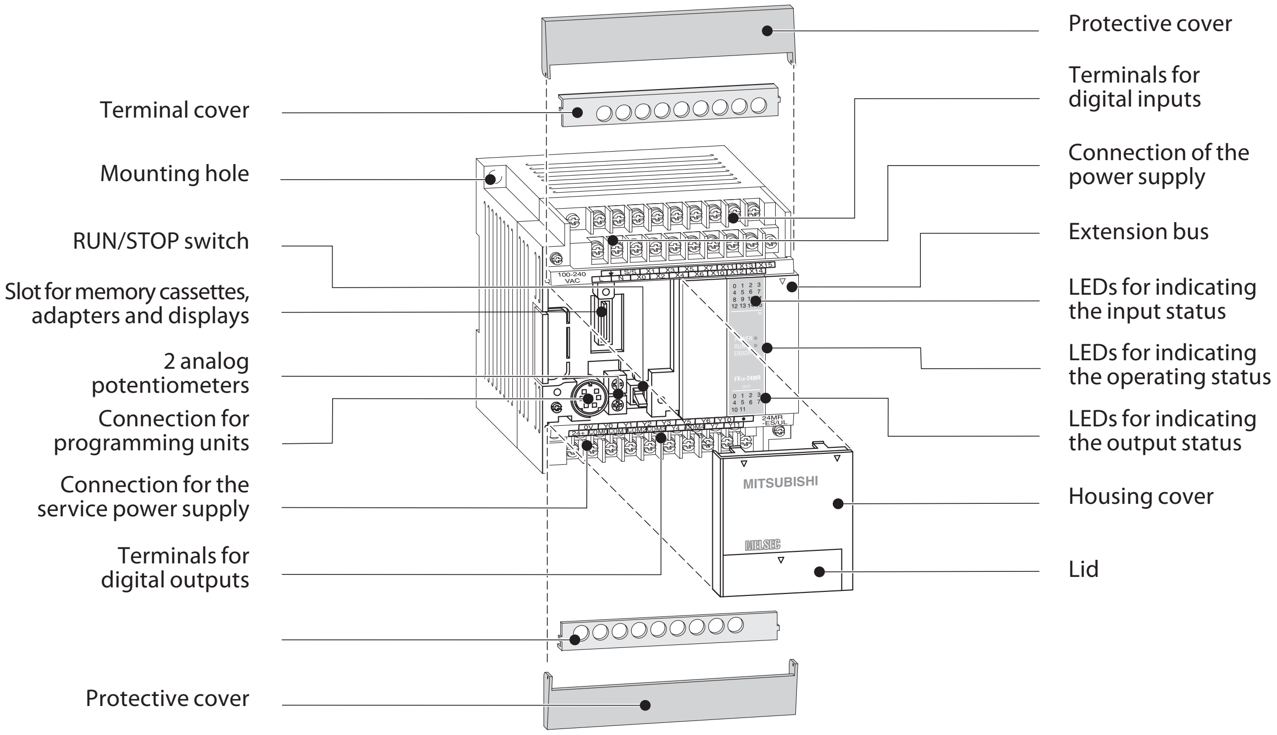
The switching voltage at all the digital inputs must have a certain value (e.g. 24 V DC). This voltage can be taken from the PLC's integrated power supply unit. If the switching voltage at the inputs is less than the rated value (e.g. <24 V DC) then the input will not be processed.

The maximum output currents are 2 A on 250 V three-phase AC and non-reactive loads with relay outputs and 0.5 A on 24 V DC and non-reactive loads.

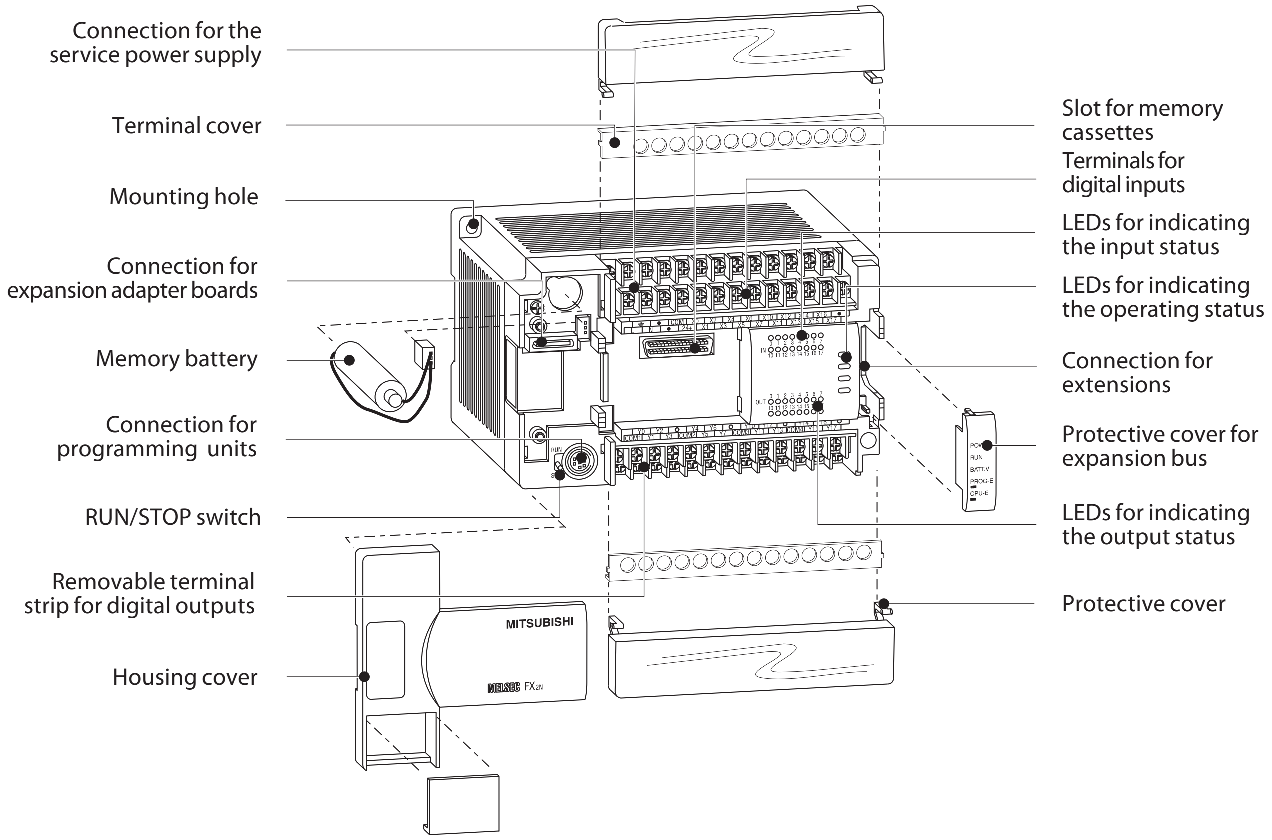
2.6.2 Layout of the MELSEC FX1S base units


Protective cover
Terminal cover
Mounting hole
Power supply connection
Terminals for digital inputs
Interface for expansion adapter boards
LEDs for indicating the input status
Cutout for adapters or control panel
RUN/STOP switch
2 analog potentiometers
LEDs for indicating the operating status
Connection for programming units
LEDs for indicating the output status
Connection for the service power supply
Protective cover
Terminals for digital outputs
100-240 VAC
MITSUBISHI
MELSEC

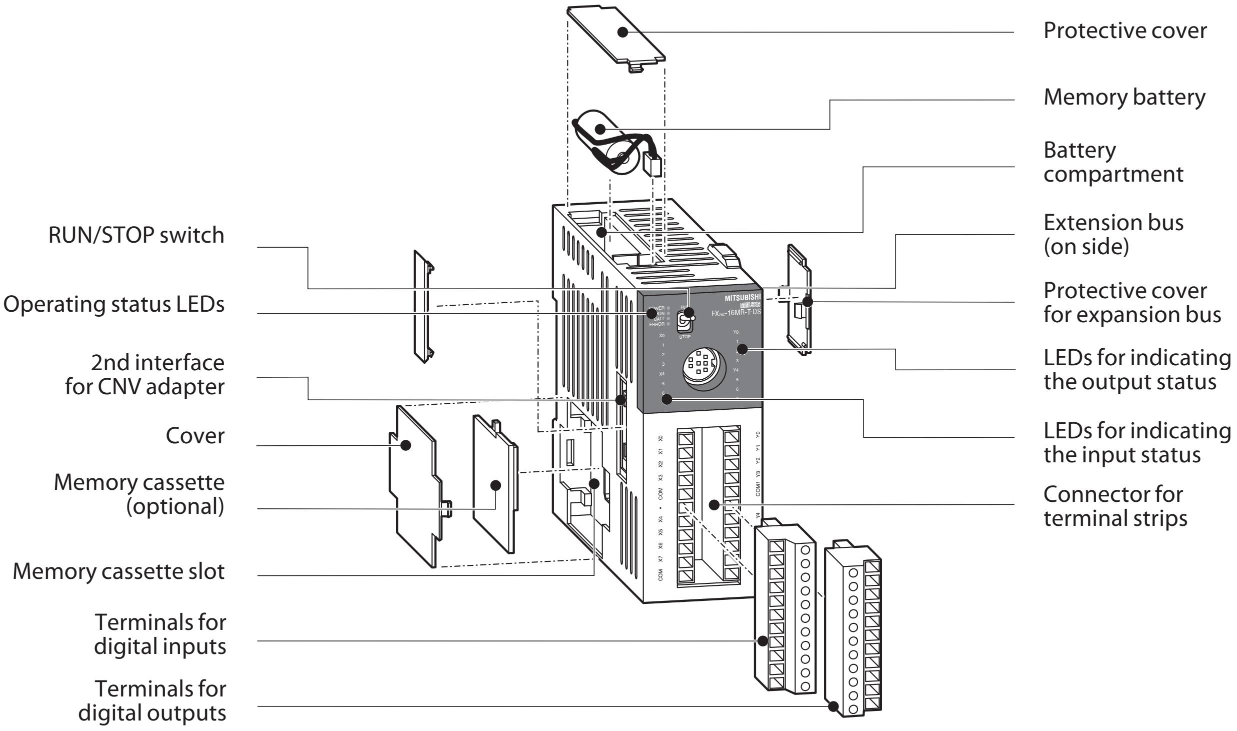
2.6.3 Layout of the MELSEC FX1N base units


Protective cover
Terminal cover
Terminals for digital inputs
Mounting hole
Connection of the power supply
RUN/STOP switch
Extension bus
Slot for memory cassettes, adapters and displays
LEDs for indicating the input status
2 analog potentiometers
LEDs for indicating the operating status
Connection for programming units
LEDs for indicating the output status
Connection for the service power supply
Housing cover
Terminals for digital outputs
Lid
Protective cover
100-240 VAC
24MR
MITSUBISHI
MELSEC

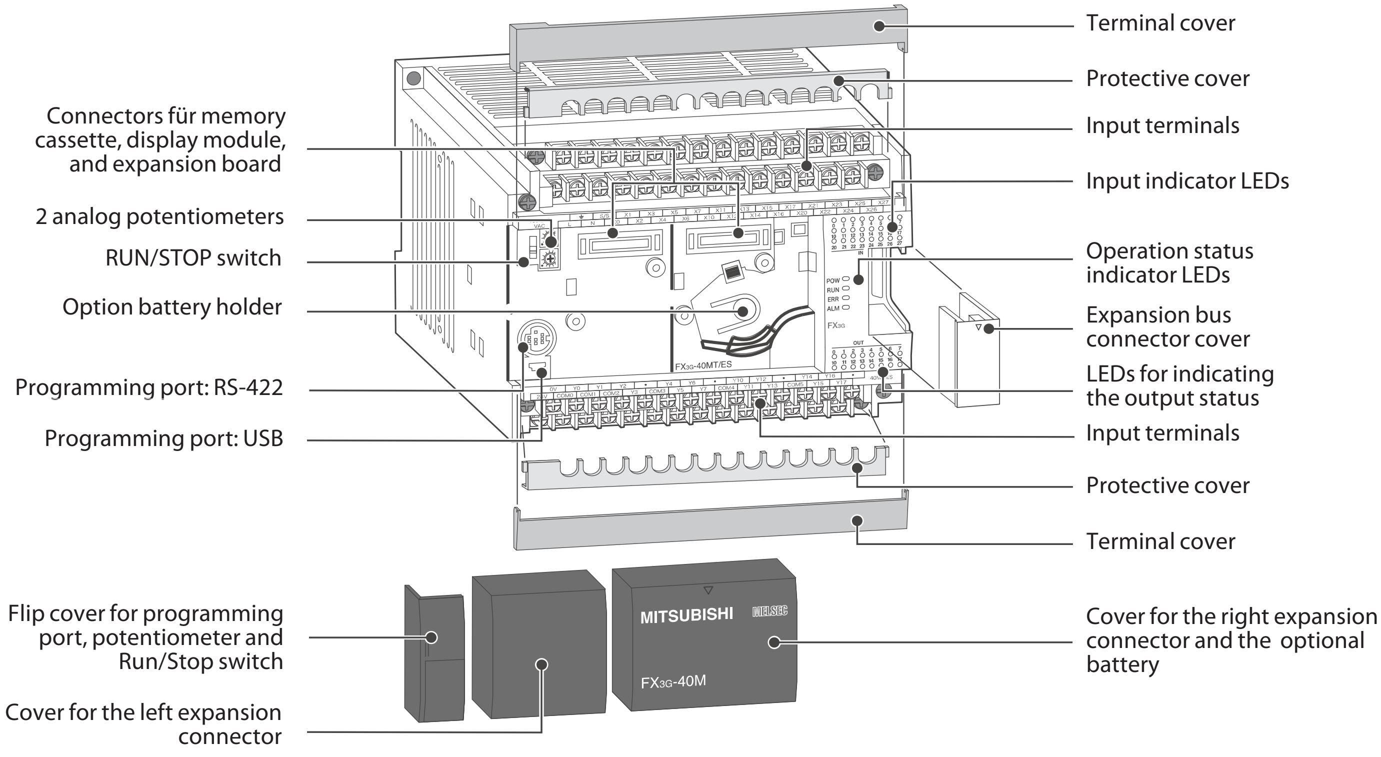
2.6.4 Layout of the MELSEC FX2N base units


Connection for the service power supply
Terminal cover
Slot for memory cassettes
Terminals for digital inputs
Mounting hole
LEDs for indicating the input status
Connection for expansion adapter boards
LEDs for indicating the operating status
Memory battery
Connection for extensions
Connection for programming units
Protective cover for expansion bus
RUN/STOP switch
LEDs for indicating the output status
Removable terminal strip for digital outputs
Protective cover
Housing cover
MITSUBISHI
MELSEC FX2N

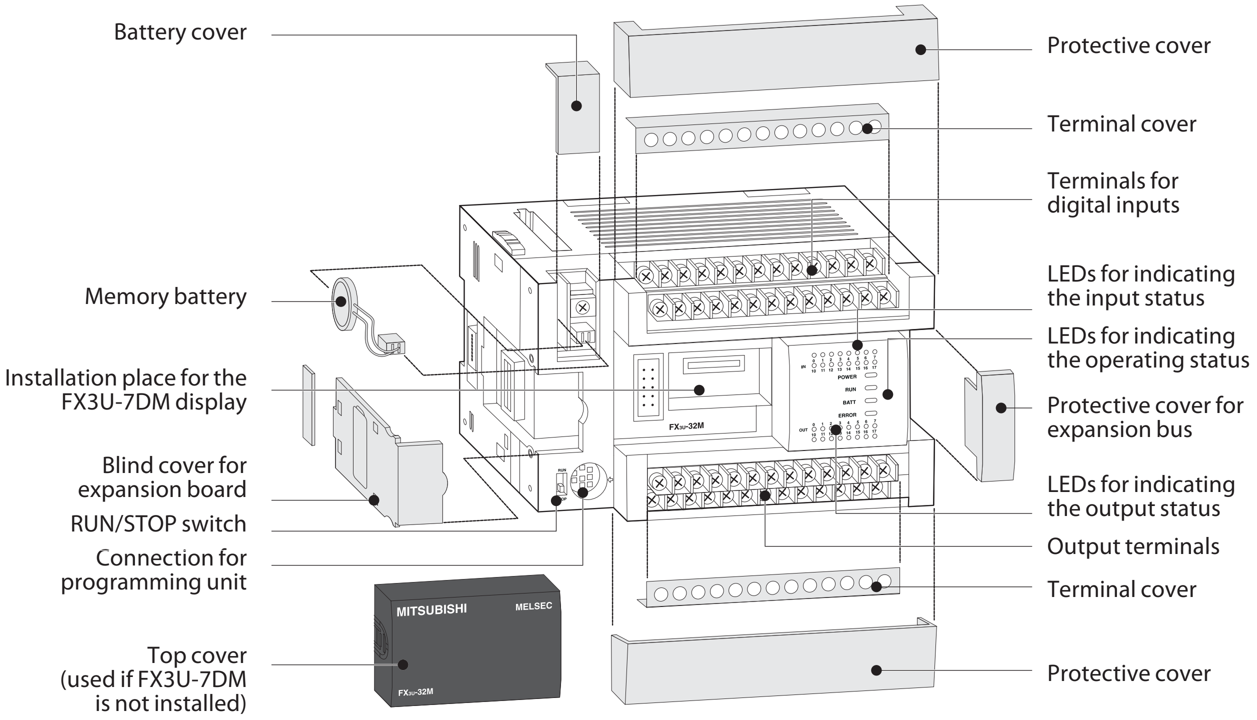
2.6.5 Layout of the MELSEC FX2NC base units


Protective cover
Memory battery
Battery compartment
RUN/STOP switch
Extension bus (on side)
Operating status LEDs
Protective cover for expansion bus
2nd interface for CNV adapter
LEDs for indicating the output status
Cover
LEDs for indicating the input status
Memory cassette (optional)
Connector for terminal strips
Memory cassette slot
Terminals for digital inputs
Terminals for digital outputs
MITSUBISHI
FX2NC-16MR-T-DS

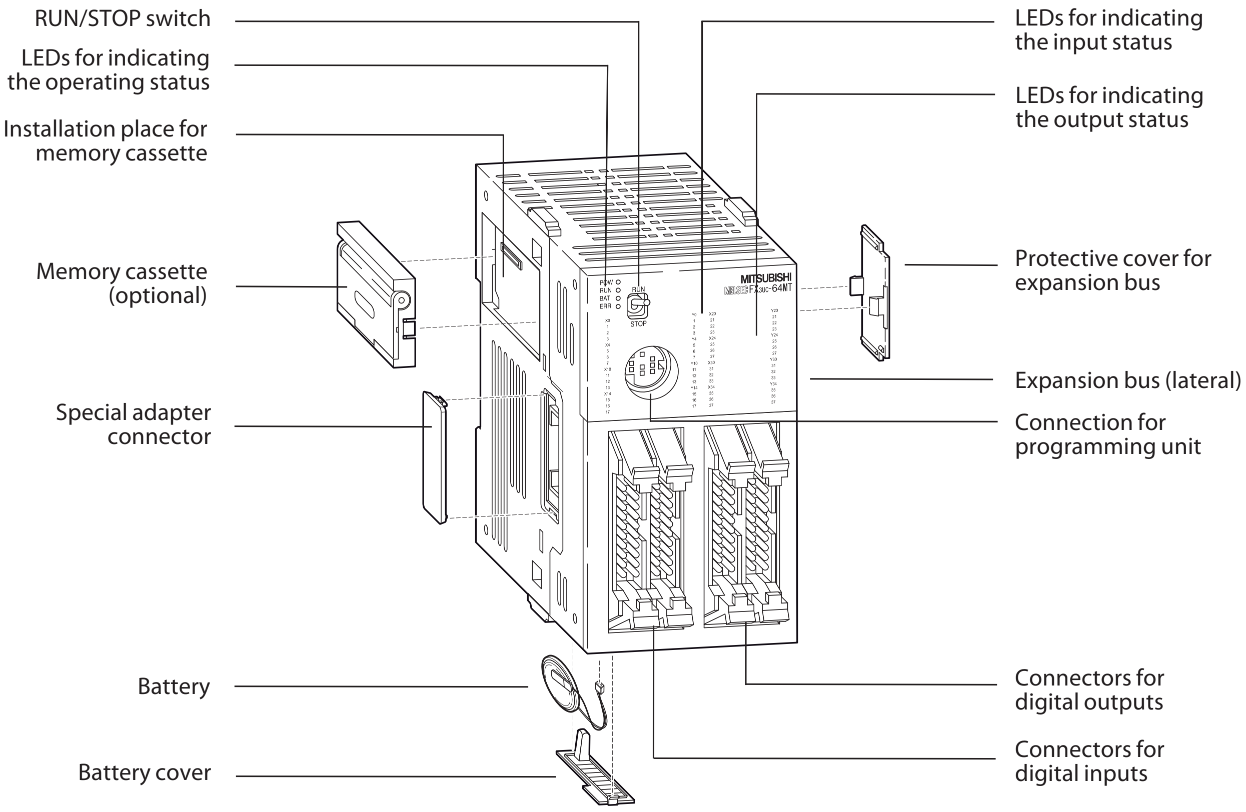
2.6.6 Layout of the MELSEC FX3G base units


Terminal cover
Protective cover
Connectors für memory cassette, display module, and expansion board
Input terminals
Input indicator LEDs
2 analog potentiometers
RUN/STOP switch
Operation status indicator LEDs
Option battery holder
Expansion bus connector cover
LEDs for indicating the output status
Programming port: RS-422
Input terminals
Programming port: USB
Protective cover
Terminal cover
FX3G-40MT/ES
Flip cover for programming port, potentiometer and Run/Stop switch
Cover for the right expansion connector and the optional battery
Cover for the left expansion connector
MITSUBISHI
FX3G-40M

2.6.7 Layout of the MELSEC FX3U base units


Battery cover
Memory battery
Installation place for the FX3U-7DM display
Blind cover for expansion board
RUN/STOP switch
Connection for programming unit
Top cover (used if FX3U-7DM is not installed)
MITSUBISHI
MELSEC
FX3U-32M
POWER
RUN
BATT
ERROR
Protective cover
Terminal cover
Terminals for digital inputs
LEDs for indicating the input status
LEDs for indicating the operating status
Protective cover for expansion bus
LEDs for indicating the output status
Output terminals
Terminal cover
Protective cover

2.6.8 Layout of the MELSEC FX3UC base units


RUN/STOP switch
LEDs for indicating the operating status
Installation place for memory cassette
Memory cassette (optional)
Special adapter connector
Battery
Battery cover
MITSUBISHI
FX3UC-64MT
RUN
STOP
LEDs for indicating the input status
LEDs for indicating the output status
Protective cover for expansion bus
Expansion bus (lateral)
Connection for programming unit
Connectors for digital outputs
Connectors for digital inputs

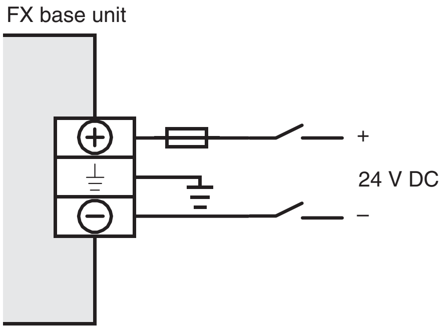
2.7 Wiring

2.7.1 Power Supply

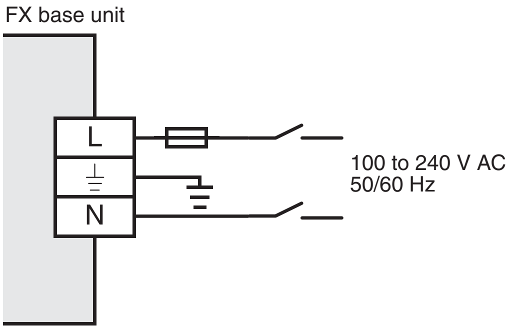
Power Supply Specifications

Specification	Units for DC Power Supply		Units for AC Power Supply
Rated voltage	12 to 24 V DC	24 V DC	100 to 240 V AC
Voltage range	10.2 to 26.4 V DC	20.4 to 26.4 V DC	85 to 264 V AC
Allowable momentary power failure time	5 ms		20 ms

Connection of units with DC power supply



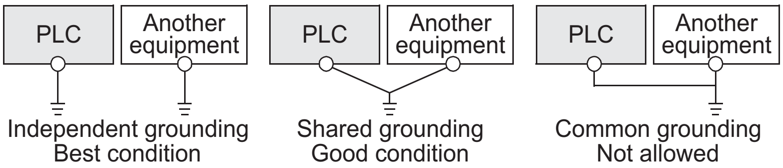
Connection of units with AC power supply



Grounding

The PLC should be grounded.

- The grounding resistance should be 100 Ω or less.
- The grounding point should be close to the PLC. Keep the grounding wires as short as possible.
- Independent grounding should be performed for best results. When independent grounding is not performed, perform "shared grounding" of the following figure.

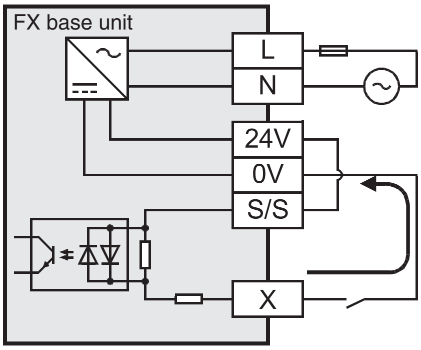


- The ground wire size should be at least 2 mm^2.

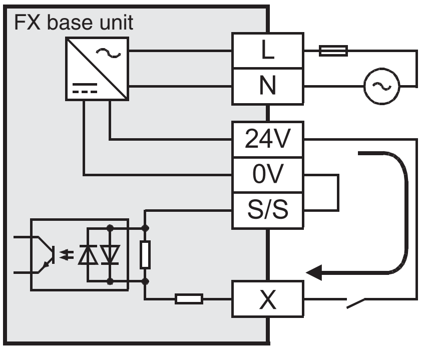
2.7.2 Wiring of Inputs

Connecting sink or source devices

The base units of the FX family series can be used with sink or source switching devices. The decision is made by the different connections of the "S/S" terminal.



In the case of the **sink input type**, the S/S terminal is connected to the 24V terminal of the service power supply or, when a DC powered base unit is used, to the positive pole of the power supply.
Sink input means that a contact wired to the input (X) or a sensor with NPN open collector transistor output connects the input of the PLC with the **negative** pole of a power supply.

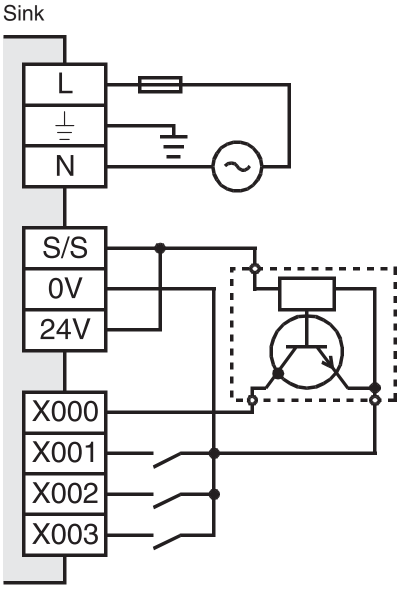


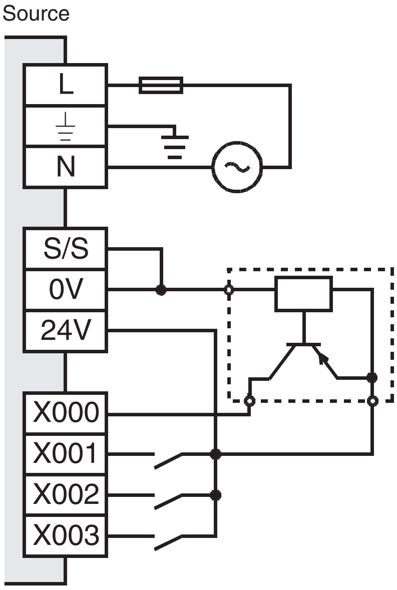
In the case of the **source input type**, the S/S terminal is connected to the 0V terminal of the service power supply or, when a DC powered base unit is used, to the negative pole of the power supply.
Source input means that a contact wired to the input (X) or a sensor with PNP open collector transistor output connects the input of the PLC with the **positive** pole of a power supply.

All inputs of a base unit or an extension unit can be either used as sink or source inputs, but it is not possible to mix sink and source inputs in one unit. Separate units in one PLC however can be set as sink or source inputs types, since the base unit and input/output powered extension units are individually set to sink or source input mode.

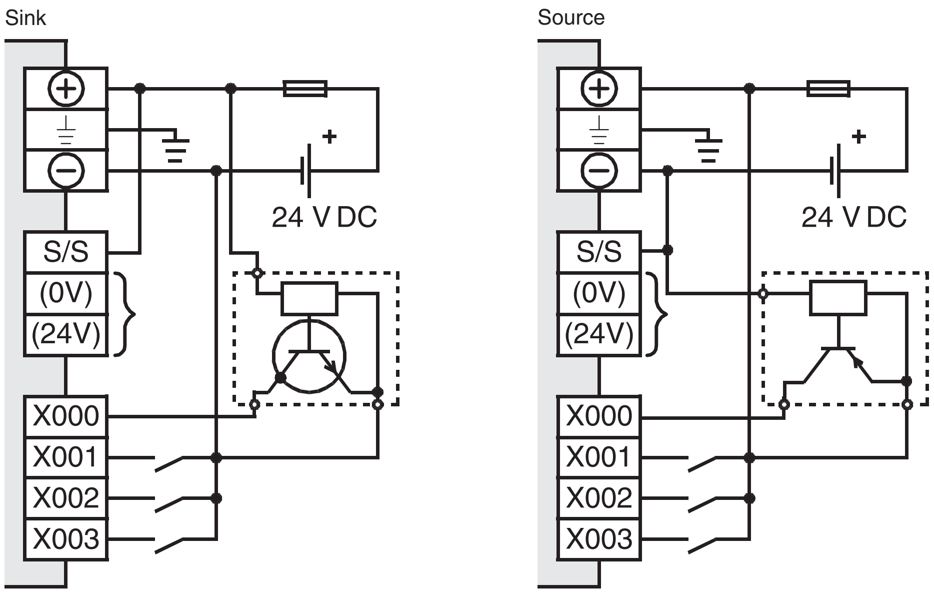
Examples for input types

AC powered base units





DC powered base units


Sink

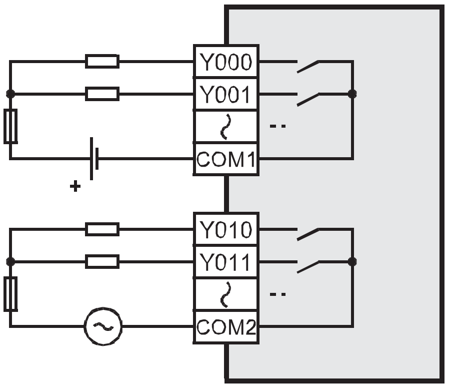
Source

2.7.3 Wiring of Outputs

In case of base units with only few outputs (e. g. FX3G-14M□ or FX3U-16M□) each output can be connected separately. In case of the base units with more outputs, the outputs are pooled into groups of 2, 3, 4, 8 or 16 outputs. Each group has a common contact for the load voltage. These terminals are marked "COM□" for base units with relays outputs or transistor outputs of the sink type and "+V□" for base units with source transistor outputs. "□" stands for the number of the output group e. g. "COM1".

Because the outputs groups are isolated against each other, one base unit can switch several voltages with different potentials. base units with relay outputs can even switch AC and DC voltages.

FX3U base unit with relay outputs



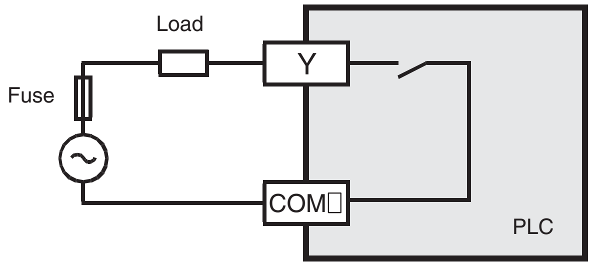
The first group of outputs is used to switch a DC voltage.

The second group of relays controls AC powered loads.

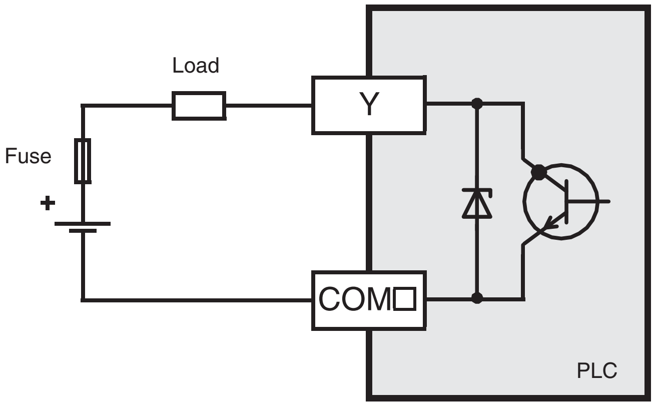
The selection of sink and source output type is done by the selection of a correspondent base unit. Both types are available with DC or AC power supply. The output type is given in the model designation code: base units with the code "MT/□S" provide transistor sink type outputs (e. g. FX3U-16MT/ES) while base units with the code "MT/□SS" provide transistor source type outputs (e. g. FX3U-16MT/ESS).

Examples of output wiring

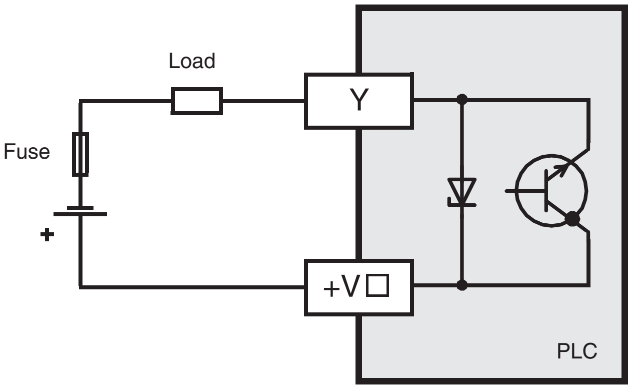
Relay output



Transistor output (sink)



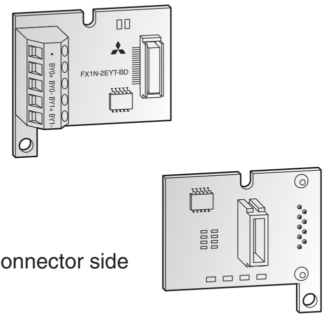
Transistor output (source)



2.8 Extending the Range of Digital Inputs/Outputs

For the MELSEC FX family of PLCs several ways and means are available to provide a base unit with additional inputs and outputs.

2.8.1 Extension Boards



FX$_{1N}$-2EYT-BD with two digital outputs

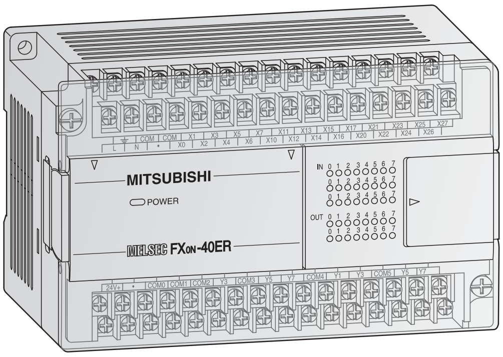
Connector side

For a small number of I/O (2 to 4) an extension adapter board can be installed directly in a FX$_{1S}$ or FX$_{1N}$ base unit. Extension boards therefore do not require any additional installation space.
The state of the additional input and outputs is reflected in special relays in the PLC (see section A.1.5). In the program these relays are used instead of X and Y devices.

Designation	Number of I/O			Output type	Power supply	FX$_{1S}$	FX$_{1N}$	FX$_{2N}$ FX$_{2NC}$	FX$_{3G}$	FX$_{3U}$ FX$_{3UC}$
	Total	No. of inputs	No. of outputs							
FX$_{1N}$-4EX-BD	4	4	—	—	From base unit	●	●	○	○	○
FX$_{1N}$-2EYT-BD	2	—	2	Transistor						

● : The extension board can be used with a base unit of this series.
○ : The extension board cannot be used with this series.

2.8.2 Compact Extension Units



The powered compact input/output extension units have their own power supply. The integrated service power supply (24 V DC) of AC powered extension units can be used for the supply of external devices.
It is possible to choose between relay and transistor (source) output type.

Compact Extension Units of the FX$_{0N}$ Series

Designation	Number of I/O			Output type	Power supply	FX$_{1S}$	FX$_{1N}$	FX$_{2N}$ FX$_{2NC}$	FX$_{3G}$	FX$_{3U}$ FX$_{3UC}$
	Total	No. of inputs	No. of outputs							
FX$_{0N}$-40ER/ES-UL	40	24	16	Relay	100–240 V AC	○	●	○	○	○
FX$_{0N}$-40ER/DS	40	24	16	Relay	24 V DC					
FX$_{0N}$-40ET/DSS	40	24	16	Transistor						

● : The extension unit can be used with a base unit of this series.
○ : The extension unit cannot be used with this series.

Compact Extension Units of the FX2N Series

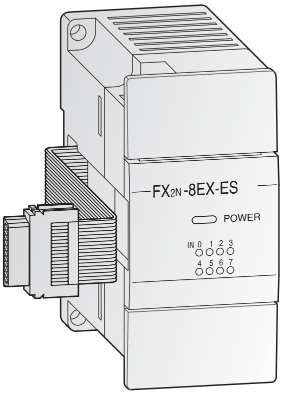
Designation	Number of I/O			Output type	Power supply	FX1S	FX1N	FX2N FX2NC	FX3U	FX3U FX3UC
	Total	No. of inputs	No. of outputs							
FX2N-32ER-ES/UL	32	16	16	Relay	100–240 V AC	○	●	●*	●	●*
FX2N-32ET-ESS/UL	32	16	16	Transistor						
FX2N-48ER-ES/UL	48	16	16	Relay						
FX2N-48ET-ESS/UL	48	24	24	Transistor						
FX2N-48ER-DS	48	24	24	Relay	24 V DC					
FX2N-48ET-DSS	48	24	24	Transistor						

● : The extension unit can be used with a base unit of this series.

○ : The extension unit cannot be used with this series.

* These extension units cannot be connected to a base unit of the FX2NC or FX3UC series.

2.8.3 Modular Extension Blocks



Modular extension blocks have no build-in power supply but very compact dimensions. The FX2N series modular extension blocks are available with 8 or 16 input/output points. It is possible to choose between relay and transistor (source) output type.

The FX2NC series extension blocks are available with 16 or 32 integrated I/O with selectable relay or transistor 16-output models (source type).

Designation	Number of I/O			Output type	Power supply	FX1S	FX1N	FX2N FX2NC	FX3G	FX3U FX3UC
	Total	No. of inputs	No. of outputs							
FX2N-8ER-ES/UL	16①	4	4	Relay	100–240 V AC	○	●	●		●
FX2N-8EX-ES/UL	8	8	—	—						
FX2N-16EX-ES/UL	16	16	—	—						
FX2N-8EYR-ES/UL	8	—	8	Relay						
FX2N-8EYT-ESS/UL	8	—	8	Transistor	24 V DC					
FX2N-16EYR-ES/UL	16	—	16	Relay						
FX2N-16EYT-ESS/UL	16	—	16	Transistor						
FX2NC-16EX-DS	16	16	—	—	From base unit	○	○	●②	○	●②
FX2NC-16EX-T-DS	16	16	—	—						
FX2NC-32EX-DS	32	32	—	—						
FX2NC-16EYT-DSS	16	—	16	Transistor	From base unit					
FX2NC-16EYR-T-DS	16	—	16	Relay						
FX2NC-32EYT-DSS	32	—	32	Transistor						

① The extension block FX2N-8ER-ES/UL occupies 16 input/output points of the PLC. Four inputs and four outputs are occupied but cannot be used.

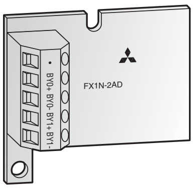
② The FX2NC series extension blocks can only be connected to a base unit of the FX2NC or FX3UC series.

2.9 Extending for Special Functions

A variety of hardware for special functions are available for the MELSEC FX family.

Adapter Boards

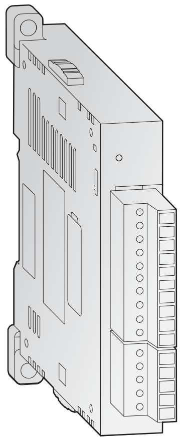
Adapter boards are small circuit boards that are installed directly in the FX1S, FX1N or FX3G controllers, which means that they don't take up any extra space in the switchgear cabinet.



In the case of analog adapter boards, the digital values generated from the signals coming from the analog input adapter's two input channels are written directly to special registers, which makes it particularly easy to process them.
The output value for the analog output adapter is written by the program also to a special register and then converted by the adapter and sent to the output.

Special Adapter

Special adapters can only be connected on the left side of a base unit of the MELSEC FX3G, FX3U andr FX3UC series.

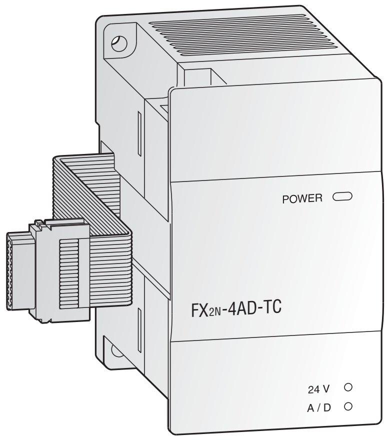


You can install one analog special adapter to a FX3G base unit with 14 or 24 inputs and outputs. Up to two analog special adapter can be mounted to a FX3G base unit with 40 or 60 inputs and outputs. To a FX3U or FX3UC base unit up to four analog special adapters can be connected.

Special adapters do not use any input or output points in the base unit. They communicate directly with the base unit via special relays and registers. Because of this, no instructions for communication with special function modules are needed in the program.

Special function modules

Up to eight special function modules can be connected on the right side of a single base unit of the MELSEC FX family.



In addition to analog modules the available special function modules include communication modules, positioning modules and other types. Each special function module occupies eight input points and eight output points in the base unit.

Communication between the special function module and the PLC base unit is carried out via the memory buffer of the special function module with the help of FROM and TO instructions.

2.9.1 Analog Modules

Without additional modules the base units of the MELSEC FX family can only process digital input and output signals (i.e. ON/OFF data). Additional analog modules are thus required for inputting and outputting analog signals.

Modul Type		Designation	No. of channels	Range	Resolution	FX1S	FX1N	FX2N FX2NC	FX3G	FX3U FX3UC
Analog Input Modules	Adapter Board	FX1N-2AD-BD	2	Voltage: 0 V to 10 V DC	2.5 mV (12 Bit)	●	●	○	○	○
				Current: 4 mA to 20 mA DC	8 µA (11 Bit)					
		FX3G-2AD-BD	2	Voltage: 0 V to 10 V DC	2.5 mV (12 Bit)	○	○	○	●	○
				Current: 4 mA to 20 mA DC	8 µA (11 Bit)					
	Special Adapter	FX3U-4AD-ADP	4	Voltage: 0 V to 10 V DC	2.5 mV (12 Bit)	○	○	○	●	●
				Current: 4 mA to 20 mA DC	10 µA (11 Bit)					
	Special Function Modules	FX2N-2AD	2	Voltage: 0 V to 5 V DC 0 V to 10 V DC	2.5 mV (12 Bit)	○	●	●	●	●
				Current: 4 mA to 20 mA DC	4 µA (12 Bit)					
		FX2N-4AD	4	Voltage: -10 V to 10 V DC	5 mV (with sign, 12 bits)	○	●	●	●	●
				Current: 4 mA to 20 mA DC -20 mA to 20 mA DC	10 µA (with sign, 11 bits)					
		FX2N-8AD[1]	8	Voltage: -10 V to 10 V DC	0.63 mV (with sign, 15 bits)	○	●	●	●	●
				Current: 4 mA to 20 mA DC -20 mA to 20 mA DC	2.50 µA (with sign, 14 bits)					
		FX3U-4AD FX3UC-4AD	4	Voltage: -10 V to 10 V DC	0.32 mV (with sign, 16 bits)	○	○	○	●[2]	●[2]
				Current: 4 mA to 20 mA DC -20 mA to 20 mA DC	1.25 µA (with sign, 15 bits)					
Analog Output Modules	Adapter Board	FX1N-1DA-BD	1	Voltage: 0 V to 10 V DC	2,5 mV (12 Bit)	●	●	○	○	○
				Current: 4 mA to 20 mA DC	8 µA (11 Bit)					
		FX3G-1DA-BD	1	Voltage: 0 V to 10 V DC	2,5 mV (12 Bit)	○	○	○	●	○
				Current: 4 mA to 20 mA DC	8 µA (11 Bit)					
	Special Adapter	FX3U-4DA-ADP	4	Voltage: 0 V to 10 V DC	2,5 mV (12 Bit)	○	○	○	●	●
				Current: 4 mA to 20 mA DC	4 µA (12 Bit)					
	Special Function Modules	FX2N-2DA	2	Voltage: 0 V to 5 V DC 0 V to 10 V DC	2.5 mV (12 Bit)	○	●	●	●	●
				Current: 4 mA to 20 mA DC	4 µA (12 Bit)					

[1] The special function block FX2N-8AD is able to measure voltage, current and temperature.

[2] The FX3UC-4AD can be connected to base units of the FX3UC series only.

Modul Type		Designation	No. of channels	Range	Resolution	FX1S	FX1N	FX2N FX2NC	FX3G	FX3U FX3UC
Analog Output Modules	Special Function Modules	FX2N-4DA	4	Voltage: -10 V to 10 V DC	5 mV (with sign, 12 bits)	○	●	●	●	●
				Current: 0 mA to 20 mA DC 4 mA to 20 mA DC	20 µA (10 Bit)					
		FX3U-4DA	4	Voltage: -10 V to 10 V DC	0.32 mV (with sign, 16 bits)	○	○	○	●	●
				Current: 0 mA to 20 mA DC 4 mA to 20 mA DC	0.63 µA (15 Bit)					
Combined Analog Input & Output Modules	Special Function Modules	FX0N-3A	2 inputs	Voltage: 0 V to 5 V DC 0 V to 10 V DC	40 mV (8 Bit)	○	●	●	○	●[1]
				Current: 4 mA to 20 mA DC	64 µA (8 Bit)					
			1 output	Voltage: 0 V to 5 V DC 0 V to 10 V DC	40 mV (8 Bit)					
				Current: 4 mA to 20 mA DC	64 µA (8 Bit)					
		FX2N-5A	4 inputs	Voltage: -100 mV to 100 mV DC -10 V to 10 V DC	50 µV (with sign, 12 bits) 0.312 mV (with sign, 16 bits)	○	●	●	●	●
				Current: 4 mA to 20 mA DC -20 mA to 20 mA DC	10 µA/1,25 µA (with sign, 15 bits)					
			1 output	Voltage: -10 V to 10 V DC	5 mV (with sign, 12 bits)					
				Current: 0 mA to 20 mA DC	20 µA (10 Bit)					
	Special Adapter	FX3U-3A-ADP	2 inputs	Voltage: 0 V to 10 V DC	2,5 mV (12 Bit)	○	○	○	●	●
				Current: 4 mA to 20 mA DC	5 µA (12 Bit)					
			1 output	Voltage: 0 V to 10 V DC	2.5 mV (12 Bit)					
				Current: 4 mA to 20 mA DC	4 µA (12 Bit)					
Temperature Acquisition Modules	Special Adapter	FX3U-4AD-PT-ADP	4	Pt100 resistance thermometer: -50 °C to 250 °C	0.1 °C	○	○	○	●	●
		FX3U-4AD-PTW-ADP	4	Pt100 resistance thermometer: -100 °C to 600 °C	0.2 °C to 0.3 °C	○	○	○	●	●
		FX3U-4AD-PNK-ADP	4	Pt100 resistance thermometer: -50 °C to 250 °C	0.1 °C	○	○	○	●	●
				Ni1000 resistance thermometer: -40 °C to110 °C	0.1 °C	○	○	○	●	●
		FX3U-4AD-TC-ADP	4	Thermocouple type K: -100 °C to 1000 °C	0.4 °C	○	○	○	●	●
				Thermocouple type J: -100 °C to 600 °C	0.3 °C					

[1] A FX0N-3A can not be connected to base units of the FX3UC series.

Modul Type		Designation	No. of channels	Range	Resolution	FX1S	FX1N	FX2N FX2NC	FX3G	FX3U FX3UC
Temperature Acquisition Modules	Special Function Modules	FX2N-8AD①	8	Thermocouple type K: -100 °C to 1200 °C	0.1 °C	○	●	●	●	●
				Thermocouple type J: -100 °C to 600 °C	0.1 °C					
				Thermocouple type T: -100 °C to 350 °C	0.1 °C					
		FX2N-4AD-PT	4	Pt100 resistance thermometer: -100 °C to 600 °C	0.2 °C to 0.3 °C	○	●	●	●	●
		FX2N-4AD-TC	4	Thermocouple type K: -100 °C to 1200 °C	0.4 °C	○	●	●	●	●
				Thermocouple type J: -100 °C to 600 °C	0.3 °C					
Temperature Control Modules (Special Function Modules)		FX2N-2LC	2	For example with a thermocouple type K: -100 °C to 1300 °C Pt100 resistance thermometer: -200 °C to 600 °C	0.1 °C or 1 °C (depends on temperature probe used)	○	●	●	●	●
		FX3U-4LC	4			○	○	○	●	●

* The special function block FX2N-8AD is able to measure voltage, current and temperature.

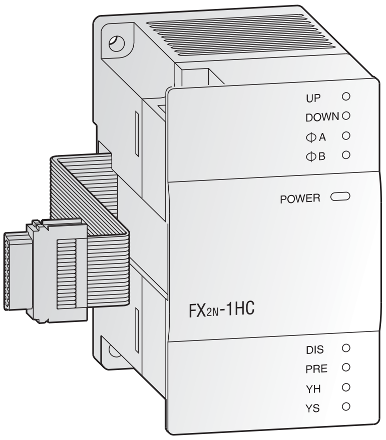
● The adapter board, special adapter or special function module can be used with a base unit or expansion unit of this series.

○ The adapter board, special adapter or special function module cannot be used with this series.

2.9.2 High-Speed Counter Modules and Adapters

FX2N-1HC, FX2NC-1HC and FX3U-2HC

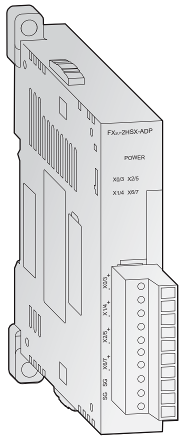
In addition to the internal high-speed MELSEC FX counters, the high-speed counter modules FX2N-1HC, FX2NC-1HC and FX3U-2HC provide the user with an external counter. They count 1- or 2-phase pulses up to a frequency of 50 kHz resp. 200 kHz for the FX3U-2HC. The counting range covers either 16 or 32 bit.

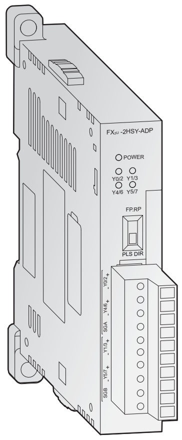


The two integrated transistor outputs can be switched independently of one another by means of internal comparison functions. Hence, simple positioning tasks can also be realized economically. In addition, the high-speed counter modules can be used as ring counters.

FX3U-4HSX-ADP and FX3U-2HSY-ADP

These adapter modules allow direct processing of positioning application data.





The FX3U-4HSX-ADP (far left) provides four high speed counter inputs up to 200 kHz while the FX3U-2HSY-ADP (left) delivers two channels of pulse train outputs up to 200 kHz.

CAUTION:
When connecting these special adapters, same input resp. output numbers are allocated to the base unit and the special adapter. Wire either one of the input or output terminals

Overview of High-Speed Counter Modules/Adapters

Module type	Designation	Description	FX1S	FX1N	FX2N	FX2NC	FX3G	FX3U	FX3UC
Special function module	FX2N-1HC	1-ch high speed counter	○	○	●	●	○	●	●
	FX2NC-1HC		○	○	○	●	○	○	●
	FX3U-2HC	2-ch high speed counter	○	○	○		○	●	●
Special adapter	FX3U-4HSX-ADP	Differential line driver input (high-speed counter)	○	○	○	○	○	●	○
	FX3U-2HSY-ADP	Differential line driver input (positioning output)							

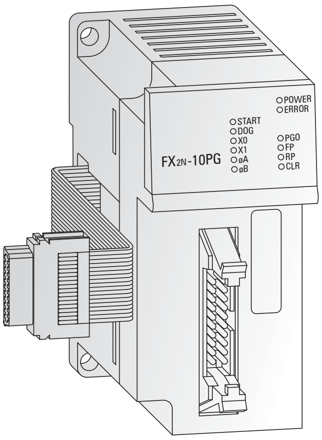
● The special adapter or special function module can be used with a base unit or expansion unit of this series.

○ The special adapter or special function module cannot be used with this series.

2.9.3 Positioning Modules

FX2N-1PG-E, FX2N-10PG

The positioning modules FX2N-1PG-E and FX2N-10PG are extremely efficient single-axis positioning modules for controlling either step drives or servo drives (by external regulator) with a pulse chain.



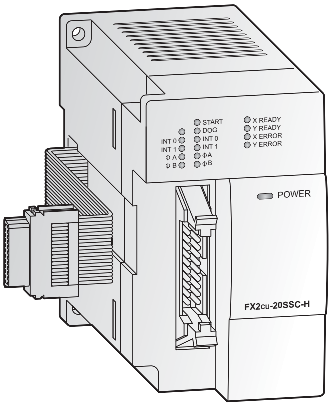
They are very suitable for achieving accurate positioning in combination with the MELSEC FX series. The configuration and allocation of the position data are carried out directly via the PLC program.

The FX2N-1PG-E provides an 100 kHz open collector output while the FX2N-10PG is equipped with a 1 MHz differential line driver output.

A very wide range of manual and automatic functions are available to the user.

FX3U-20SSC-H

The SSCNET* module FX3U-20SSC-H can be used in combination with a FX3U or FX3UC programmable controller to achieve a cost effective solution for high precision, high speed positioning. The plug-and-play fiber optic SSCNET cabling reduces setup time and increases control distance for positioning operations in a wide range of applications.



Servo parameters and positioning information for the FX3U-20SSC-H are easily set up with an FX3U or FX3UC base unit and a personal computer. For parameter setting, monitoring and testing the easy programming software FX Configurator-FP is available.

* SSCNET: **S**ervo **S**ystem **C**ontroller **Net**work

Overview of Positioning Modules

Module type	Designation	Description	FX1S	FX1N	FX2N FX2NC	FX3G	FX3U FX3UC
Special function modules	FX2N-1PG-E	Pulse output for independent 1-axis control	○	○	●	○	●
	FX2N-10PG						
	FX3U-20SSC-H	Simultaneous 2-axis (independent 2-axis) control (Applicable to SSCNET III)	○	○	○	○	●

● The special function module can be used with a base unit or expansion unit of this series.
○ The special function module cannot be used with this series.

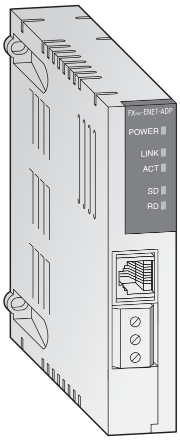
2.9.4 Network Modules for ETHERNET

ETHERNET is the most widespread network for connection of information processors such as personal computers and work stations. By loading an ETHERNET interface into the PLC, production-related management information can be transmitted rapidly to personal computers or work stations. ETHERNET is a platform for a very wide range of data communications protocols. The combination of ETHERNET and the extremely widespread TCP/IP protocol enables high-speed data communications between process supervision systems and the MELSEC PLC series. TCP/IP provides logical point-to-point links between two ETHERNET stations.

The programming software GX IEC Developer provides function blocks or setup routines for the PLCs, making the configuration of one or more TCP/IP links a quick and easy process.

FX2NC-ENET-ADP

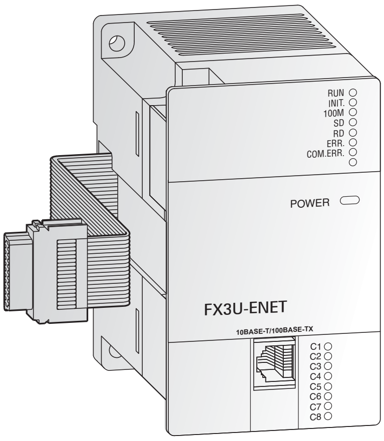
The FX2NC-ENET-ADP communications adapter is an Ethernet interface with 10BASE-T specifications for the FX1S, FX1N, FX2NC and FX2N series*.



PCs with GX IEC Developer or MX Component and the virtual COM port driver installed are enabled for program upload/download through this module.

* When connecting this special adapter to a FX1S or FX1N PLC the communications adapter FX1N-CNV-BD is required. When connecting this adapter to a FX2N PLC the communications adapter FX2N-CNV-BD is required.

FX3U-ENET



The FX3U-ENET communications module provides the FX3G, FX3U or FX3UC with a direct connection to an Ethernet network.

The FX3U-ENET enables 8 ports of simultaneous Ethernet communication with features such as peer-to-peer communication, extensive e-mail send/receive options, and program upload/download. The FX3U-ENET is also used to communicate with GOTs via the Ethernet. Easy communication parameter setup and module troubleshooting is also possible using the dedicated software, FX Configurator-EN.

Overview of Network Modules for ETHERNET

Module type	Designation	Description	FX1S	FX1N	FX2N FX2NC	FX3G	FX3U FX3UC
Special function modules	FX2NC-ENET-ADP	ETHERNET network modules	●	●	●	○	○
	FX3U-ENET		○	○	○	●	●

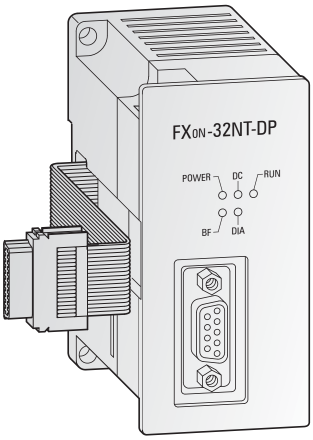
2.9.5 Network Modules for Profibus/DP

The Profibus/DP network enables communication between a master module and decentralised slave modules, with data transfer rates of up to 12 Mbps. With a MELSEC PLC as master, PROFIBUS/DP allows quick and simple connection of sensors and actuators, even from different manufacturers.

A MELSEC PLC, serving as slave in a PROFIBUS/DP network, can execute decentralised control tasks and simultaneously exchange data with the PROFIBUS/DP master.

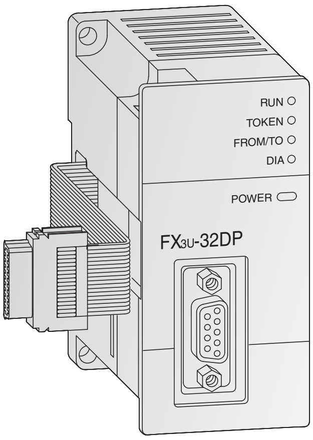
To help reduce costs PROFIBUS/DP uses RS485 technology with shielded 2-wire cabling.

FX0N-32NT-DP



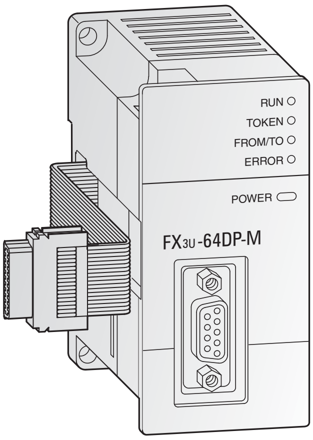
The FX0N-32NT-DP PROFIBUS DP slave module enables the attached FX base unit to be a slave station on a PROFIBUS DP network. Transfer of up to 40 bytes of data per cycle is supported at up to 12 Mbps.

FX3U-32DP



Like the FX0N-32NT-DP, the FX3U-32DP is a PROFIBUS/DP slave module. It allows the integration of a FX3G, FX3U, or FX3UC PLC into a PROFIBUS/DP network.

FX3U-64DP-M

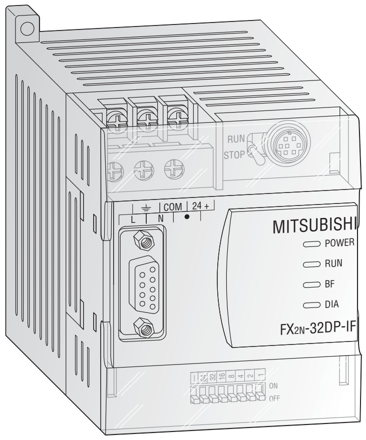


The FX3U-64DP-M PROFIBUS DP master module is available for the FX3U and FX3UC base units and enables the attached FX base unit to be a master station on a PROFIBUS DP-V1 network. PROFIBUS DP allows for the implementation of decentralized control with comprehensive data and alarm processing capabilities.

Easy setup is available by using the GX Configurator-DP software package.

FX2N-32DP-IF

The remote I/O station FX2N-32DP-IF forms an extremely compact communication unit and provides a connection of I/O modules with up to 256 I/O points and/or up to 8 special function modules as an alternative.



It is not necessary to install an FX base unit to a remote I/O station. The FX2N-32DP-IF connects the connected I/O modules or special function modules to the master station of a PROFIBUS/DP network.

Used in combination with a FX3U or FX3UC base unit and a FX3U-64DP-M master station, the creation of a high performance remote I/O system, consisting entirely of MELSEC FX modules, is possible.

PROFIBUS data such as the baud rate or I/O data can be monitored directly with the programming software or on the hand-held programming unit FX-20P-E. This facilitates an easy error diagnosis directly on the remote I/O station.

Overview of Profibus/DP modules

Module type	Designation	Description		FX1S	FX1N	FX2N FX2NC	FX3G	FX3U FX3UC
Special function modules	FX0N-32NT-DP	PROFIBUS/DP slave		●	●	●	○	●
	FX3U-32DP			○	○	○	●	●
	FX3U-64DP-M	PROFIBUS/DP master		○	○	○	○	●
—	FX2N-32DP-IF	PROFIBUS/DP remote I/O station	Power supply: 100–240 V AC	Compatible with PROFIBUS/DP masters				
	FX2N-32DP-IF-D		Power supply: 24 V DC					

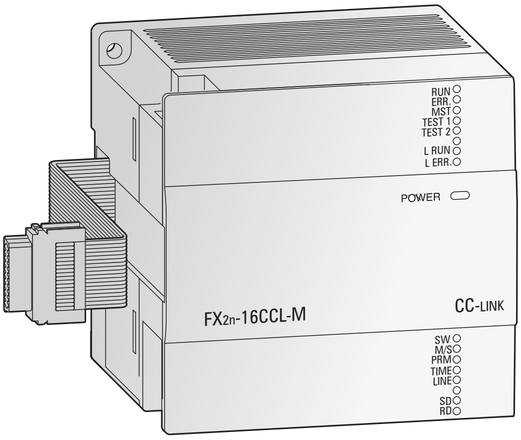
● The special function module can be used with a base unit or expansion unit of this series.
○ The special function module cannot be used with this series.

2.9.6 Network Modules for CC-Link

CC-Link Master Module FX2N-16CCL-M

The CC-Link network enables the controlling and monitoring of decentralized I/O modules at the machine.

The CC-Link master module FX2N-16CCL-M is a special extension block which assigns an FX series PLC as the master station of the CC-Link system.



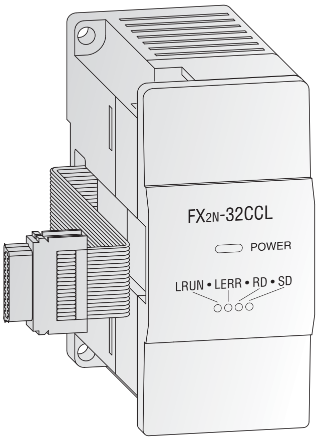
The setting of all modules within the network is handled directly via the master module.

Up to 15 remote stations (7 remote I/O stations and up to 8 remote device stations) can be connected to the master station. Two master modules can be connected to one base unit.

The maximum communications distance is 1200 m without repeater.

CC-Link Communication Modules FX2N-32CCL and FX3U-64CCL

The communication modules FX2N-32CCL and FX3U-64CCL enable the user to connect to the CC-Link network with a superior PLC system as master CPU. This gives him access to the network of all MELSEC PLC systems and frequency inverters and to additional products from other suppliers.



Thus the network is expandable via the digital inputs/outputs of the FX modules to a maximum of 256 I/Os.

Overview of Network Modules for CC-Link

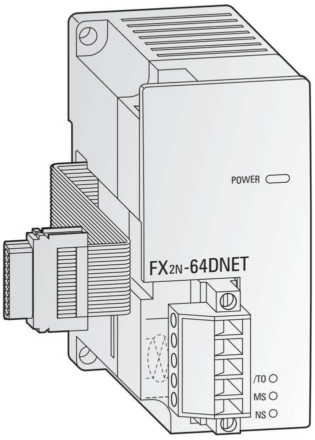
Module type	Designation	Description	FX1S	FX1N	FX2N FX2NC	FX3G	FX3U FX3UC
Special function modules	FX2N-16CCL-M	Master for CC-Link	○	●	●	●	●
	FX2N-32CCL	Remote device station for CC-Link	○	●	●	●	●
	FX3U-64CCL		○	○	○	●	●

● The special function module can be used with a base unit or expansion unit of this series.

○ The special function module cannot be used with this series.

2.9.7 Network Module for DeviceNet

DeviceNet represents a cost-effective solution for the network integration of low-level terminal equipment. Up to 64 devices including a master can be integrated in one network. For the data exchange a cable with two shielded twisted-pair cables is used.



The DeviceNet slave module FX2N-64DNET can be used to connect FX2N, FX2NC and FX3U programmable controllers to a DeviceNet network.

The FX2N-64DNET can communicate to the master by the master/slave communication (using the master/slave I/O connection), and to other nodes supporting the UCMM connection by client/server communication.

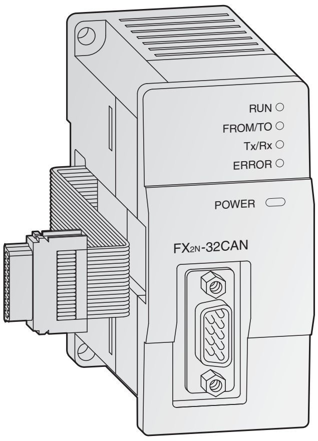
The communication between the programmable controller and the internal buffer memory of the FX2N-64DNET is handled by FROM/TO instructions.

Module type	Designation	Description	FX1S	FX1N	FX2N FX2NC	FX3G	FX3U	FX3UC
Special function module	FX2N-64DNET	DeviceNet slave module	○	○	●	○	●	○

● The special function module can be used with a base unit or expansion unit of this series.
○ The special function module cannot be used with this series.

2.9.8 Network Module for CANopen

CANopen is an “open” implementation of the Controller Area Network (CAN), which is defined in the EN50325-4 standard. CANopen offers cost effective network communications with fault-resistant network structure where components of different manufacturers can be integrated quickly and easily. CANopen networks are used for connecting sensors, actuators and controllers in a variety of applications. The bus uses inexpensive twisted-pair cabling.



The FX2N-32CAN communications module makes it possible to connect an FX2N, FX3G, FX3U or FX3UC PLC to an existing CANopen network.

In addition to real-time capabilities and high-speed data transfer at rates of up to 1 Mbps the CANopen module also shines with high transfer reliability and simple network configuration. Up to 120 words of data can be sent and received as process data objects (30 PDOs).

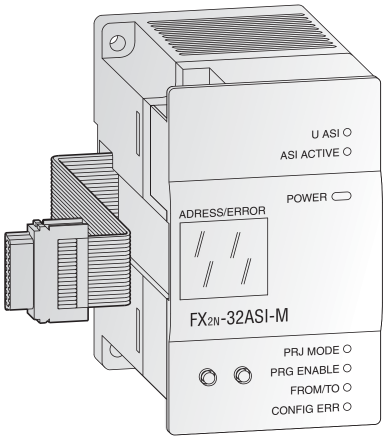
Communication with the module’s memory buffer is performed with simple FROM/TO instructions

Module type	Designation	Description	FX1S	FX1N	FX2N	FX2NC	FX3G	FX3U FX3UC
Special function module	FX2N-32CAN	CANopen module	○	○	●	○	●	●

● The special function module can be used with a base unit or expansion unit of this series.
○ The special function module cannot be used with this series.

2.9.9 Network Module for AS-Interface

The Actuator Sensor interface (AS interface or ASi) is an international standard for the lowest field bus level. The network suits versatile demands, is very flexible and particularly easy to install. The ASi is suitable for controlling sensors, actuators and I/O units.



The FX2N-32ASI-M serves as master module for the connection of the FX1N/FX2N and FX3U/FX3UC PLC to the AS-Interface system. Up to 31 slave units with up to 4 inputs and 4 outputs can be controlled.

For status and diagnosis messages a 7-segment display is integrated.

Module type	Designation	Description	FX1S	FX1N	FX2N	FX2NC	FX3G	FX3U
Special function module	FX2N-32ASI-M	Master for AS-i system	○	●	●	○	○	●

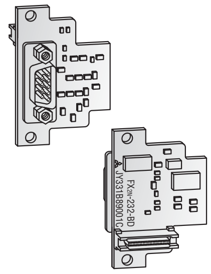
● The special function module can be used with a base unit or expansion unit of this series.

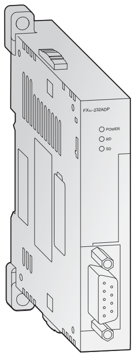
○ The special function module cannot be used with this series.

2.9.10 Interface Modules and Adapters

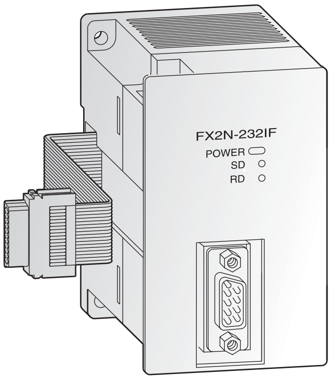
For serial data communication a large range of interface modules/adapters is available. Shown below are only some examples, but the following table covers all available interfaces.

RS232C interface adapter board FX2N-232-BD





Communication special adapter FX3U-232ADP (RS232C interface)



Interface Module FX2N-232IF

The interface module FX2N-232IF provides an RS232C interface for serial data communications with the MELSEC FX2N, FX2NC, FX3U and FX3UC.

Communication with PCs, printers, modems, barcode readers etc. is handled by the PLC program. The send and receive data are stored in the FX2N-232IF's own buffer memory.

Overview of Interface Modules and Adapters

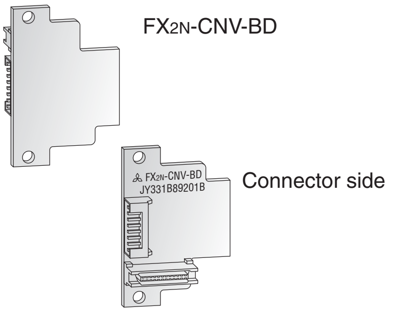
Module type	Designation	Description	FX1S	FX1N	FX2N FX2NC	FX3G	FX3U	FX3UC
Adapter boards	FX1N-232-BD	RS232C interfaces	●	●	○	○	○	○
	FX2N-232-BD		○	○	●	○	○	○
	FX3G-232-BD		○	○	○	●	○	○
	FX3U-232-BD		○	○	○	○	●	○
Special adapter	FX2NC-232ADP*		●	●	●	○	○	○
	FX3U-232ADP-MB		○	○	○	●	●	●
Special function module	FX2N-232IF		●	●	●	○	●	●
Adapter boards	FX1N-422-BD	RS422 interfaces	●	●	○	○	○	○
	FX2N-422-BD		○	○	●	○	○	○
	FX3G-422-BD		○	○	○	●	○	○
	FX3U-422-BD		○	○	○	○	●	○
Adapter boards	FX1N-485-BD	RS485 interfaces	●	●	○	○	○	○
	FX2N-485-BD		○	○	●	○	○	○
	FX3G-485-BD		○	○	○	●	○	○
	FX3U-485-BD		○	○	○	○	●	○
Special adapter	FX2NC-485ADP*		●	●	●	○	○	○
	FX3U-485ADP-MB		○	○	○	●	●	●
Adapter board	FX3U-USB-BD	USB interface	○	○	○	○	●	○

* The FX2NC-232ADP and the FX2NC-485ADP require a FX2N-CNV-BD or FX1N-CNV-BD interface adapter when connecting to a FX1S, FX1N or FX2N base unit.

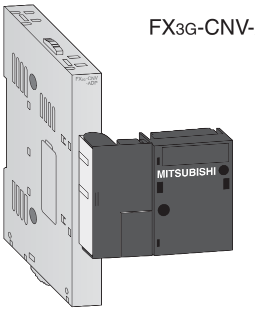
2.9.11 Communication Adapters

Communication adapters boards

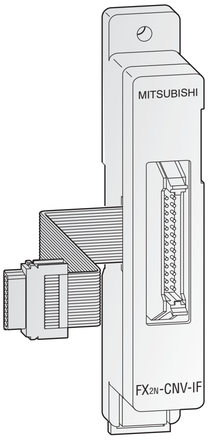
Communication adapters boards (product code FX□□-CNV-□□) are are installed directly in a base unit. They are needed to connect special adapters (FX□□-□□□ADP) to the left-hand side of base units of the FX1N, FX2N, FX3G or FX3U series.


FX2N-CNV-BD

Connector side


FX3G-CNV-ADP

FX2N-CNV-IF



The FX2N-CNV-IF interface allows special function modules of the old FX series to be connected to the base units of the FX family.

Overview of Communication Adapters

Module type	Designation	Description	FX1S	FX1N	FX2N	FX2NC	FX3G	FX3U	FX3UC
Adapter boards	FX1N-CNV-BD	Communication adapters for connection of special adapters	●	●	○	○	○	○	○
	FX2N-CNV-BD		○	○	●	○	○	○	○
	FX2NC-CNV-IF		○	○	○	●	○	○	●
	FX3G-CNV-ADP		○	○	○	○	●	○	○
	FX3U-CNV-BD		○	○	○	○	○	●	○
Adapter	FX2N-CNV-IF	Communication adapter for connection of FX series modules	●	●	●	○	○	●	○

● The adapter can be used with a base unit of this series.

○ The adapter cannot be used with this series.

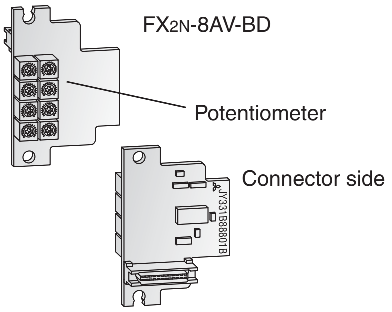
2.9.12 Setpoint Adapter Boards

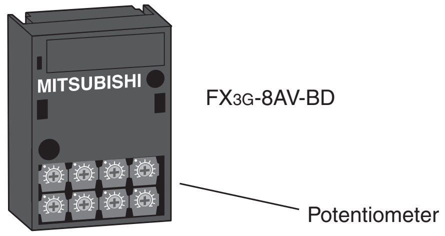
These analog setpoint adapters enable the user to set 8 analog setpoint values. The analog values (0 to 255) of the potentiometers are read into the controller and used as default setpoint values for timers, counters and data registers by the user's PLC programs.

Each potentiometer value can also be read as an 11 position rotary switch (positions 0 to 10).

Setpoint value polling is performed in the PLC program using the dedicated instruction VRRD. The position of an rotary switch is read using the VRSC instruction.

The analog setpoint adapters are installed in the expansion slot of the base unit. No additional power supply is required for operation.





Module type	Designation	Description	FX1S	FX1N	FX2N	FX2NC	FX3G	FX3U FX3UC
Adapter boards	FX1N-8AV-BD	Analog setpoint adapters	●	●	○	○	○	○
	FX2N-8AV-BD		○	○	●	○	○	○
	FX3G-8AV-BD		○	○	○	○	●	○

● The adapter board can be used with a base unit or expansion unit of this series.
○ The adapter board cannot be used with this series.

2.10 System Configuration

A basic FX PLC system can consist of a stand alone base unit, with the functionality and I/O range increased by adding extension I/O and special function modules. An overview of available options is given in sections 2.8 and 2.9.

Base Units

Base units are available with different I/O configurations from 10 to 128 points but can be expanded to 384 points depending upon the FX range selected.

Extension Boards

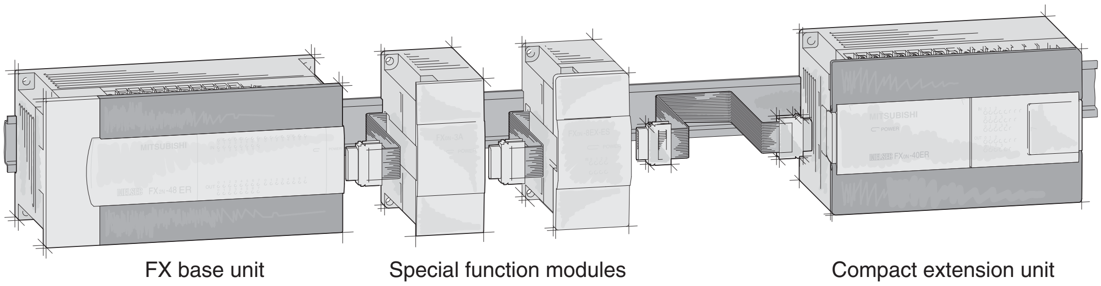
Extension adapter boards can be installed directly into the base unit and therefore do not require any additional installation space. For a small number of I/O (2 to 4) an extension adapter boards can be installed directly into the FX1S or FX1N controller. Interface adapter boards can also provide the FX PLC with additional RS232 or RS485 interfaces.

Extension I/O Modules

With the exception of the FX1S series, unpowered modular extension blocks and powered compact extension units modules can be added to all base units of the FX family. For modular extension blocks powered by the base unit, the power consumption has to be calculated as the 5 V DC bus can only support a limited number of expansion I/O.

Special Function Modules / Special Adapters

A wide variety of special function modules are available for all FX PLCs, again the exception is the FX1S. They cover networking functionality, analog control, pulse train outputs and temperature inputs (for further details please refer to section 2.9).



Expansion Options

PLC	Number of modules on the left side of base unit	Number of boards in expansion board port of base unit	Number of modules on the right side of base unit
FX$_{1S}$	The modules FX$_{0N}$-485ADP and FX$_{0N}$-232ADP can be mounted in combination with a communication adapter FX$_{1N}$-CNV-BD.	1 (product code FX□□-□□□-BD)	—
FX$_{1N}$			Up to 2 special function modules of the FX$_{2N}$ series.
FX$_{2N}$			Up to 8 special function modules of the FX$_{2N}$ series.
FX$_{2NC}$	The modules FX$_{0N}$-485ADP and FX$_{0N}$-232ADP can be mounted on the left side directly. An adapter is not required.	—	Up to 4 special function modules of the FX$_{2N}$ series.
FX$_{3G}$	Up to 4 special adapters of the FX$_{3U}$ series can be mounted on the left side of the base unit in combination with an adapter board FX$_{3G}$-CNV-BD.	Up to 2 (dependent on the type of base unit) (product code FX$_{3G}$-□□□-BD)	Up to 8 special function modules of the FX$_{2N}$ or FX$_{3U}$ series.
FX$_{3U}$	Up to 10 special adapters of the FX$_{3U}$ series can be mounted on the left side of the base unit directly or in combination with an interface/communication adapter FX$_{3U}$-□□□-BD.	1 (product code FX$_{3U}$-□□□-BD)	
FX$_{3UC}$	Up to 6 special adapters of the FX$_{3U}$ series can be directly mounted on the left side of the base unit.	—	

The difference between a base unit, extension unit and extension block is described as follows:

- A base unit is made up of 4 components i.e. power supply (for base units with AC power supply only), inputs, outputs and CPU.
- An extension unit is made up of 3 components i.e. power supply, inputs and outputs.
- An extension block is made up of 1or 2 components i.e. inputs and/or outputs.

It can be seen that the extension block does not have a power supply. It therefore obtains its power requirement from either the base unit or extension unit.

Hence it is necessary to determine how many of these unpowered units can be connected before the "On Board" power supply capacity is exceeded.

2.10.1 Connection of Special Adapters

Special adapters of the FX3U series can be mounted on the left side of the base unit of the FX3G, FX3U, and FX3UC series.

NOTE

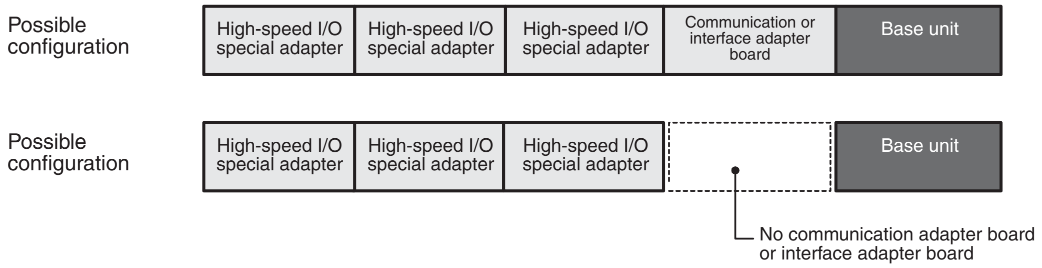
The following rules apply to FX3U base units. For the rules of system configuration for the FX3G or FX3UC series, please refer to the appropriate manual.

High-speed input/output special adapters

Up to two high-speed input special adapters FX3U-4HSX-ADP and up to two high-speed output special adapters FX3U-2HSY-ADP can be connected to a base unit.

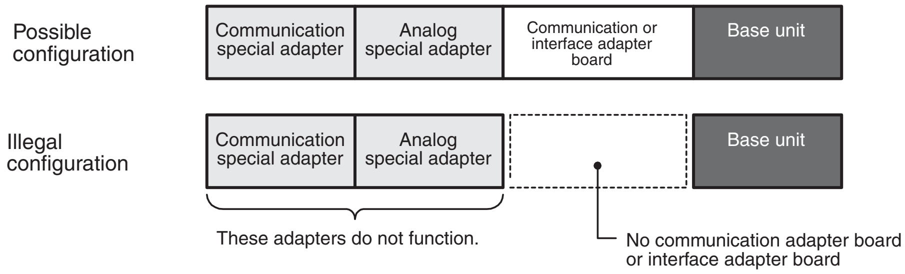
Connect all high-speed I/O special adapters before connecting other special adapters when they are used in combination. A high-speed I/O special adapter can not be mounted on the left side of a communication or analog special adapter.

When only high-speed input/output special adapters are connected, the adapters can be used without a communication or interface adapter board installed in the base unit.


Possible configuration
High-speed I/O special adapter
High-speed I/O special adapter
High-speed I/O special adapter
Communication or interface adapter board
Base unit
Possible configuration
High-speed I/O special adapter
High-speed I/O special adapter
High-speed I/O special adapter
Base unit
No communication adapter board or interface adapter board

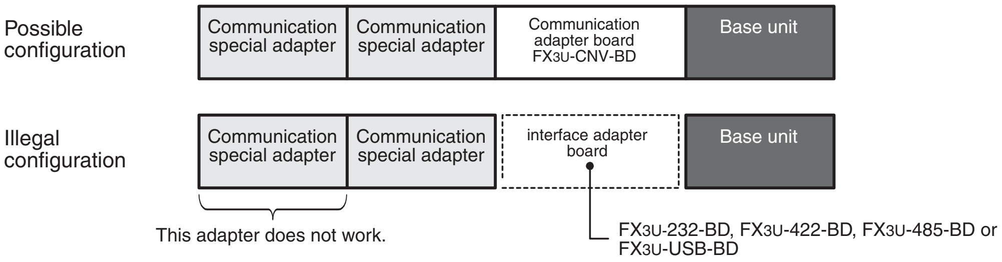
Combination of analog and communication special adapters

Analog and communication special adapters must be used with a communication adapter board or an interface adapter board installed in the base unit.


Possible configuration
Communication special adapter
Analog special adapter
Communication or interface adapter board
Base unit
Illegal configuration
Communication special adapter
Analog special adapter
Base unit
These adapters do not function.
No communication adapter board or interface adapter board

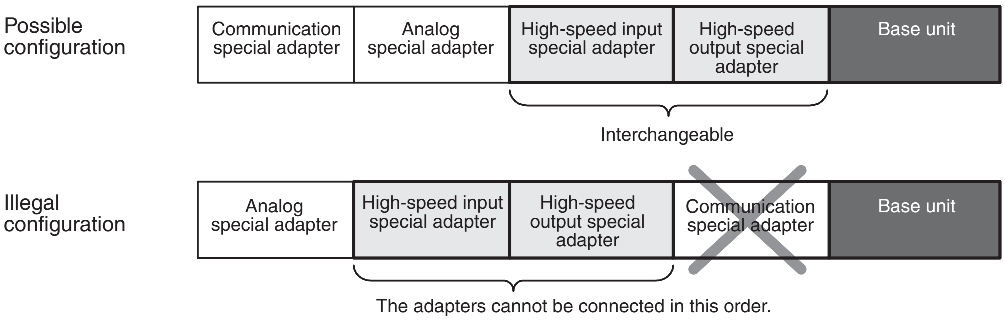
Combination of communication special adapters and an interface adapter board

When instead of a communication adapter board FX3U-CNV-BD an interface adapter board FX3U-232-BD, FX3U-422-BD, FX3U-485-BD, or FX3U-USB-BD is mounted, one communication special adapter FX3U-232ADP or FX3U-485ADP may be used.


Possible configuration
Communication special adapter
Communication special adapter
Communication adapter board FX3U-CNV-BD
Base unit
Illegal configuration
Communication special adapter
Communication special adapter
interface adapter board
Base unit
This adapter does not work.
FX3U-232-BD, FX3U-422-BD, FX3U-485-BD or FX3U-USB-BD

Combination of high-speed input/output, analog and communication special adapters

When these adapters are used, connect the high-speed input/output special adapters on the left side of the base unit. The high-speed input/output special adapters cannot be connected on the downstream side of any communication/analog special adapter.


Possible configuration
Communication special adapter
Analog special adapter
High-speed input special adapter
High-speed output special adapter
Base unit
Interchangeable
Illegal configuration
Analog special adapter
High-speed input special adapter
High-speed output special adapter
Communication special adapter
Base unit
The adapters cannot be connected in this order.

Summary

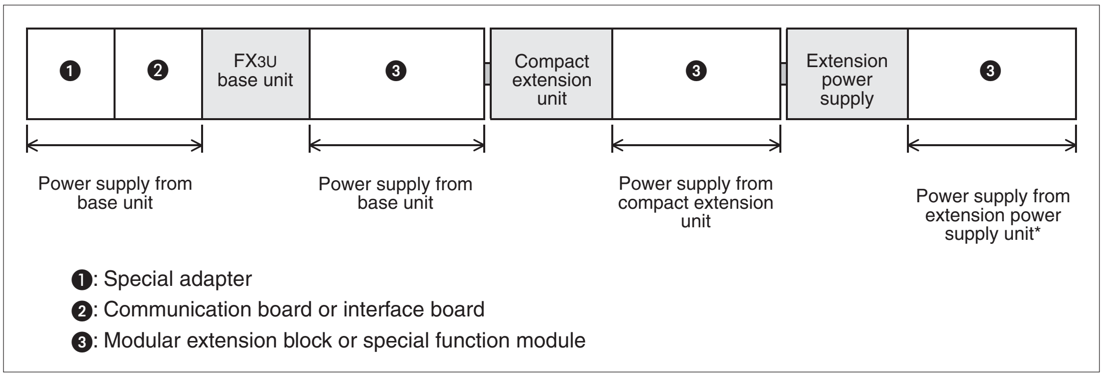
Mounted communication adapter board or interface adapter board	Number of connectable special adapters			
	Communication special adapter	Analog special adapter	High-speed input special adapter	High-speed output special adapter
No adapter board installed	These special adapters cannot connected.		2	2
FX3U-CNV-BD	2	4	2	2
FX3U-232-BD FX3U-422-BD FX3U-485-BD FX3U-USB-BD	1	4	2	2

2.10.2 Basic Rules for System Configuration

The following considerations should be taken into account when configuring a system with extension units or special function modules:

- Current consumption from 5 V DC backplane bus
- 24 V DC current consumption
- The total number of inputs and outputs point must be smaller than the number of max. I/Os.

The following figure shows the distribution of the power supply in case of an FX3U.


❶
❷
FX3U base unit
❸
Compact extension unit
❸
Extension power supply
❸
Power supply from base unit
Power supply from base unit
Power supply from compact extension unit
Power supply from extension power supply unit*

❶: Special adapter
❷: Communication board or interface board
❸: Modular extension block or special function module

* When connecting an **input** extension block on the downstream side of an extension power supply unit, this input extension block is supplied from the base unit or from an input/output powered extension unit which is mounted between base unit and extension power supply unit.

Calculation of current consumption

The power is supplied to each connected device from the built-in power supply of the base unit, the input/output powered extension unit or – for FX3U and FX3UC only– the extension power supply unit.

There are three types of built-in power supplies

- 5V DC
- 24V DC (for internal use)
- 24V DC service power supply (only in AC powered base units).

The following table shows the capacities of the built-in power supplies:

Model		5 V DC built-in power supply	24 V DC built-in power supply (internal / service power supply
Base units	FX1N	Suitable to power all connected modules	400 mA
	FX2N	290 mA	250 mA (FX2N-16M□, FX2N-32M□) 460 mA (all other base units)
	FX3G	Sufficient for 2 special function modules or 32 additional I/O	—
	FX3U	500 mA	400 mA (FX3U-16M□, FX3U-32M□) 600 mA (all other base units)
	FX3UC	400 / 480 / 560/ 600 mA	—
Compact extension unit	FX2N	690 mA	250 mA (FX2N-32E□) 460 mA (FX2N-48E□)

When only input/output extension blocks are added, a quick reference matrix can be used.

When also special function modules are added, calculate the current consumption to ensure that the total current to be consumed by the additional modules can be supplied by the built-in power supply. For details of the power consumption please refer to the appendix (section A.4).

2.10.3 Quick Reference Matrixes

When only input/output extension blocks without a built-in power supply are added to a base unit, a quick reference matrix can be used. The following examples are valid for base units of the FX3U series.

AC powered base units

In the following quick reference matrixes, the value at the intersection of the number of input points to be added (horizontal axis) with the number of output points to be added (vertical axis) indicates the remaining power supply capacity.

For FX3U-16MR/ES, FX3U-16MT/ES, FX3U-16MT/ESS, FX3U-32MR/ES, FX3U-32MT/ES or FX3U-32MT/ESS:

see example

Not allowed to add

Number of additional outputs

40	25								
32	100	50	0						
24	175	125	75	25					
16	250	200	150	100	50	0			
8	325	275	225	175	125	75	25		
0	400	350	300	250	200	150	100	50	0
	0	8	16	24	32	40	48	56	64

Number of additional inputs

- Example

  When a 16-input and a 16-output point extension block are connected to a base unit FX3U-16M□ or FX3U-32M□, the residual current of the 24V DC service power supply is 150 mA.

For FX3U-48MR/ES, FX3U-48MT/ES, FX3U-48MT/ESS, FX3U-64MR/ES, FX3U-64MT/ES, FX3U-64MT/ESS, FX3U-80MR/ES, FX3U-80MT/ES, FX3U-80MT/ESS, FX3U-128MR/ES, FX3U-128MT/ES or FX3U-128MT/ESS:

see example

Number of additional outputs

64	0												
56	75	25											
48	150	100	50	0									
40	225	175	125	75	25								
32	300	250	200	150	100	50	0						
24	375	325	275	225	175	125	75	25					
16	450	400	350	300	250	200	150	100	50	0			
8	525	475	425	375	325	275	225	175	125	75	25		
0	600	550	500	450	400	350	300	250	200	150	100	50	0
	0	8	16	24	32	40	48	56	64	72	80	88	96

Number of additional inputs

- Example

  When a 32-input and a 16-output point extension block are connected to an AC powered base unit with 48, 64, 80 or 128 I/Os, the 24 V DC service power supply can still deliver a maximum current of 250 mA to other devices.

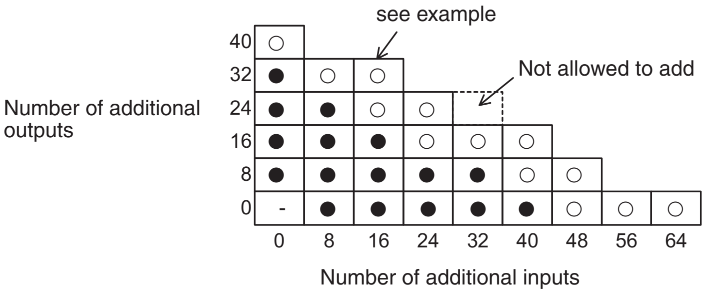
Confirm the current capacity of 24 V DC service power supply from the value shown in the quick reference matrix. This remaining power supply capacity (current) can be used as a power supply to external loads (sensors or the like) by the user. When special function modules are connected, it is necessary to consider whether they can be powered by the remaining power supply capacity.

DC powered base units

The DC power type base units have restrictions in expandable I/O points since they lack a built-in service power supply.

The following matrixes show the expandable units up to the ○ mark, where the desired inputs (horizontal axis) and outputs (vertical axis) intersect. System are expandable up to the ● mark when the supply voltage is 16.8 V to 19.2 V.

For FX3U-16MR/DS, FX3U-16MT/DS, FX3U-16MT/DSS, FX3U-32MR/DS, FX3U-32MT/DS or FX3U-32MT/DSS:



Number of additional outputs									
40	○								
32	●	○	○						
24	●	●	○	○					
16	●	●	●	○	○	○			
8	●	●	●	●	●	○	○		
0	-	●	●	●	●	●	○	○	○
	0	8	16	24	32	40	48	56	64

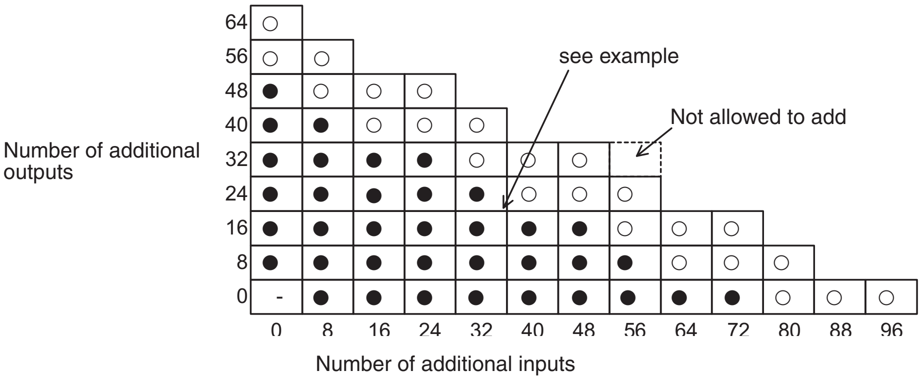
Number of additional inputs

see example
Not allowed to add

- Example

  When adding 16 inputs to a DC powered base unit with 16 or 32 I/O, a maximum of 32 outputs are expandable. When adding 16 inputs under the supply voltage 16.8 V to 19.2 V, a maximum of 16 outputs are expandable.

For FX3U-48MR/DS, FX3U-48MT/DS, FX3U-48MT/DSS, FX3U-64MR/DS, FX3U-64MT/DS, FX3U-64MT/DSS, FX3U-80MR/DS, FX3U-80MT/DS or FX3U-80MT/DSS:



Number of additional outputs													
64	○												
56	○	○											
48	●	○	○	○									
40	●	●	○	○	○								
32	●	●	●	●	○	○	○						
24	●	●	●	●	●	○	○	○					
16	●	●	●	●	●	●	●	○	○	○			
8	●	●	●	●	●	●	●	●	○	○	○		
0	-	●	●	●	●	●	●	●	●	●	○	○	○
	0	8	16	24	32	40	48	56	64	72	80	88	96

Number of additional inputs

see example
Not allowed to add

- Example

  When adding 32 inputs to a DC powered base unit with 48, 64, or 80 I/Os, a maximum of 40 outputs are expandable. But when adding 32 inputs under the supply voltage 16.8 V to 19.2 V, a maximum of 24 outputs are expandable.

2.11 I/O Assignment

The assignment of the inputs and outputs in a PLC of the MELSEC FX family is fixed and can not be altered.

When power is turned on after input/output powered extension units/blocks have been connected, the base unit automatically assigns the input/output numbers (X/Y) to the units/blocks.

Therefore, it is unnecessary to specify the input/output numbers with parameters.

Input/output numbers are not assigned to special function units/blocks.

2.11.1 Concept of assigning

Input/output numbers (X/Y) are octal

The inputs and outputs of a PLC of the MELSEC FX family are counted in the octal numeral system. This is a base-8 number system and uses the digits 0 to 7.

The following table shows a comparison between some decimal and some octal numbers:

Decimal	Octal
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	10
9	11
10	12
11	13
12	14
13	15
14	16
15	17
16	20
:	:

Octal numbers are assigned as input/output numbers (X/Y) as shown below.

– X000 to X007, X010 to X017, X020 to X027......, X070 to X077, X100 to X107...

– Y000 to Y007, Y010 to Y017, Y020 to Y027......, Y070 to Y077, Y100 to Y107...

Numbers for added input/output unit/block

To an added input/output powered extension unit/block, input numbers and output numbers following the input numbers and output numbers given to the preceding device are assigned.

The last digit of the assigned numbers must begin with 0.

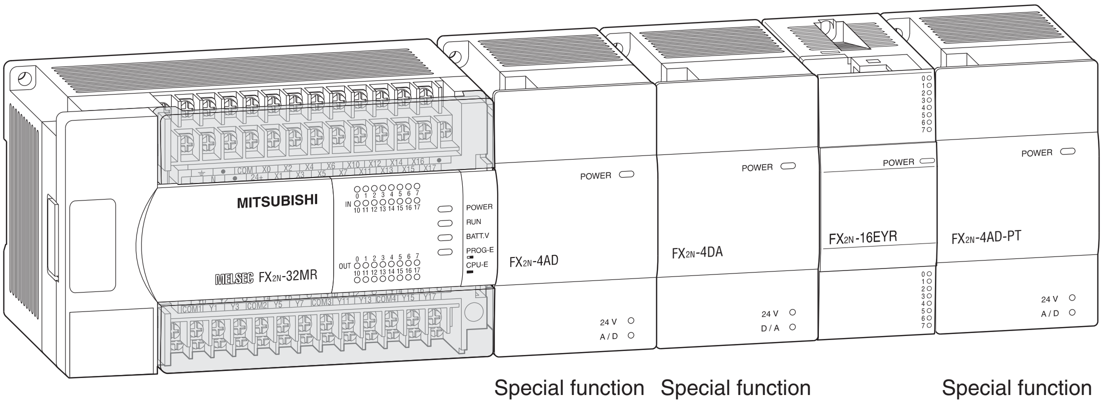
For example, when the last number on the preceding device is Y43, the output numbers are assigned to the next device starting from Y50.

X000 to X017	X020 to X037	X040 to X043*	X050 to X057
Base unit FX3U-32MR/ES	Input extension block FX2N-16EX-ES/UL (16 inputs)	Input/output extension block FX2N-8ER-ES/UL (4 inputs / 4 outputs)	Input extension block FX2N-8EX-ES/UL (8 inputs)
Y000 to Y017		Y020 to Y023*	

* The inputs from X044 to X047 and the outputs from Y024 to Y027 are occupied by the FX2N-8ER-ES/UL, but they can not used.

2.11.2 Special function module address

Since you can attach multiple special function modules to a single base unit each module needs to have a unique identifier so that you can address it to transfer data to and from it. Each module is automatically assigned a numerical ID in the range from 0 – 7 (you can connect a maximum of 8 special function modules). The numbers are assigned consecutively, in the order in which the modules are connected to the PLC.



Special function module 0

Special function module 1

Special function module 2

Special function module addresses are **not** assigned to the following products:

- Input/output powered extension units (e. g. FX2N-32ER-ES/UL or FX2N-48ET-ESS/UL)
- Input/output extension blocks (e. g. FX2N-16EX-ES/UL or FX2N-16EYR-ES/UL)
- Communication adapter (e.g. FX3U-CNV-BD)
- Interface adapter (e. g. FX3U-232-BD
- Special adapter (e. g. FX3U-232ADP)
- Extension power supply unit FX3U-1PSU-5V

3 Programming

3.1 Concepts of the IEC61131-3 Standard

IEC 61131-3 is the international standard for PLC programs, defined by the International Electromechanical Commission (IEC). It defines the programming languages and structuring elements used for writing PLC programs.

This system enables structured programs to be created using a high degree of modularisation. This provides increased efficiency, where tested programs and routines may be reused with a reduction of the number of programming errors.

Through use of structured programming techniques, IEC1131-3 eases fault finding procedures as individual operational program elements may be examined independently.

One important advantage of IEC61131-3 is that at assists in project management and quality control procedures. In particular, the structured methods encompassed within IEC61131-3 aid the **Validation** of processes incorporating PLC’s. In fact, in some industries it is now considered mandatory to adopt this approach of structured programming. This is commonplace in the Pharmaceutical and Petrochemical industries where some processes can be considered safety critical.

It is considered, in some quarters that the IEC method of programming requires excessive work to create the final code. However, it is generally accepted that the advantages a structured approach has to offer over “un-structured” and “open” programming techniques makes IEC61131-3 a worthwhile advantage.

PLCopen

PLCopen is an independent vendor and product organisation that has been established in order to further the use of IEC61131-3 throughout users of Industrial Control Systems. This organisation has defined 3 levels of compliancy for the design and implementation of systems to IEC61131-3.

PLCopen has established:

- an accreditation procedure
- accredited test institutes
- development test software, shared amongst members
- a defined certification procedure
- members with certified products

This assures compliancy now, and in the future.

PLCopen Certification

IEC 61131-3

Mitsubishi’s GX IEC Developer is fully compliant with PLCopen to “**Base Level IL**” (Instruction List) and “**Base Level ST”** (Structured Text) and has been fully certified to these standards.

3.2 Software Structure and Definition of Terms

In the following section, the primary terms used within GX IEC Developer will be defined:

- POU’s
- GLOBAL VARIABLES
- LOCAL VARIABLES
- USER DEFINED FUNCTIONS & FUNCTION BLOCKS
- TASK POOL
- PROGRAM EDITORS:
  - Instruction List
  - Ladder Diagram
  - Function Block Diagram
  - Sequential Function Chart
  - Structured Text
  - MELSEC Instruction List

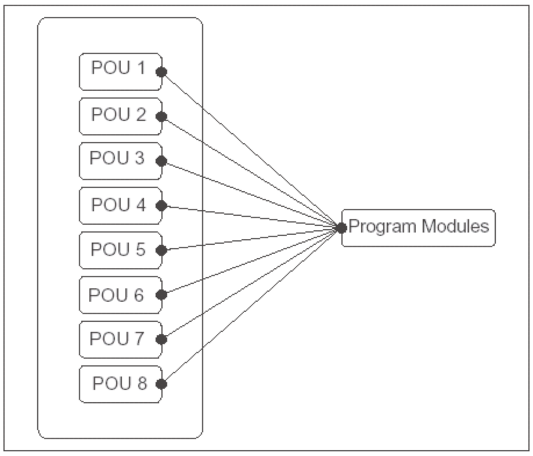
3.2.1 Definition of Terms in IEC61131-3

Projects

A Project contains the programs, documentation and parameters needed for an application.

POU - Program Organisation unit

The structured programming approach replaces the former unwieldy collection of individual instructions with a clear arrangement of the program into program modules. These modules are referred to as Program Organisation Units (POU’s), which form the basis of the new approach to programming PLC systems.



Program organisation units (POU’s) are used to implement **all** programming tasks.

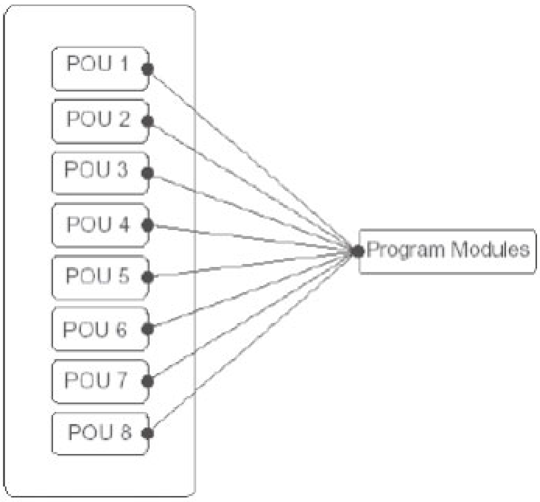
There are three different classes of POU’s, classified on the basis of their functionality:

- Programs
- Functions
- Function Blocks

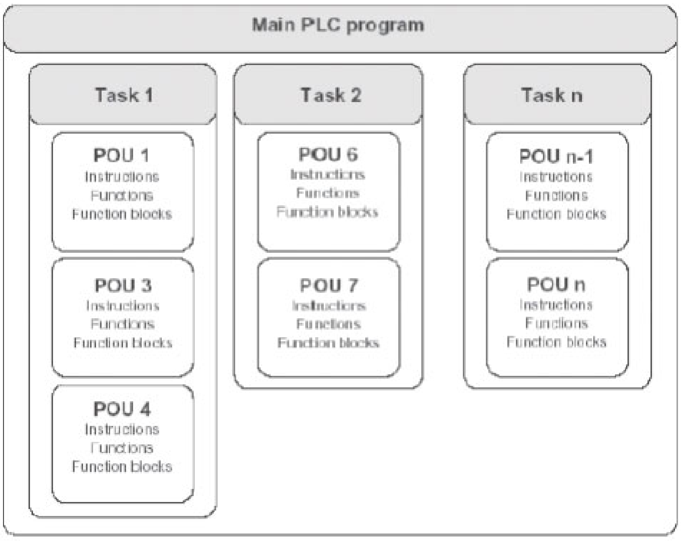
POU’s declared as Function Blocks can be considered as **programming instructions in their own right** and they can be used as such in every module of your programs.

The final program is compiled from the POU’s that you define as programs. This process is handled by the task management, in the Task Pool. Program POU’s are put together in groups referred to as “**Tasks**”.

Tasks

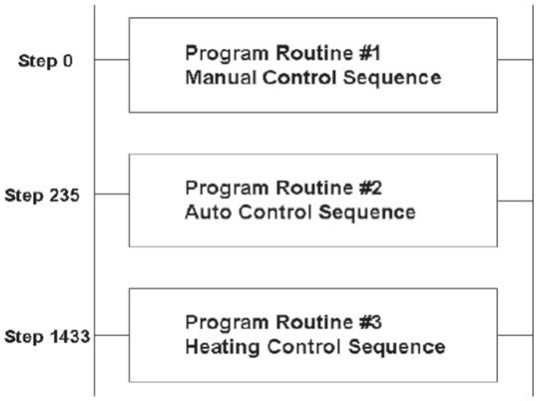


The Program POU’s are grouped together in tasks



In turn, all the tasks are grouped together to form the actual PLC program.

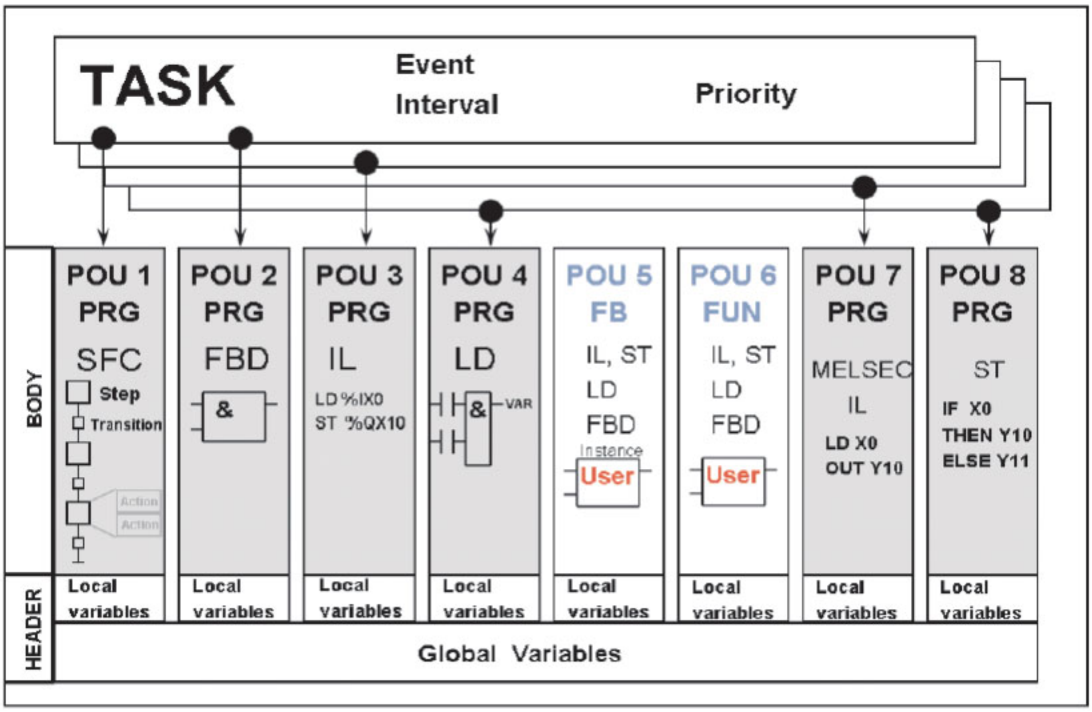
Most PLC programs consist of areas of code which perform specific tasks.They may form part of one large program, or be written in sub-routines, with program control instructions to select the current routine i. e. CALL, CJ etc.



Typical PLC program event sequence

In the above program, GX IEC Developer considers that each program routine which carries out a specific task to be a POU or program organisation unit.

Each POU can be written using any of the supported editors i.e. LD, IL, FBD, SFC, ST as shown below:



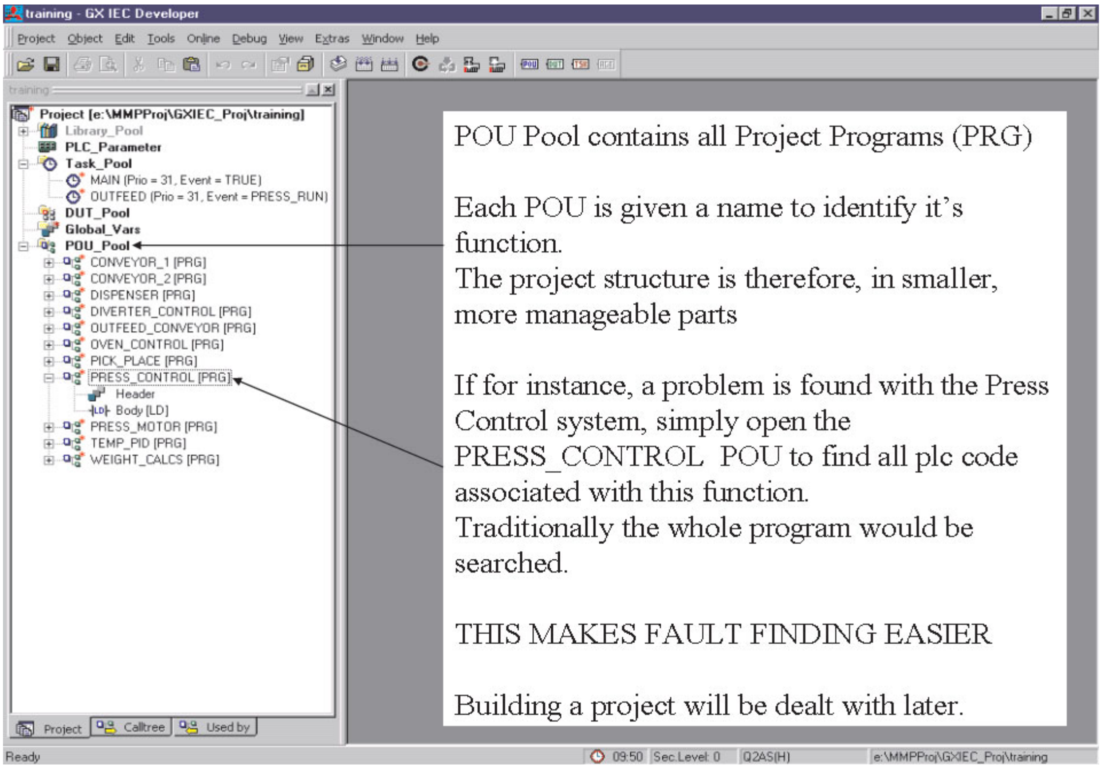
Overall Project Configuration illustrating POU integration using SFC, FBD, IL, LD and MELSEC IL and ST format programs.

POU Pool

A Project will consist of many POU's, each providing a dedicated control function and held in a POU Pool. Each POU could be written in any of the IEC editors. Therefore in any given project, the best language for the required function can be chosen. The compiler will assemble the project into code the PLC can understand but the user interface remains as written.

In this way, perhaps complicated interlocking routines, could be written in a ladder POU, whilst complex calculations or algorithms, might be better suited to one of the textual, or FDB editors.

It is the choice of the designer/user but this environment allows flexibility.



POU Pool contains all Project Programs (PRG)

Each POU is given a name to identify it's function.
The project structure is therefore, in smaller, more manageable parts

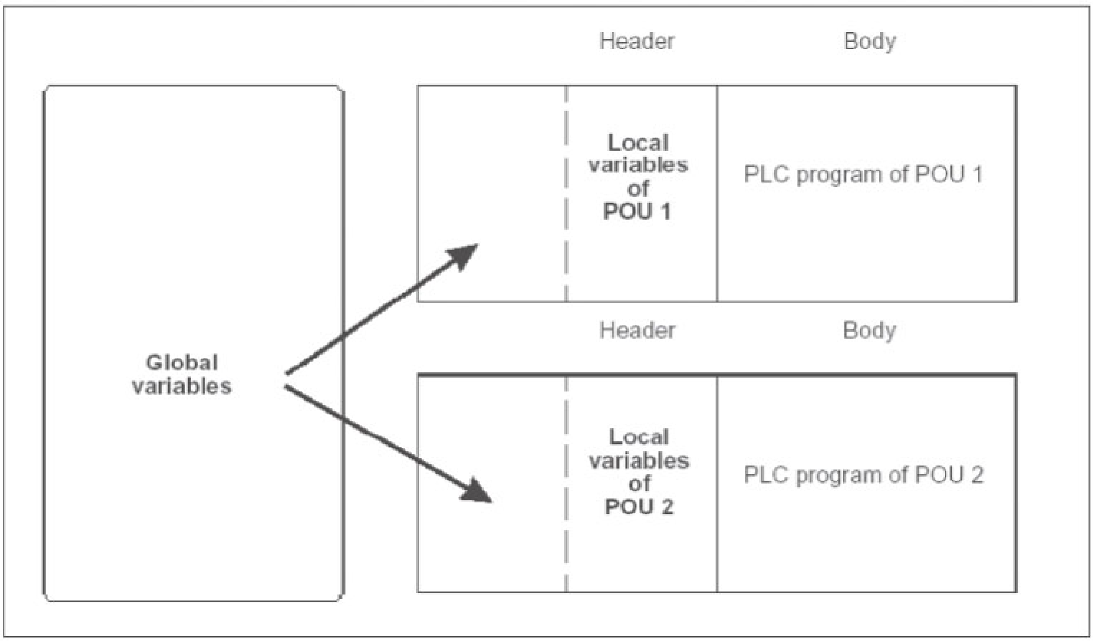
If for instance, a problem is found with the Press Control system, simply open the PRESS_CONTROL POU to find all plc code associated with this function.
Traditionally the whole program would be searched.

THIS MAKES FAULT FINDING EASIER

Building a project will be dealt with later.

Above an example of the GX-Developer display is shown illustrating an example POU Pool.

Composition of a POU



Definition of Variables – GLOBAL and LOCAL

- Variables

  Before a program can be constructed, it must be decided what variables are going to be required in each particular program module. Each POU has a list of Local Variables, which are defined and declared for use only for use within a particular POU. Global Variables can be used by all the POU's in the program and are declared in a separate list.

- Local Variables

  When program elements are declared as Local Variables, GX IEC Developer, automatically, uses some of its System Variables, as appropriate storage devices within a specific POU. These variables are exclusive to each POU and are not available to any other routine within a project.

- Global Variables

  Global Variables can be regarded as "shared" variables and are the interface to physical PLC devices. They are made available to all POU's and reference an actual physical PLC I/O or named internal devices within the PLC. External HMI and SCADA devices may interface with the user program using Global Variables.

IEC61131-3 Verses MELSEC Variables

GX IEC Developer supports program creation, using either symbolic declarations (tag names), or absolute Mitsubishi addresses (X0, M0 etc), assigned to the program elements.

The use of symbolic declarations complies with IEC 61131.3.

If symbolic declarations are used, then the tag names must be cross referenced to real PLC addresses.

Local Variable List

For a particular POU to access a Global Variable, it must be declared in its Local Variable List (LVL), in the POU Header.

The LVL can be made up of both Global Variables and Local Variables.

A Local Variable can be thought of as an intermediate result, i.e. if the program performs a five stage calculation, using three values and ending with one result, traditionally, the programmer would construct software, which produced several intermediate results, held in data registers before ending with the final register result.

It is likely that these intermediate results, serve no purpose other than for storage and only the final result is used elsewhere.

With GX IEC Developer, the intermediate results can be declared, as Local Variables and in this case, only the original three numbers and the result, declared as Global Variables.

The Global Variable List

The Global Variable List (GVL) provides the interface for all names, which relate to real PLC addresses, i.e. I/O data registers etc.

The GVL is available and can be read by all POU's created in the project.

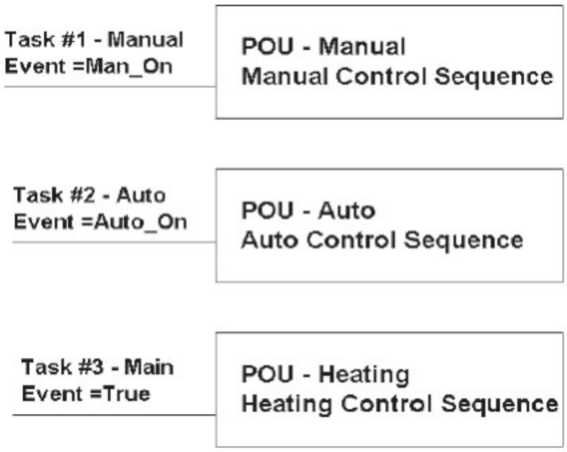
Task Pool and Task Manager

If we now think of our routines as POU's written for each function and given names, we can create a Task for each of our assigned POU's.

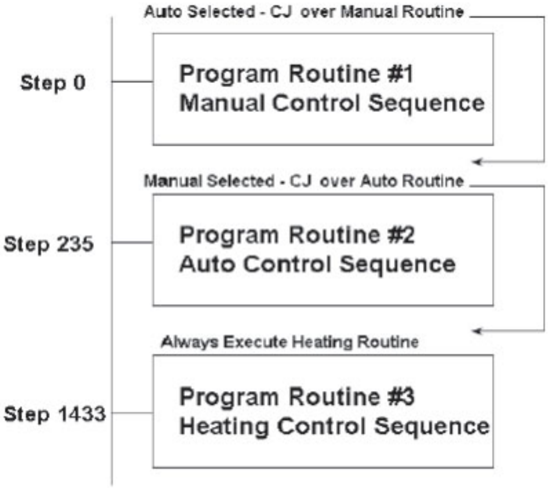
Each Task can have different operating conditions, or events.

- Task #1 only runs when a tag named, 'Man_On' is true.
- Task #2 only runs when a tag, named, 'Auto_On' is true.
- Task #3 runs all the time (event = True denotes this)

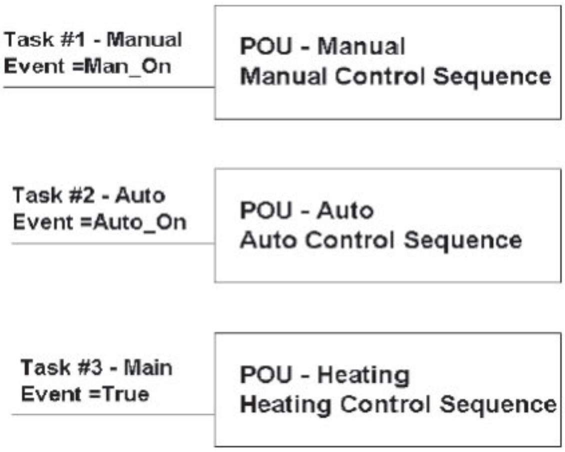
These tag names would be declared as Global Variables and assigned to PLC bit devices (they could be addresses i.e. X0).



Consider our original control program. Conditional Jump (CJ) instructions could be used to isolate, either routines #1 or #2, when not in use. The Heating control routine is always required to run.



If these routines are considered as tasks, then routines #1 & #2, are driven by event, i.e. when either auto or manual is selected, whereas, routine #3 is always on.



Task #1 - Manual
Event =Man_On

POU - Manual
Manual Control Sequence

Task #2 - Auto
Event =Auto_On

POU - Auto
Auto Control Sequence

Task #3 - Main
Event =True

POU - Heating
Heating Control Sequence

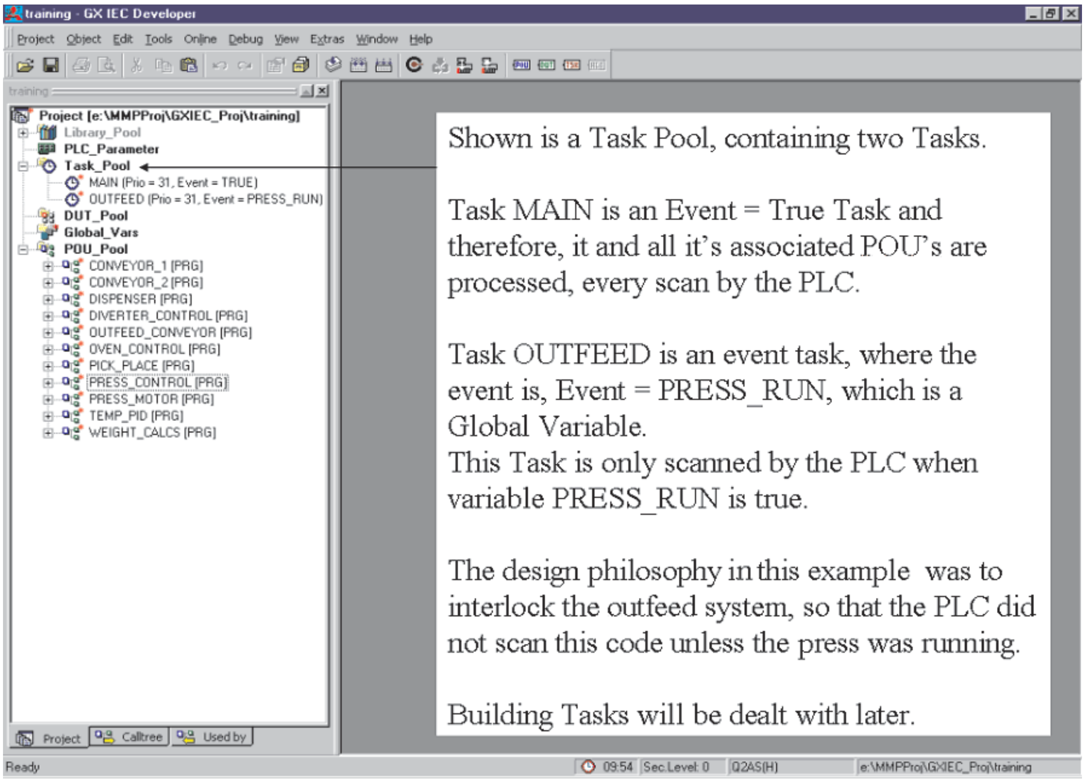
When GX IEC Developer compiles the project, it automatically inserts, program branching instructions, into the program, in line with event driven tasks.

A Task can have more than one POU assigned to it, typically, a task where Event = True, would contain all POU's which needed to operate every scan of the PLC. A POU of a particular name cannot be assigned to more than one task in any one project.

NOTE

Any POU's **not** assigned to Tasks, ARE NOT SENT TO THE PLC during program transfer. Don't forget – this applies to the default download. Tasks can be prioritised, either on a time or interrupt basis.

The **Task Pool** contains all the assigned tasks in the project.



Shown is a Task Pool, containing two Tasks.

Task MAIN is an Event = True Task and therefore, it and all it's associated POU's are processed, every scan by the PLC.

Task OUTFEED is an event task, where the event is, Event = PRESS_RUN, which is a Global Variable.
This Task is only scanned by the PLC when variable PRESS_RUN is true.

The design philosophy in this example was to interlock the outfeed system, so that the PLC did not scan this code unless the press was running.

Building Tasks will be dealt with later.

The **Task Pool** allows the user to efficiently manage the PLC scan, ensuring that only the routines that require scanning are executed. It also provides an easy method of allocating specific routines to events and timed or priority interrupts.

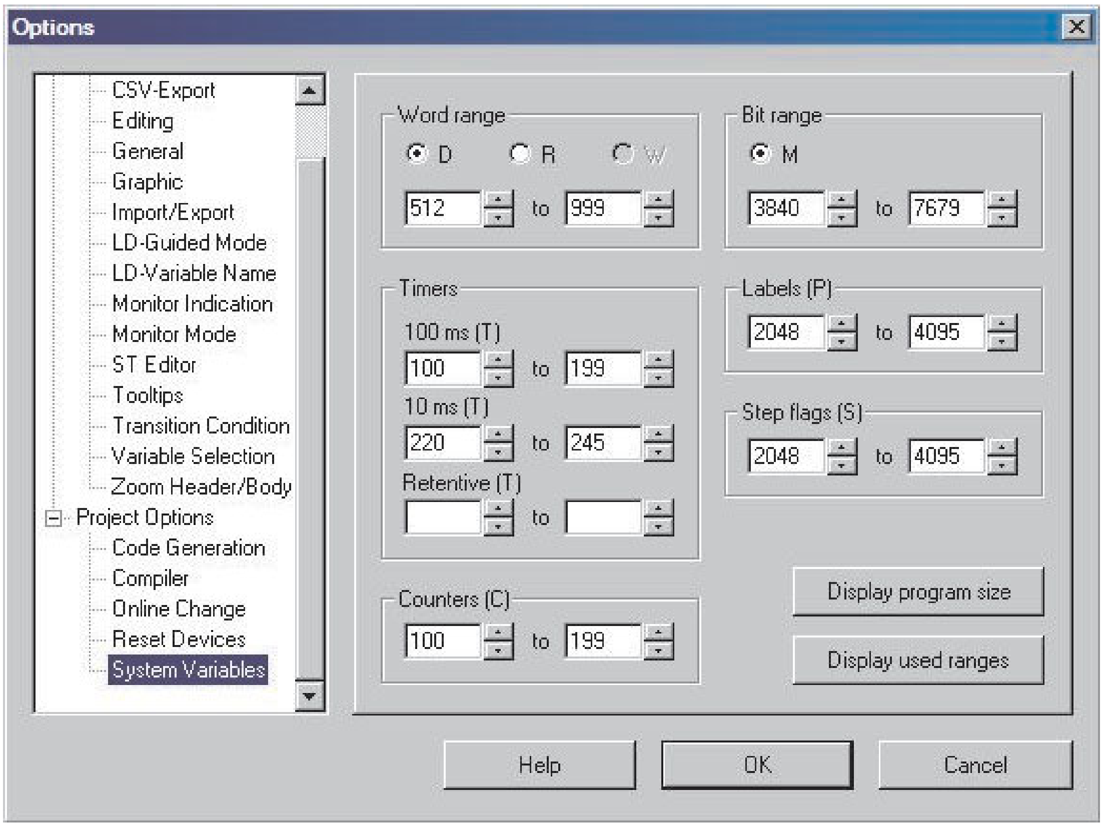
The software engineer need only be concerned about the program content, not whether the branch instructions are correct and obey the rules.

Machines/processes, consisting of standard parts, can have individual POU’s written for each part. The full machine may consist of many POU’s.

For each variant of the machine, the supplier can choose to assign to the Task Manager, only the relevant POU’s, for that machine, as only POU’s assigned will be transferred to the PLC on download.

3.2.2 System Variables

The device ranges that GX IEC Developer allocated to system variables can be edited here. This feature is displayed using the ***Options*** command under the ***Extras*** menu:



Options

- CSV-Export
- Editing
- General
- Graphic
- Import/Export
- LD-Guided Mode
- LD-Variable Name
- Monitor Indication
- Monitor Mode
- ST Editor
- Tooltips
- Transition Condition
- Variable Selection
- Zoom Header/Body
- Project Options
  - Code Generation
  - Compiler
  - Online Change
  - Reset Devices
  - System Variables

Word range: ● D ○ R ○ W — 512 to 999

Bit range: ● M — 3840 to 7679

Timers
100 ms (T): 100 to 199
10 ms (T): 220 to 245
Retentive (T): to

Labels (P): 2048 to 4095

Step flags (S): 2048 to 4095

Counters (C): 100 to 199

Systen variable ranges for the actual project.

- Word range

  D: D devices are used as word system variables.

  R: R devices are used as word system variables.

  W: W devices are used as word system variables.

  From/to: PLC type dependant, as defined in the parameters.

- Timers

  Standard (T) – From/to: PLC type dependant, as defined in the parameters.

  Retentive (ST) – From/to: PLC type dependant, as defined in the parameters.

- Counters (C)

  From/to: PLC type dependant, as defined in the parameters.

- Bit range

  M: M devices are used as bit system variables.

  From/to: PLC type dependant, as defined in the parameters.

- Labels (P)

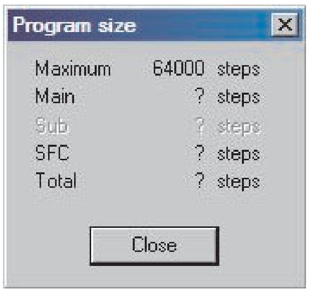
  From/to: PLC type dependant, as defined in the adequate CNF file

- Step flags (S)

  From/to: PLC type dependant, as defined in the adequate TYP file

- Display program size

  A summary of the used program size is displayed on a separate dialogue box.



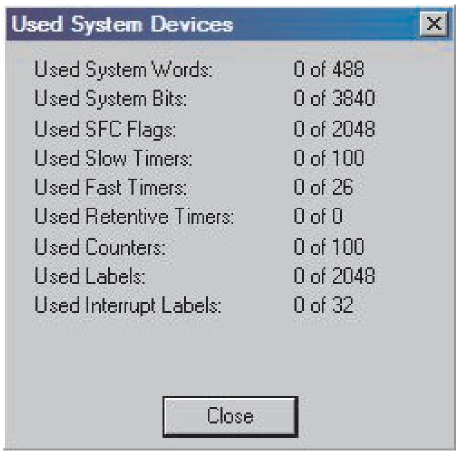
Program size

Maximum	64000	steps
Main	?	steps
Sub	?	steps
SFC	?	steps
Total	?	steps

If the program is not compiled the dialogue shows a "?" character instead of the program size. If SFC or SUB programs are not available for this CPU, the correspondent line will be grayed.

- Display used ranges

  A summary of the used system variables ranges is displayed on a separate dialogue box.



Used System Devices

Used System Words:	0 of 488
Used System Bits:	0 of 3840
Used SFC Flags:	0 of 2048
Used Slow Timers:	0 of 100
Used Fast Timers:	0 of 26
Used Retentive Timers:	0 of 0
Used Counters:	0 of 100
Used Labels:	0 of 2048
Used Interrupt Labels:	0 of 32

3.2.3 System Labels

System Labels, shown in the system variable list in chapter 3.2.2 are used by GX IEC Developer for internal management of the project. GX IEC Developer allocates system labels for the following:

- Network Labels
- Event Driven Task (not EVENT = TRUE)
- User Defined Function blocks (one per function block – unless macro code)
- System Timers (These are used by the Task Manager, for interval triggered tasks and local Timers.)

3.3 Programming Languages

GX IEC Developer provides separate editors for all the following programming languages, which can be used to program the bodies of your programs:

Text Editors

- Instruction List (IEC and MELSEC)
- Structured Text

Graphic Editors

- Ladder Diagram
- Function Block Diagram
- Sequential Function Chart

With the exception of the Sequential Function Chart language, all the editors divide PLC programs into sections, referred to as "Networks". These Networks can be given names (labels), which can consist of up to a maximum of 8 characters terminated with a colon (:). These networks are numbered consecutively and can be used as destinations for branching commands.

3.3.1 Text Editors

Instruction List (IL)

The Instruction List (IL) work area is a simple text editor with which the instructions are entered directly.

An Instruction List consists of a sequence of statements or instructions. Each instruction must contain an operator (function) and one or more operands. Each instruction must begin in a new line. You can also add optional Labels, Modifiers and comments to each instruction.

Two different types of Instruction List are used:

- IEC Instruction List

  IEC Instruction Lists are entered and edited in exactly the same way as MELSEC Instruction Lists. The following programming differences need to be observed, however:

  – MELSEC networks in IEC IL

    You can include MELSEC networks in IEC Instruction Lists, thus providing access to the MELSEC system instructions.

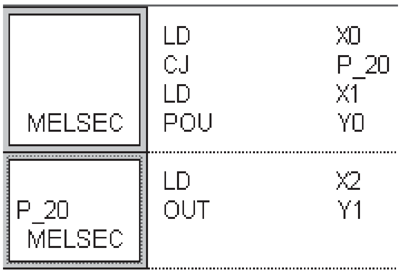
  – The accumulator

    The accumulator is a result management system familiar from high-level languages. The result of every operation is stored in the bit accumulator directly after execution of the instruction. The accumulator always contains the operation result of the last instruction executed. You do not need to program any input conditions (execution conditions) for the operations; execution always depends on the content of the accumulator.

For more information about IEC Instruction List, please refer to chapter 15.

- MELSEC Instruction List

  MELSEC Instruction Lists are entered and edited in exactly the same way as IEC Instruction Lists. However, you can only use the MELSEC instruction set; IEC standard programming is not possible.



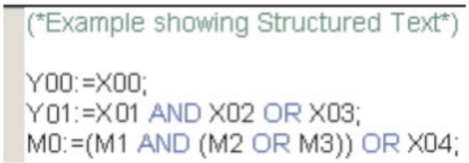
```
MELSEC     LD      X0
           CJ      P_20
           LD      X1
           POU     Y0

P_20       LD      X2
MELSEC     OUT     Y1
```

Example of a MELSEC Network

Structured Text

Structured Text is a helpful tool. Especially programmers coming from the PC world will enjoy this tool. If they program carefully and think about the way of working by PLC, they will be glad with this editor.

The Structured Text editor is compatible to the IEC 61131-3, all requirements are fulfilled.



```
(*Example showing Structured Text*)

Y00:=X00;
Y01:=X01 AND X02 OR X03;
M0:=(M1 AND (M2 OR M3)) OR X04;
```

Example for Structured Text

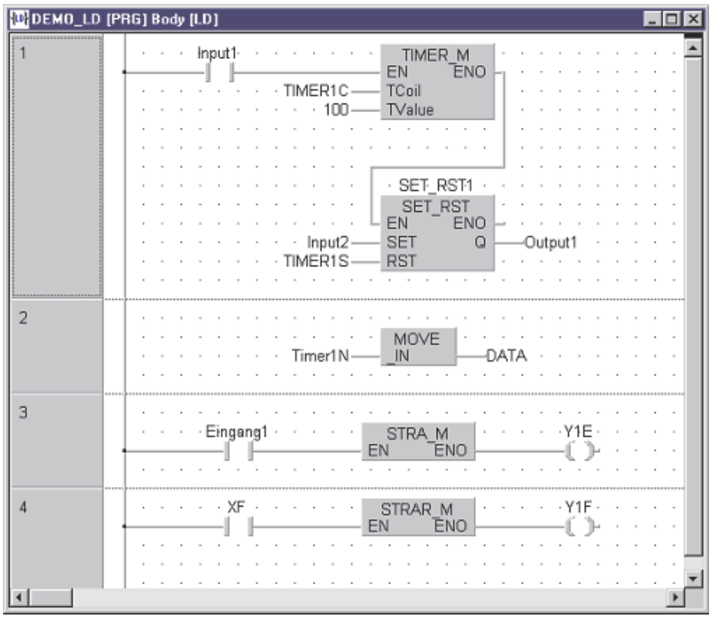
An example of Structured Text programming is given in chapter 16.

3.3.2 Graphic Editors

Ladder Diagram

A Ladder Diagram consists of input contacts (makers and breakers), output coils, function blocks and functions. These elements are connected with horizontal and vertical lines to create circuits. The circuits always begins at the bus bar (power bar) on the left.

Functions and function blocks are displayed as blocks in the diagram. In addition to the normal input and output parameters, some blocks also have a Boolean input (EN = ENable) and output (ENO = ENable Out). The status at the input always corresponds to that at the output.



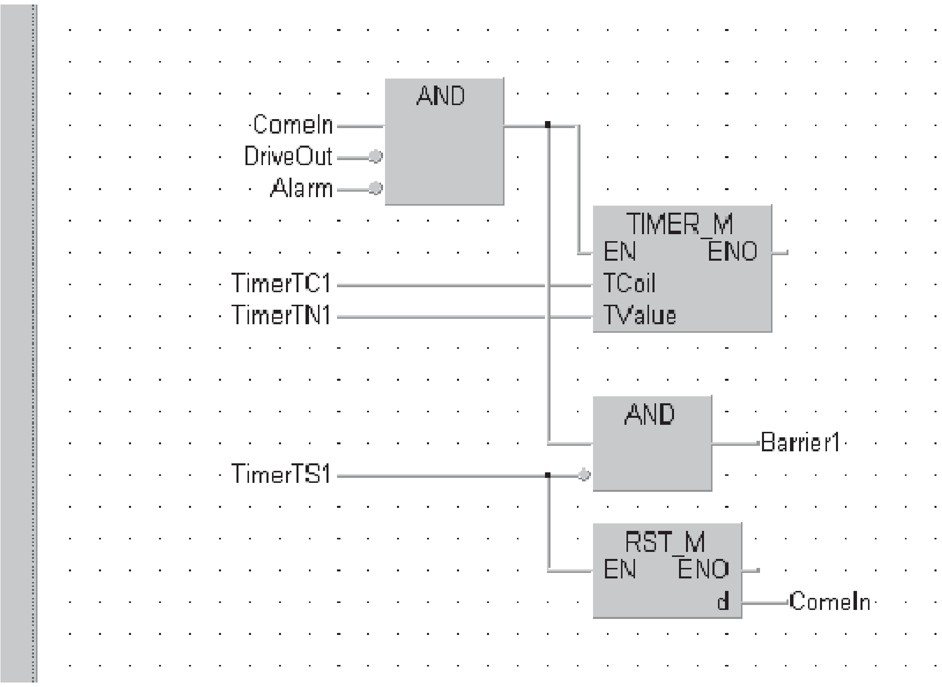
Example for Ladder diagram

Function Block Diagram

All instructions are implemented using blocks, which are connected with one another with horizontal and vertical connecting elements. There are no power bars.

In addition to the normal input and output parameters, some blocks also have a Boolean input (EN = ENable) and output (ENO = ENable Out). The status of the input always corresponds to the output status.

Example for Function Block Diagram:

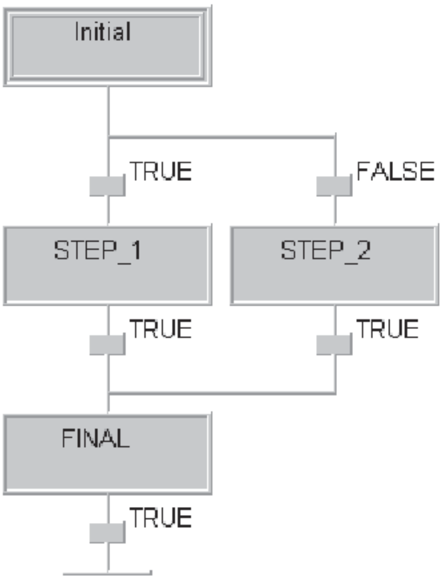


Sequential Function Chart

Sequential Function Chart is one of the graphical languages. It can be regarded as a structuring tool with which the sequential execution of processes can be represented clearly and comprehensible.

The only possible program organisation unit in SFC is the program.

Sequential Function Chart has two basic elements, Steps and Transitions. A sequence consists of a series of steps, each step separated from the next by a transition. Only one step in the sequence can be active at any one time. The next step is not activated before the previous step has been completed and the transition is satisfied.



Example for Sequential Function Chart

3.4 Data Types

GX IEC Developer supports the following data types.

3.4.1 Simple Types

Data type		Value range		Size	Applicable Devices / PLCs
BOOL	Boolean	Bit Device	0 (False), 1 (True)	1 bit	X, Y, M, B
INT	Integer	Register	-32768 to +32767	16 bit	D, W, R
DINT	Double Integer		-2,147,483.648 to 2,147,483,647	32 bit	
WORD	Bit String	K4M0	0 to 65,535	16 bit	X, Y, M. B
DWORD		K8M0	0 to 4,294,967,295	32 bit	
REAL	Floating point value	7 digits		32 bit	FX2N, FX3U
STRING	Character String	20 Characters (default)		32 bit	FX3U
TIME	Time value	-T#24d0h31m23s64800ms to T#24d20h31m23s64700 ms		32 bit	FX3U only

3.4.2 Complex Data Types

ARRAYS

An array is a field or matrix of variables of a particular type.

For example, an **ARRAY [0..2] OF INT** is a one dimensional array of three integer elements (0,1,2). If the start address of the array is D0, then the array consists of D0, D1 and D2.

Identifier	Address	Type	Length
Motor_Volts	D0	ARRAY	[0...2] OF INT

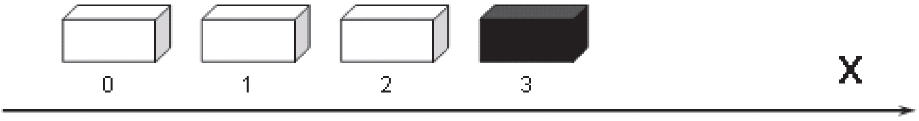
In software, program elements can use: Motor_Volts[1] and Motor_Volts[2], as declarations, which in this example mean that D1 and D2 are addressed.

Arrays can have up to three dimensions, for example: ARRAY [0...2, 0...4] has three elements in the first dimension and five in the second.

Arrays can provide a convenient way of "indexing" tag names, i.e. one declaration in the Local or Global Variable Table can access many elements.

The following diagrams illustrate graphical representation of the three array types.

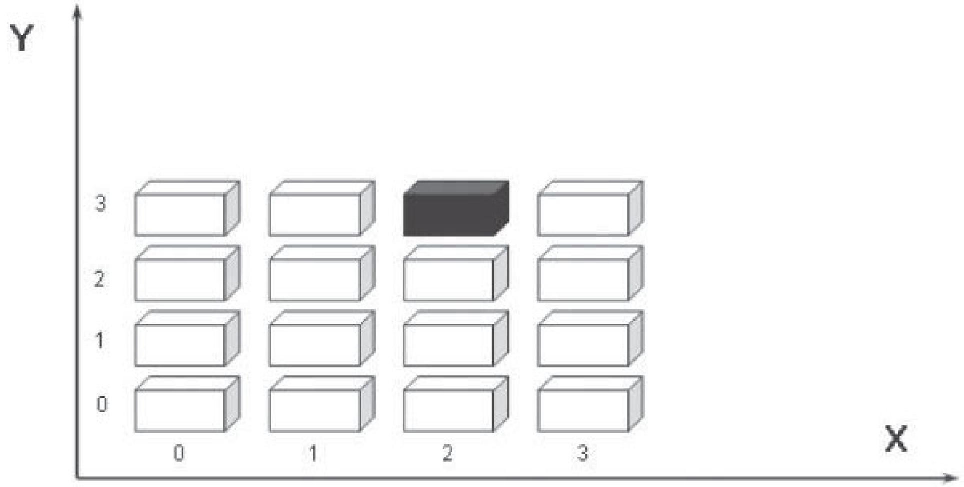
Single Dimensional Array



Identifier	Type
Motor_Speed	**ARRAY [0..3] OF INT**

■ **= Motor_Speed [3]**

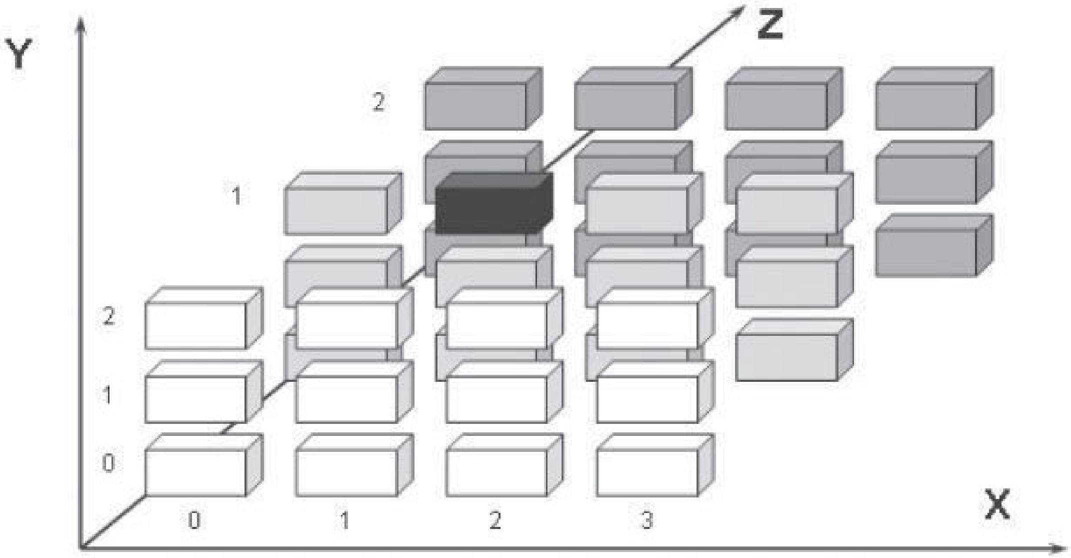
Two Dimensional Array



Identifier	Type
Motor_Volt	**ARRAY [0..3, 0...3] OF INT**

■ **= Motor_Volt [2, 3]**

Three Dimensional Array



Identifier	Type
Motor_Current	**ARRAY [0..3, 0...2, 0..2] OF INT**

■ **= Motor_Current [1, 2, 1]**

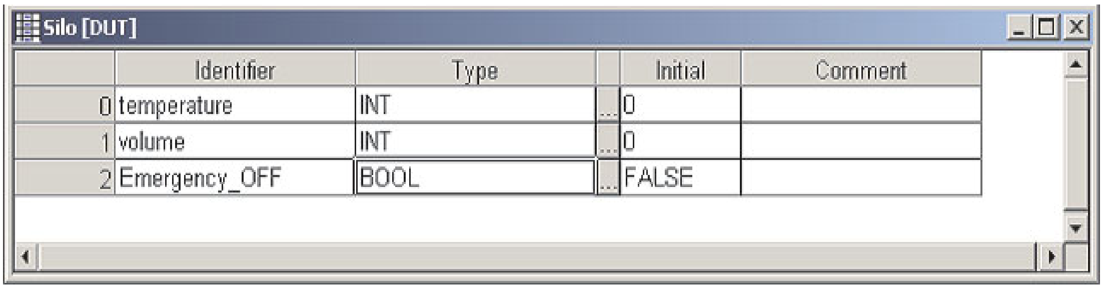
Data Unit Types (DUT)

User defined Data Unit Types (DUT), can be created. This can be useful for programs which contain common parts, for example; the control of six identical silos. Therefore a data unit type, called ‘Silo’ can be created, composing patterns of different elements, i.e. INT, BOOL etc.

When completing a global variable list, identifiers of type Silo can be used. This means that the predefined group called ‘Silo’ can be used with the elements defined as required for each silo, thus reducing design time and allowing re-use of the DUT.

Example use of a DUT

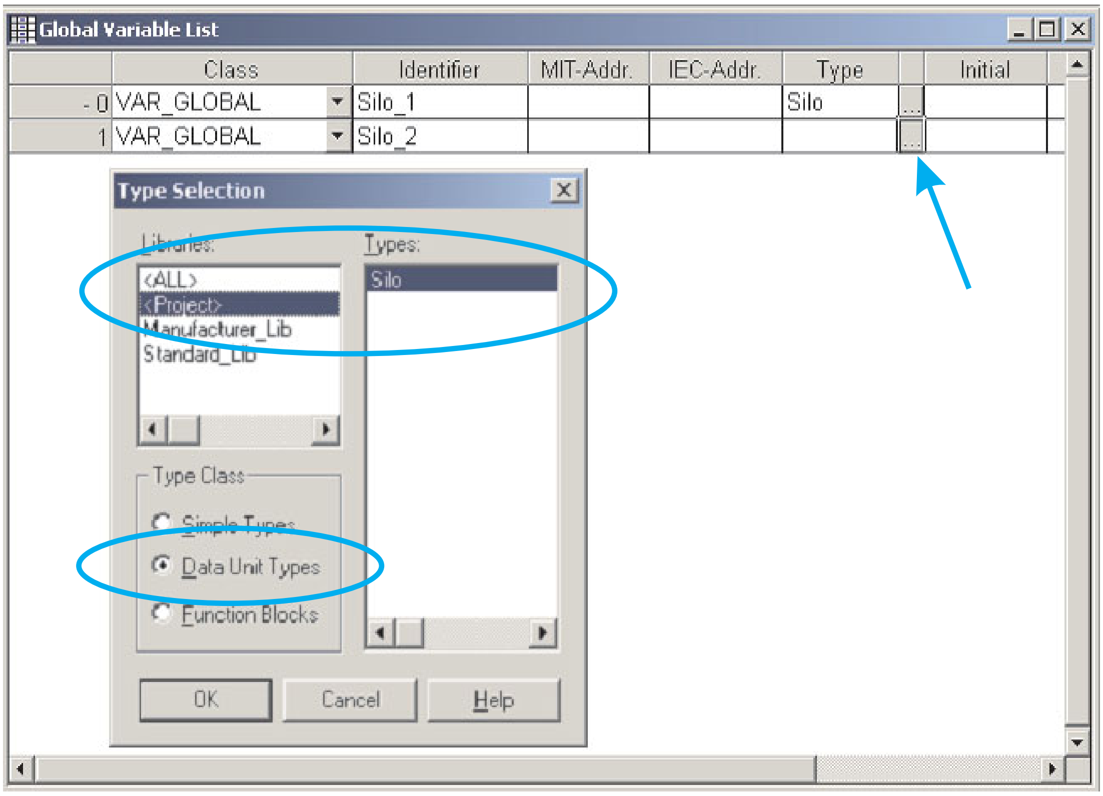
The following example shows the creation of a data type called Silo. The variable collection of Silo contains two variables of the INT and one variable of the type BOOL.



	Identifier	Type		Initial	Comment
0	temperature	INT	...	0	
1	volume	INT	...	0	
2	Emergency_OFF	BOOL	...	FALSE	

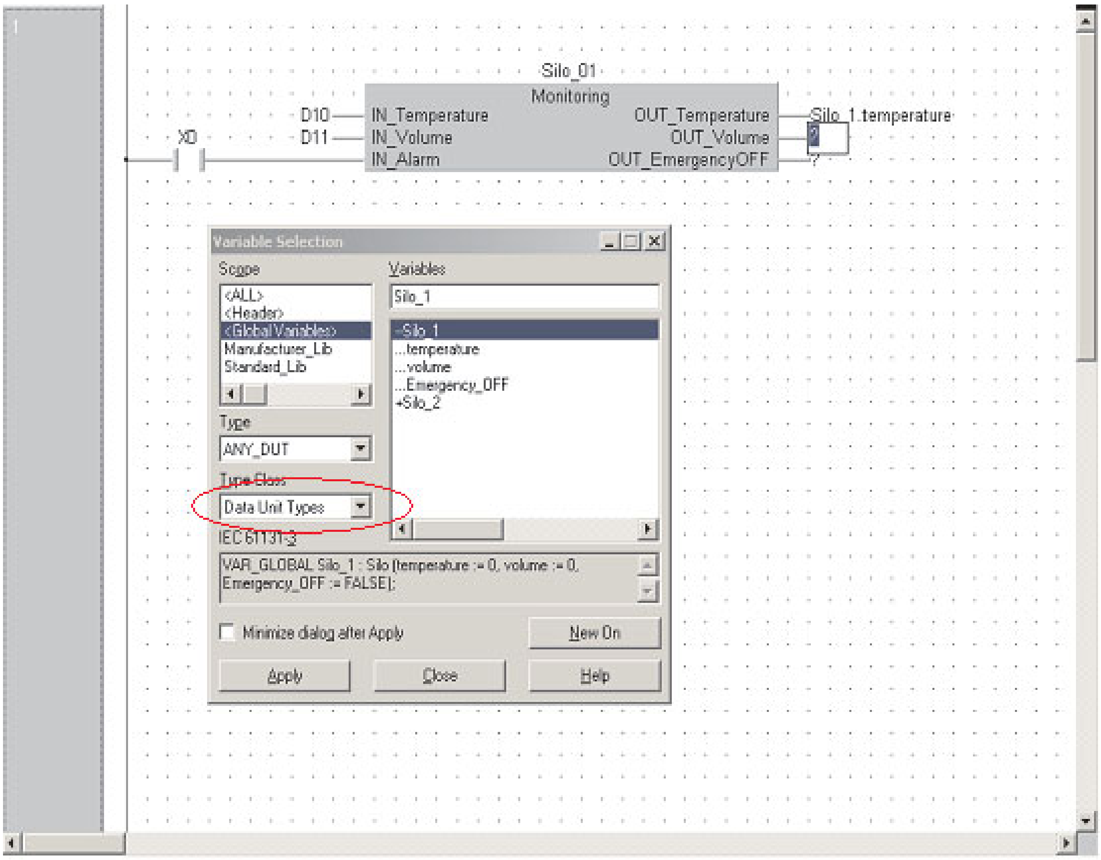
How to declare the DUT

Double-click on ***Global_Vars*** in the Project Navigator window and enter the following lines in the global variables declaration table.



	Class	Identifier	MIT-Addr.	IEC-Addr.	Type		Initial
0	VAR_GLOBAL	Silo_1			Silo	...	
1	VAR_GLOBAL	Silo_2				...	

The variables are stored in the Global Variable List. The structure of both variables, Silo_1 and Silo_2, is identical, so to reference the individual variable of each DUT you only need to prefix their names with the name of the respective global variable.



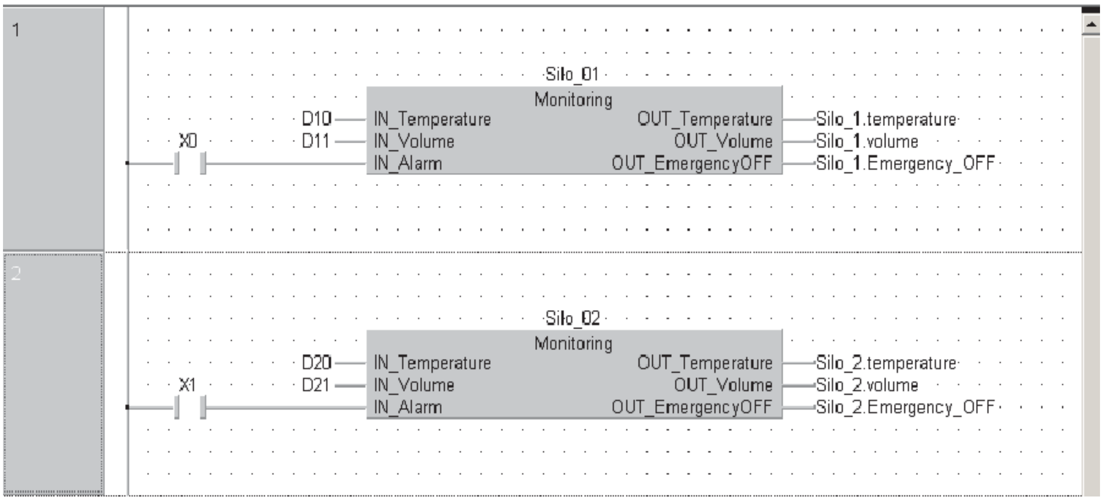
Variables: Silo_1

Type: ANY_DUT

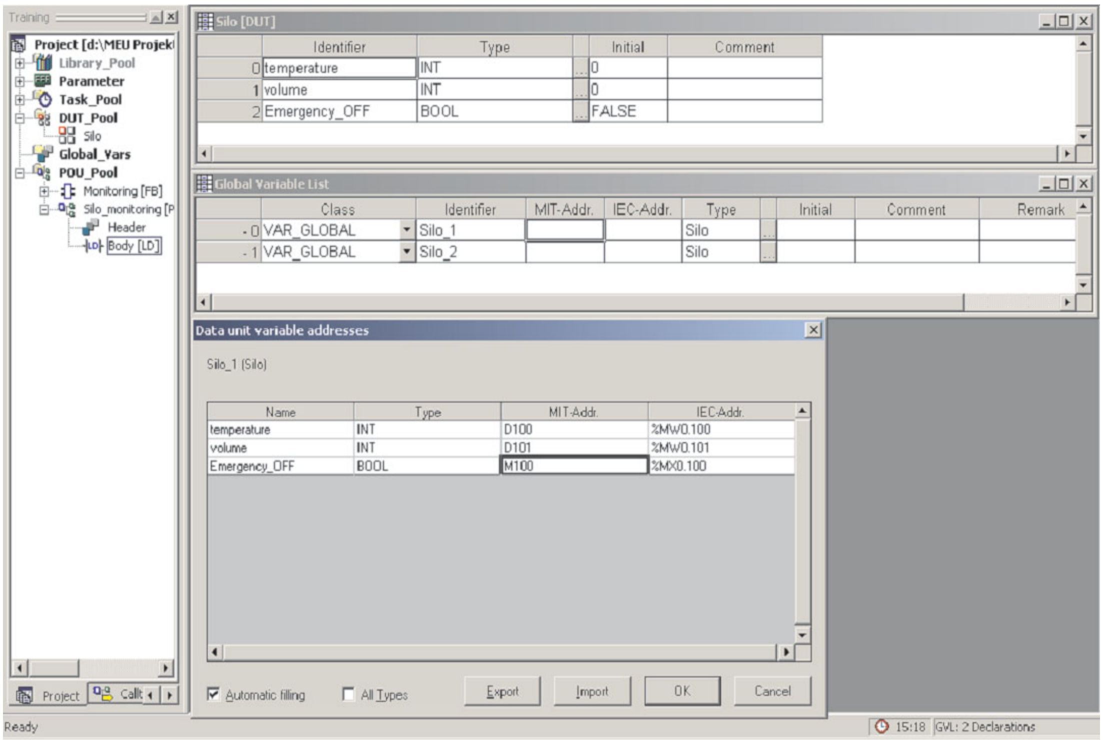
Type Class: Data Unit Types

```
VAR_GLOBAL Silo_1 : Silo (temperature := 0, volume := 0,
Emergency_OFF := FALSE);
```

In this example a function block of the type “Monitoring” has been programmed for assigning the register value and the Boolean input to the elements of the DUTs. Two separate instances (Silo_01 and Silo_02) of this function blocks were then created for two silos.



The GVL has been extended to define addresses for all elements of data unit types. Not defined addresses are handled by the system.



Silo [DUT]

	Identifier	Type		Initial	Comment
0	temperature	INT	...	0	
1	volume	INT	...	0	
2	Emergency_OFF	BOOL	...	FALSE	

Global Variable List

	Class	Identifier	MIT-Addr.	IEC-Addr.	Type		Initial	Comment	Remark
- 0	VAR_GLOBAL	Silo_1			Silo	...			
- 1	VAR_GLOBAL	Silo_2			Silo	...			

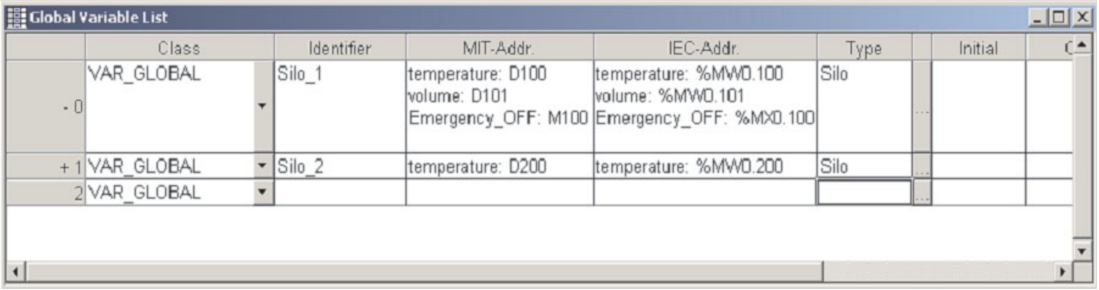
Data unit variable addresses

Silo_1 (Silo)

Name	Type	MIT-Addr.	IEC-Addr.
temperature	INT	D100	%MW0.100
volume	INT	D101	%MW0.101
Emergency_OFF	BOOL	M100	%MX0.100

☑ Automatic filling ☐ All Types Export Import OK Cancel

To view all definitions at once (if more than one definition is available), DUT entries in the GVL can be expanded by double-clicking the row number field.



Global Variable List

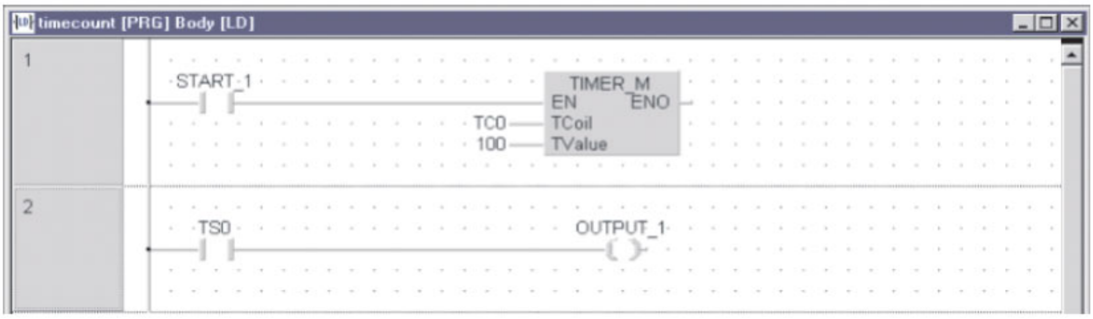
	Class	Identifier	MIT-Addr.	IEC-Addr.	Type		Initial	C
- 0	VAR_GLOBAL	Silo_1	temperature: D100 volume: D101 Emergency_OFF: M100	temperature: %MW0.100 volume: %MW0.101 Emergency_OFF: %MX0.100	Silo	...		
+ 1	VAR_GLOBAL	Silo_2	temperature: D200	temperature: %MW0.200	Silo	...		
2	VAR_GLOBAL					...		

3.4.3 MELSEC Timers and Counters

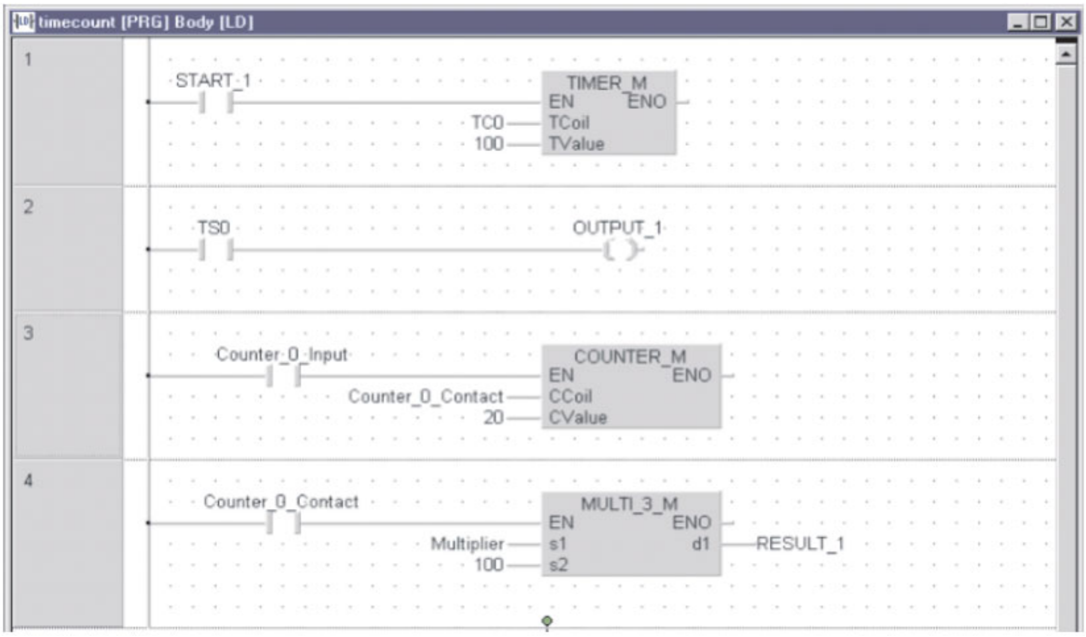
When programming standard Timers/Counters, an IEC convention must be observed:

Timer/Counter **Coil** is programmed:	**TCn / CCn**
Timer/Counter **Contact** is programmed:	**TSn / CSn**
Timer/Counter **Value** is programmed:	**TNn / CNn**

In the following example T0 becomes TC0 and TS0. In this case Mitsubishi addresses have been used, it is therefore vital to check the System Variable default T/C usage:



In the following example, the counter has been programmed using identifiers which would have to be declared in the Global and Local Variable tables:



4 Building a Project

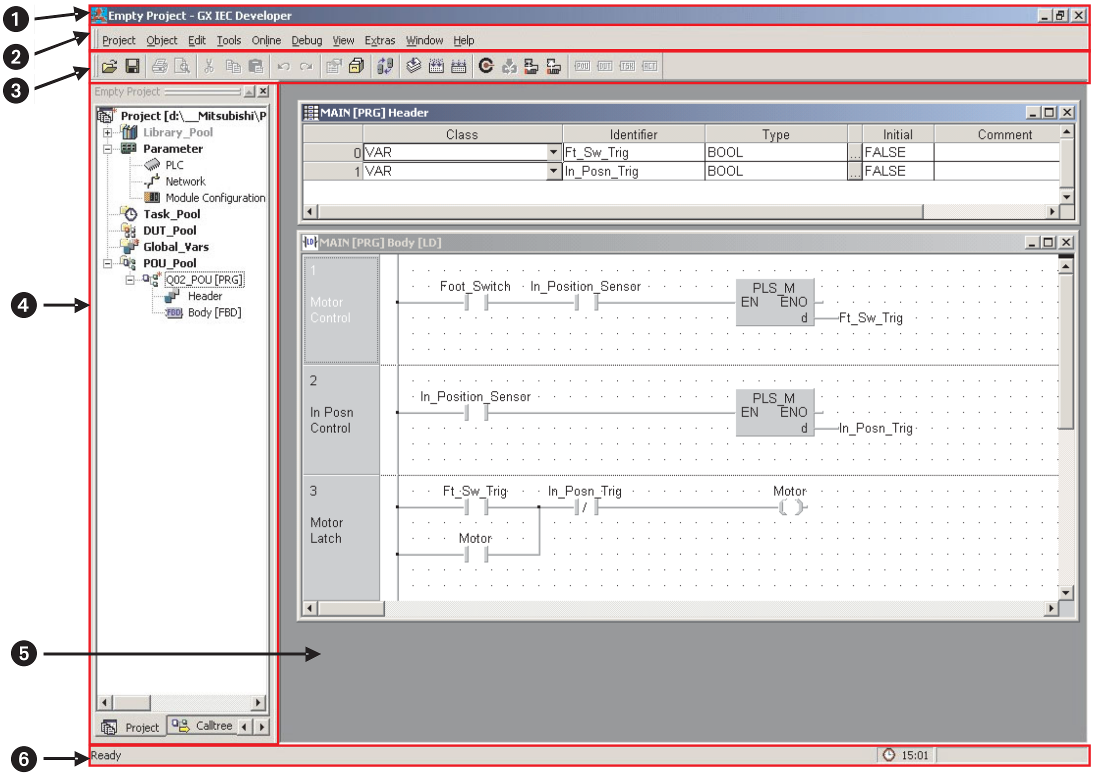
In the next section, we will build our first project, initially using the Ladder Diagram editor.

Topics covered

- Using the Project Navigator
- Using the GVL with identifiers
- Declaring variables in the Program Header
- Creating programs with the IEC ladder editor
- Programming IEC Timers/Counters
- Commenting and Documentation
- Downloading and Monitoring

4.1 Starting GX IEC Developer

After starting GX IEC Developer from Windows, the following window will be displayed:



	Class	Identifier	Type	Initial	Comment
0	VAR	Ft_Sw_Trig	BOOL	FALSE	
1	VAR	In_Posn_Trig	BOOL	FALSE	

❶ Application Title Bar
The Application title bar gives you the name of the open project.

❷ Menu Bar
The Menu Bar provides access to all the menus and commands used to control GX IEC Developer. When you select one of the entries in the bar by clicking with the mouse, a menu of options drops down. Options marked with an arrow contain submenus, which are displayed with additional options when you click on them. Selecting commands normally opens a dialog or entry box.

GX IEC Developer' menu structure is context-sensitive, changing depending on what you are currently doing in the program. Commands displayed in light grey are currently unavailable.

❸ Tool Bar
The Tool Bar icons give you direct access to the most-used commands with a single mouse click. The Tool Bar is context-sensitive, displaying a different collection of icons depending on what you are currently doing in the program.

❹ Project Navigator Window
The Project Navigator is the control centre of GX IEC Developer. The Project Navigator window is not displayed until you open an existing project or create a new one.

❺ Editor (Body)
In this area the POUs can be edited. Each POU consists out of a Header and a Body.

- Header

  A header is an integral part of a program organisation unit (POU). It is the place where the variables to be used in the POU must be declared.

- Body

  A body is an integral part of a program organisation unit (POU). It contains the code elements and syntax of the actual program, function block or function.

❻ Status Bar

This bar displayed at the bottom of the screen gives you useful information on the current status of your project. Status Bar display can be enabled or disabled, and you can also configure the individual display options to suit your needs.

4.2 Application Program

4.2.1 Example: Carousel Indexer

The following application program will be used to illustrate the creation of a simple program using the tools of GX IEC Developer.

Operational Sequence

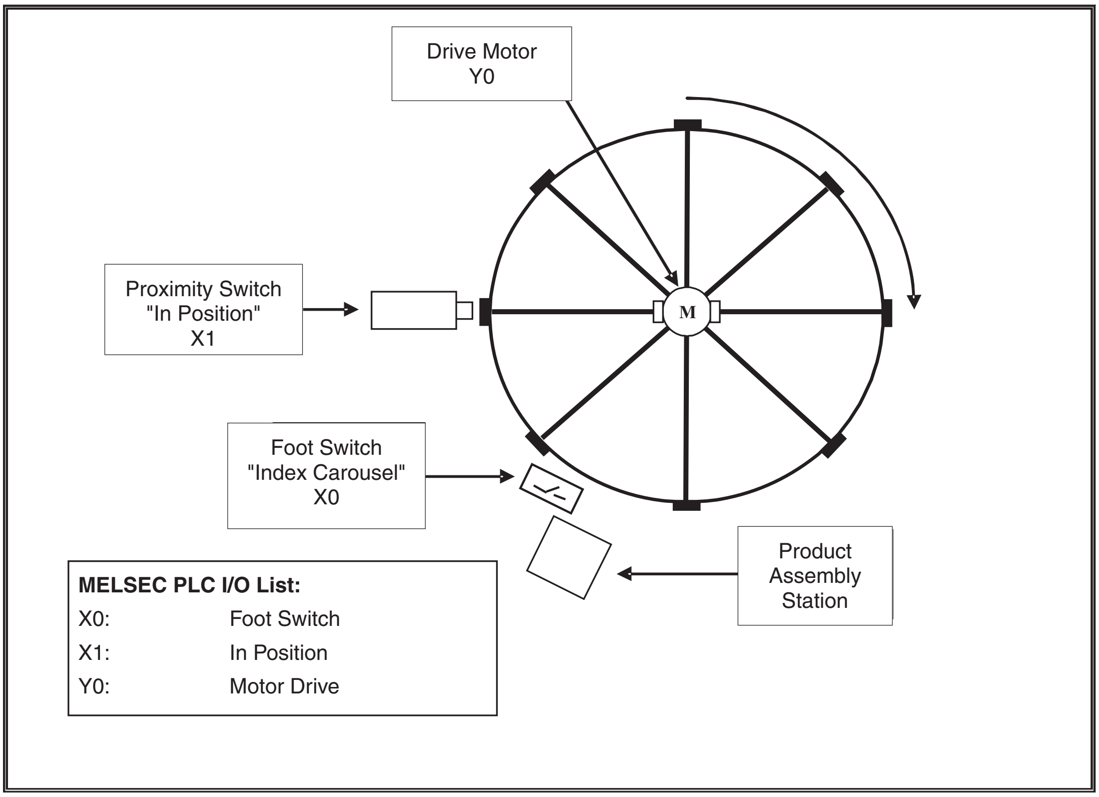
① Momentarily operate foot switch to index carousel.

② Carousel rotates – 'In-Position' sensor turns OFF as carousel begins rotating.

③ 'In-Position' sensor turns ON when carousel reaches index position.

④ Assemble product

⑤ Repeat process (Go back to ①.)



MELSEC PLC I/O List:

X0:	Foot Switch
X1:	In Position
Y0:	Motor Drive

There are a number of issues that must be addressed when designing a PLC program for the above application. Using a standard Start / Stop circuit is not possible without modification due to the following difficulties:

- The foot switch may be operated at random. Once activated, it may be possible for the operator to forget to release the switch which may cause the table to continue to rotate past its index position.
- Once "In-Position" X1 operates, it remains on, thus the table is prevented from re-indexing.

The design must therefore contain interlocks to prevent miss-operation as described above. An alternative approach to the design would suggest the use of 'Pulse Transition Logic' by means of the IEC or MELSEC "Edge Triggered" configurations.

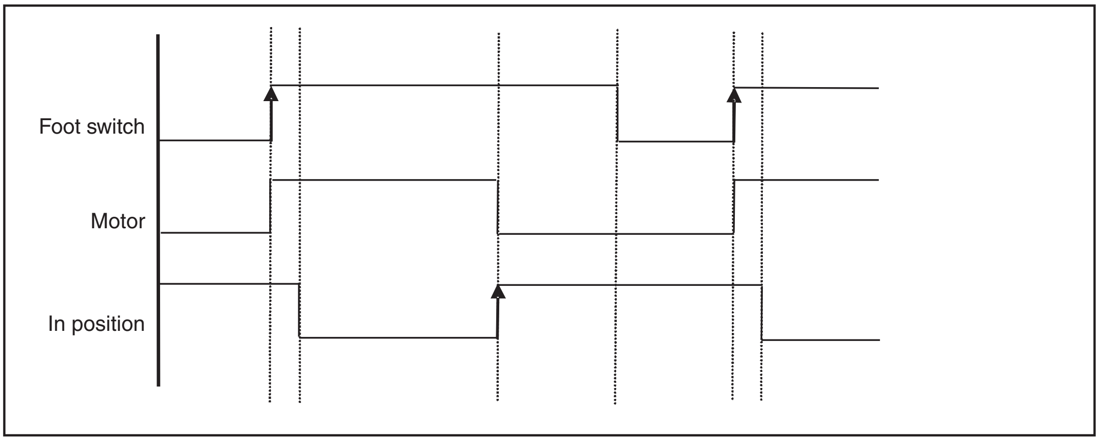
The most appropriate command to use in this application is the MELSEC 'PLS' (Rising edge Pulse). It has been adopted here instead of the IEC instruction R_TRIG (Rising edge Trigger) instruction, which would also be suitable.

The following diagram illustrates the order of sequencing of the carousel control. Note that the rising edge of the foot switch triggers the motor ON, irrespective of the "In Position" sensor being ON.

When the table begins rotating, the "In position" sensor turns OFF a little later. The motor continues to drive the carousel conveyer until the rising edge of the "In Position" sensor is detected; this turns the motor OFF. Note that the foot switch continues to be held on.

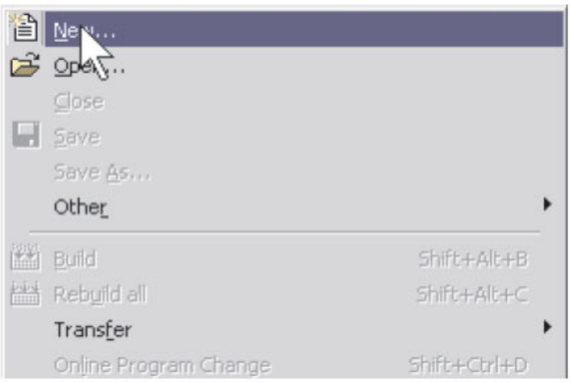
The Motor can only start rotation when the foot switch is released and subsequently reactivated. Hence the motor starts again on the rising edge of the Foot Switch being operated.

Timing Diagram of Carousel Control Logic:

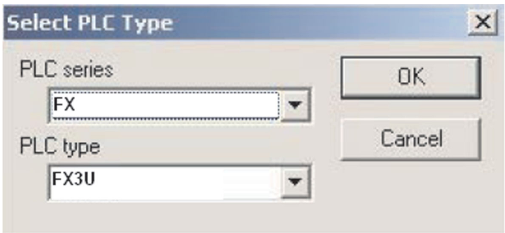


4.2.2 Creating a New Project

① From the ***Project*** menu, select ***New***.



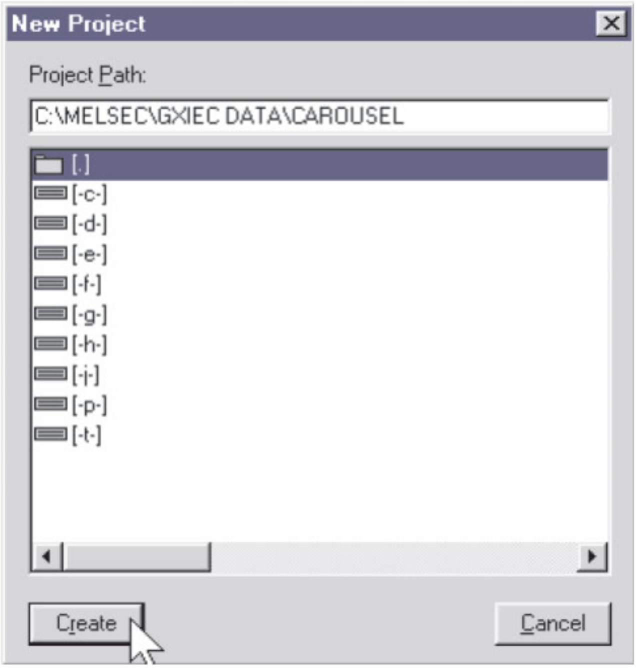
② Choose the appropriate ***PLC type*** from the selection:


Select PLC Type

PLC series: FX

PLC type: FX3U

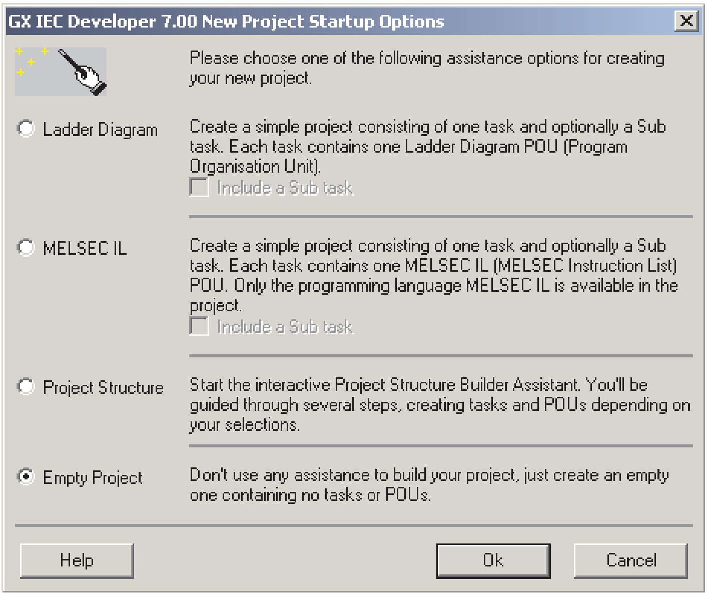
③ Provide a name for the project in the project path field. In this case use “\GXIEC DATA\CAROUSEL” and click on ***Create*** – as in the following illustration:


New Project

Project Path: C:\MELSEC\GXIEC DATA\CAROUSEL

The Wizard

The Project Startup Wizard will be displayed:



GX IEC Developer 7.00 New Project Startup Options

Please choose one of the following assistance options for creating your new project.

○ Ladder Diagram — Create a simple project consisting of one task and optionally a Sub task. Each task contains one Ladder Diagram POU (Program Organisation Unit).
□ Include a Sub task

○ MELSEC IL — Create a simple project consisting of one task and optionally a Sub task. Each task contains one MELSEC IL (MELSEC Instruction List) POU. Only the programming language MELSEC IL is available in the project.
□ Include a Sub task

○ Project Structure — Start the interactive Project Structure Builder Assistant. You'll be guided through several steps, creating tasks and POUs depending on your selections.

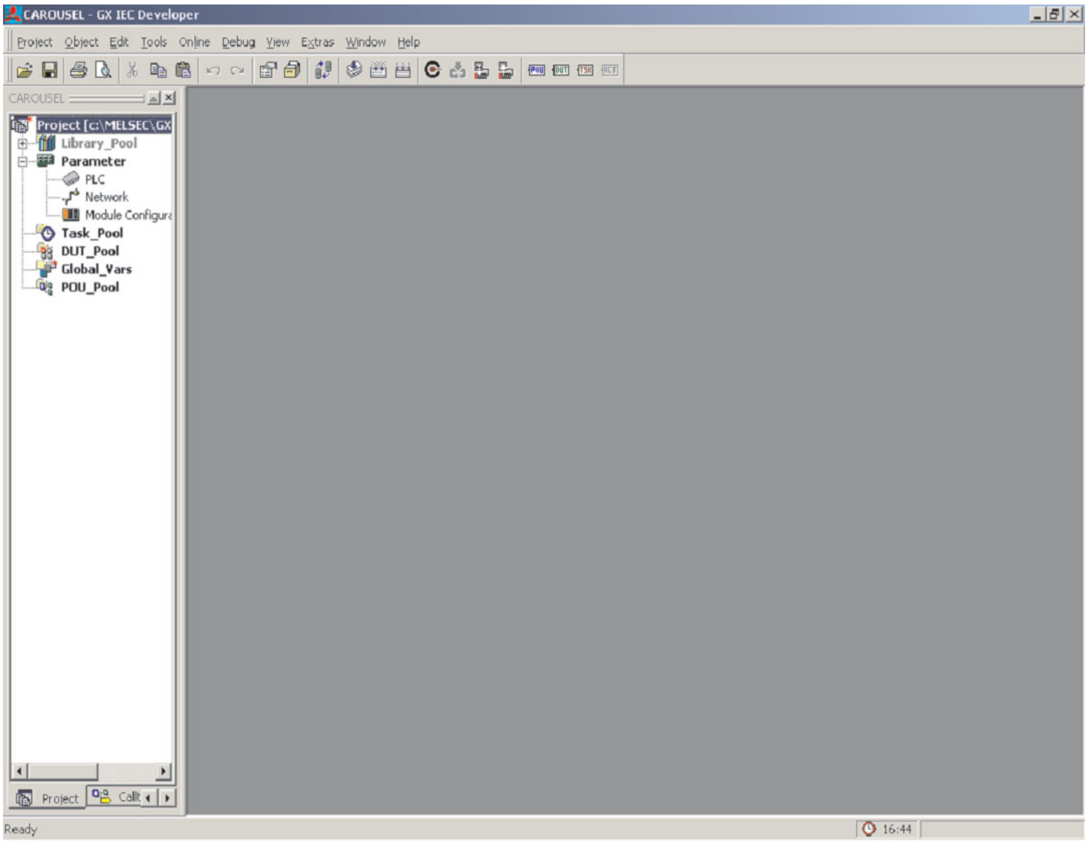
● Empty Project — Don't use any assistance to build your project, just create an empty one containing no tasks or POUs.

The Wizard provides a quick way to begin projects. It will thus create the basic starting structures for simple projects.

Select the Option, ***Empty Project*** and click ***OK***.

This effectively inhibits the Wizard from creating any project elements. Of course, the Wizard may be used if desired, but in order to fully explore the primary functions of GX IEC Developer, for training purposes we will use manual operations to create a program.

The project display screen is shown as illustrated below:

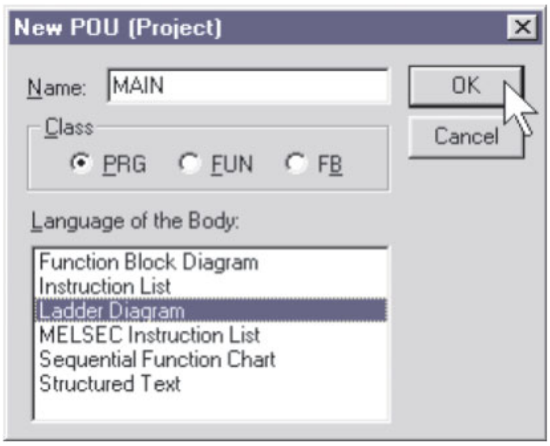


This is the primary display of the project.

The project navigation window on the left hand side of the screen enables the user to rapidly access any portion of the project by double clicking on the selection.

4.2.3 Creating a new “POU”

① Click on the “New POU” button (or “Right Click” on POU Pool) on the tool bar. The new POU specifications are to be entered as follows:



New POU (Project)

Name: MAIN

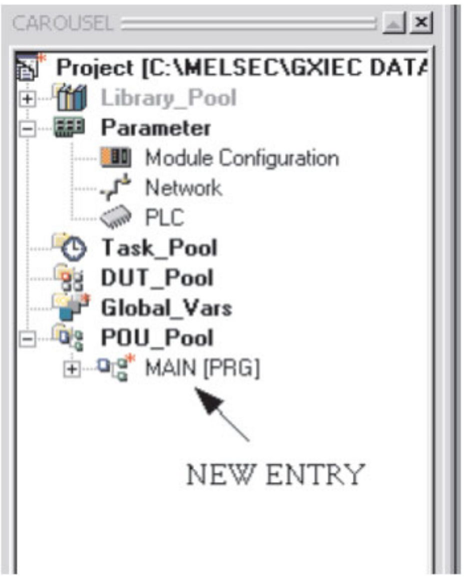
Class: ● PRG ○ FUN ○ FB

Language of the Body:

- Function Block Diagram
- Instruction List
- Ladder Diagram
- MELSEC Instruction List
- Sequential Function Chart
- Structured Text

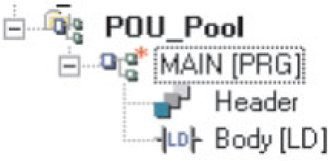
The name of the POU will be ‘MAIN’ and it should be specified as a **Ladder Diagram** of type ***PRG*** (Program).

② Click ***OK*** and note the addition to the POU Pool in the ‘Project navigation window’:



```
CAROUSEL
Project [C:\MELSEC\GXIEC DAT/
  Library_Pool
  Parameter
    Module Configuration
    Network
    PLC
  Task_Pool
  DUT_Pool
  Global_Vars
  POU_Pool
    MAIN [PRG]
```

③ Double click on ***MAIN*** program icon or click the symbol on the POU Pool in order to expand the directory branch and display the Header and Body entries:

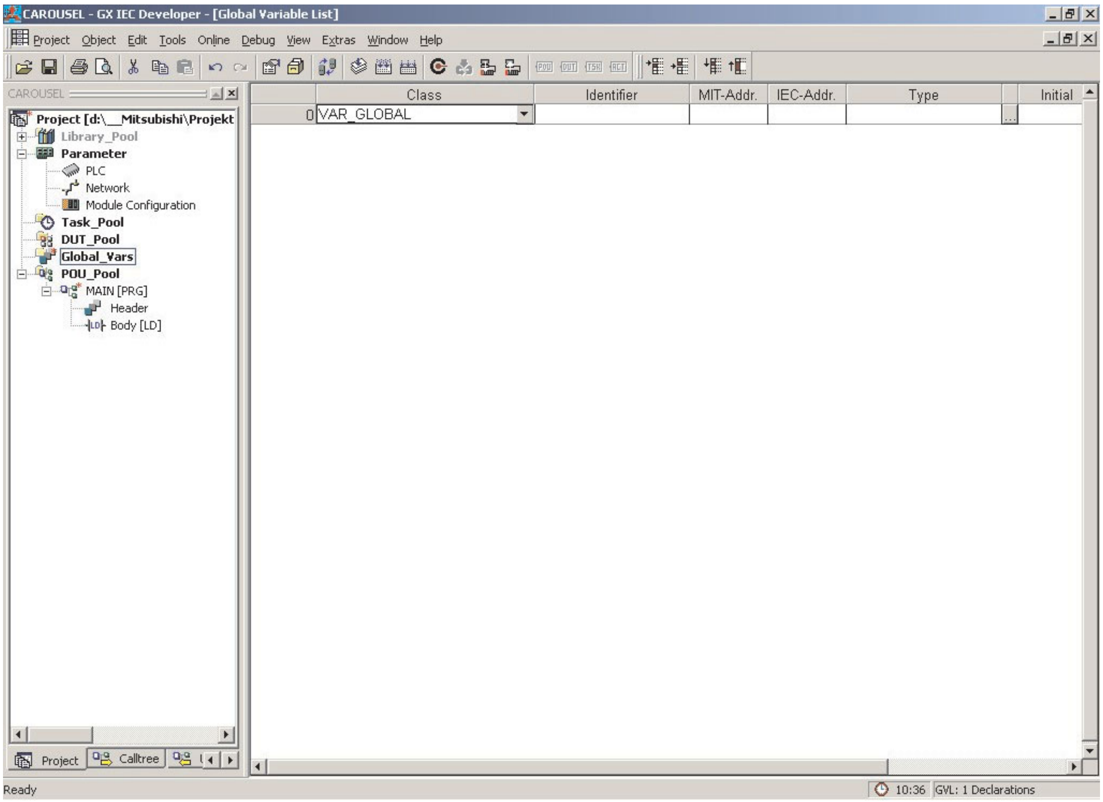


```
POU_Pool
  MAIN [PRG]
    Header
    Body [LD]
```

4.2.4 Assigning the Global Variables

Before any program code can be created, it is necessary to specify and assign all pre-allocated physical PLC inputs and outputs including any shared variables that are to be used in the project.

Double Click the mouse pointer on ***Global_Vars*** to open the Editor for the Global Variables. This is called the Global Variable List - GVL.



	Class	Identifier	MIT-Addr.	IEC-Addr.	Type	Initial
0	VAR_GLOBAL					

Global Variables are the link to the physical PLC devices.

As discussed previously, if IEC conventions are to be applied, then symbolic identifiers (names) must be used instead of discreet addresses in our program. These addresses must therefore be declared in the Global Variable List (GVL). The identifier must be filled in, using its' PLC address (either using Mitsubishi or IEC notation) and its' type, for example; whether it is a 'bit' or 'word' device. Once completed, this list can be used by all of the POU's that will be created.

Declaring Variables

As can be seen from the GVL field list, each variable has a set of elements as follows:

- Class

  The class keyboard assigns the variable a specific property that defines how it is to be used in the project

- Identifier

  Each variable is given a symbolic address, i.e. a name. This is referred to as the identifier. It consists of a string of alphanumeric characters and ‘underscore’ characters. The identifier must always begin with a letter or an underscore character. Spaces and mathematical operator characters (e.g. +,-,*) are not permitted.

- MIT-Addr

  This is the absolute address referenced in the PLC.

- IEC-Addr

  The IEC syntax of the address.

- Type

  Referrers to the data type, i.e. BOOL, INT, REAL, WORD etc.

- Initial

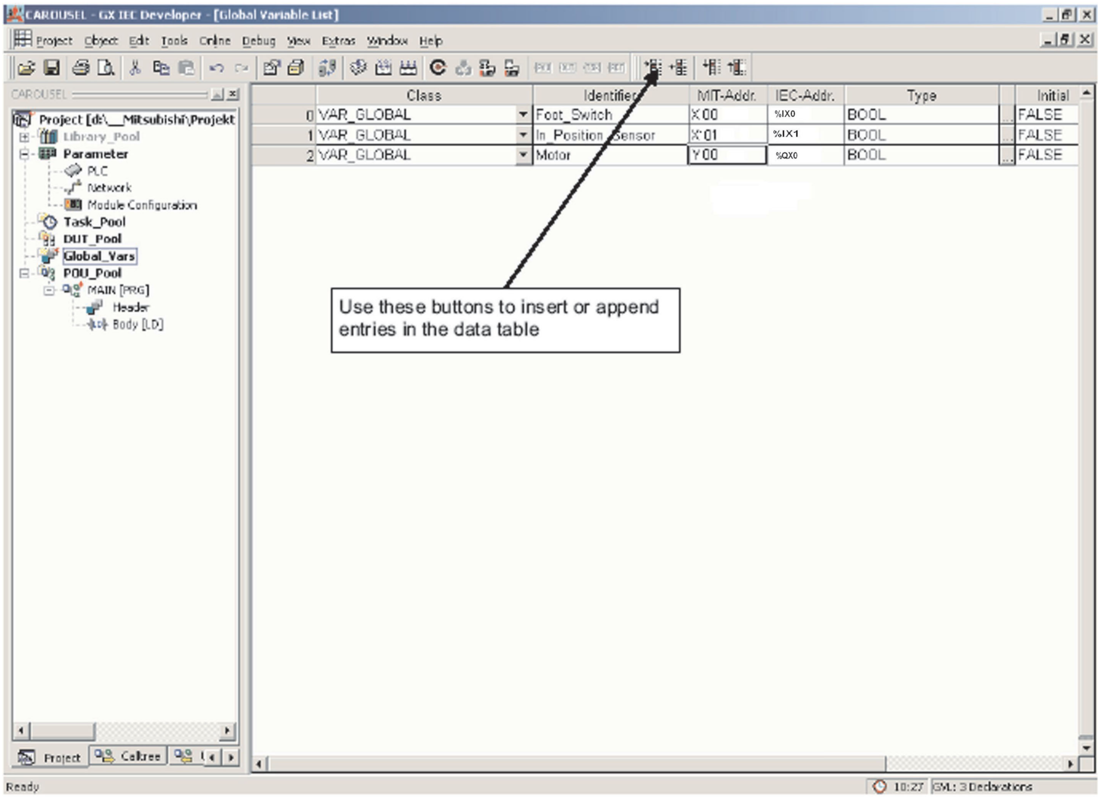
  The initial values are set automatically by the system and cannot be changed by the user.

- Comment

  Comments up to 64 characters may be added for each variable

If symbolic identifiers are not to be used in the program but only Mitsubishi addresses, then there is no need to fill out the Global Variable List (GVL). However the program will no longer be truly IEC61131-3 compliant.

Fill out the table as shown in the following illustration. The variable “Type Selection” is automatically recognised and placed by GX IEC Developer upon entry of the ‘Address’ but can be input manually or modified by clicking on the type select arrow in the ***Type*** field area. When the Mitsubishi address is entered, the system automatically converts and enters the IEC equivalent.



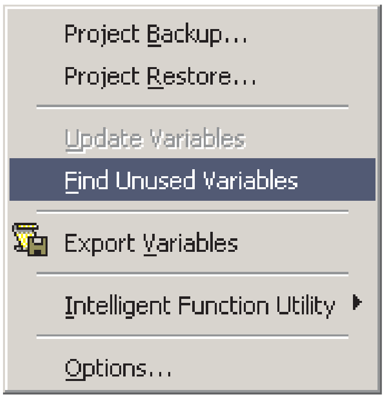
	Class	Identifier	MIT-Addr.	IEC-Addr.	Type	Initial
0	VAR_GLOBAL	Foot_Switch	X00	%IX0	BOOL	FALSE
1	VAR_GLOBAL	In_Position_Sensor	X01	%IX1	BOOL	FALSE
2	VAR_GLOBAL	Motor	Y00	%QX0	BOOL	FALSE

Use these buttons to insert or append entries in the data table

These are the Global Variables specified for the project.

Find unused variables

By using the function ***Extra*** -> ***Find Unused Variables*** you can find and delete all unused global and local variables that are declared but not used in a project. Unused global and local variables will be detected in the whole project, excluding the user libraries.



NOTE

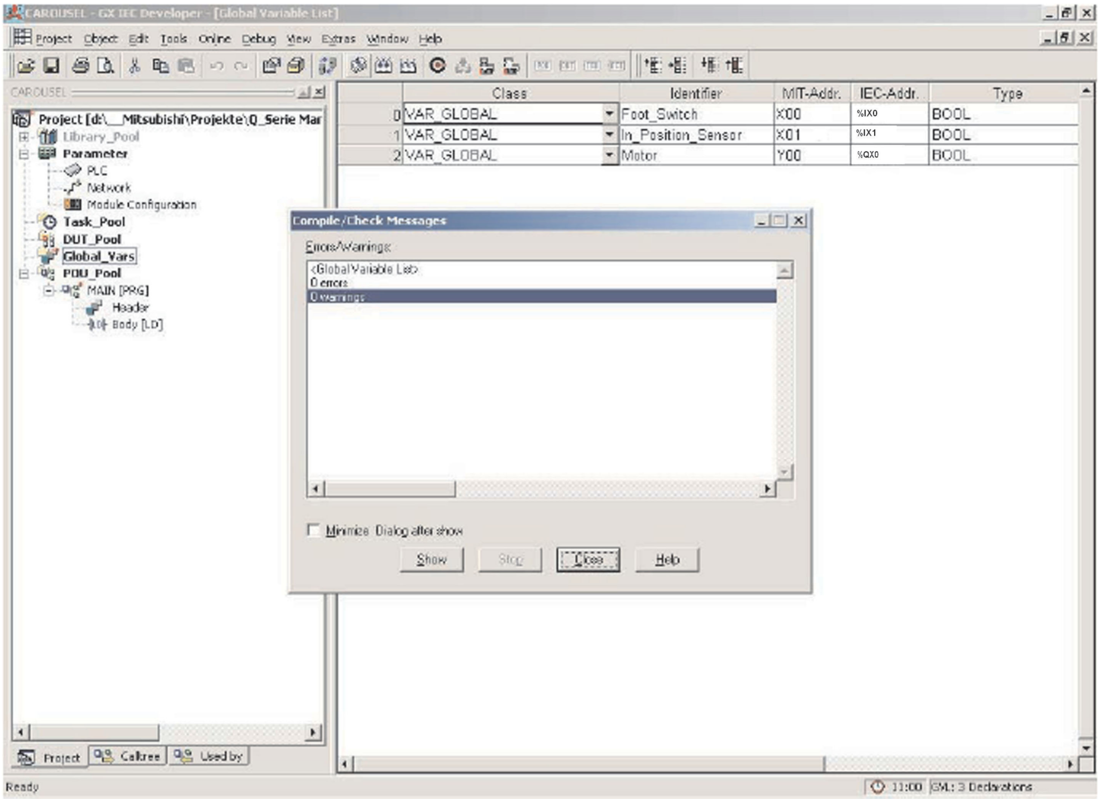
Finding unused variables can only be performed if the project has been built and was not changed since then. Otherwise a warning message will be displayed.

NOTES

The Global Variable List incorporates an “Increment new declarations” feature. If the GVL contains entries i.e. for a number of valves, ‘Valve_1’ to ‘Valve_n’ then if the first entry is made for Valve_1 and new rows are declared either via the tool bar icons or “Shift+Enter” then both the identifier and address fields are incremented. This feature is enabled by default. If this is not required it can disabled via the ***Extras*** menu (***Extras\Options\Editing***), to be described later. All or selected POU’s can be selected and all or selected variables can be deleted. When invoked, all unused Global Variables in POU’s are deleted. This feature will be explored later when appropriate.

For all FX2N, FX3G, FX3U, FX3UC, System Q and AnA(S) type CPU’s IEC Type REAL (Floating Point) values are fully supported.

When the data entry in the GVL has been completed, click the ‘Check’ button as shown:



	Class	Identifier	MIT-Addr.	IEC-Addr.	Type
0	VAR_GLOBAL	Foot_Switch	X00	%IX0	BOOL
1	VAR_GLOBAL	In_Position_Sensor	X01	%IX1	BOOL
2	VAR_GLOBAL	Motor	Y00	%QX0	BOOL

Compile/Check Messages

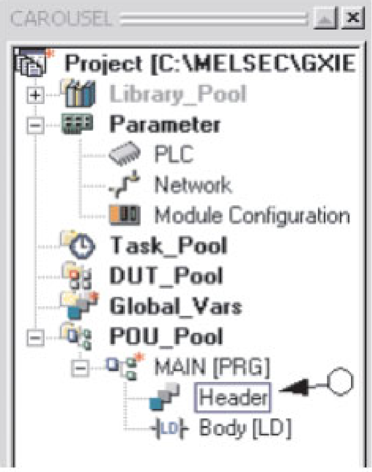
Errors/Warnings:

```
<Global Variable List>
0 errors
0 warnings
```

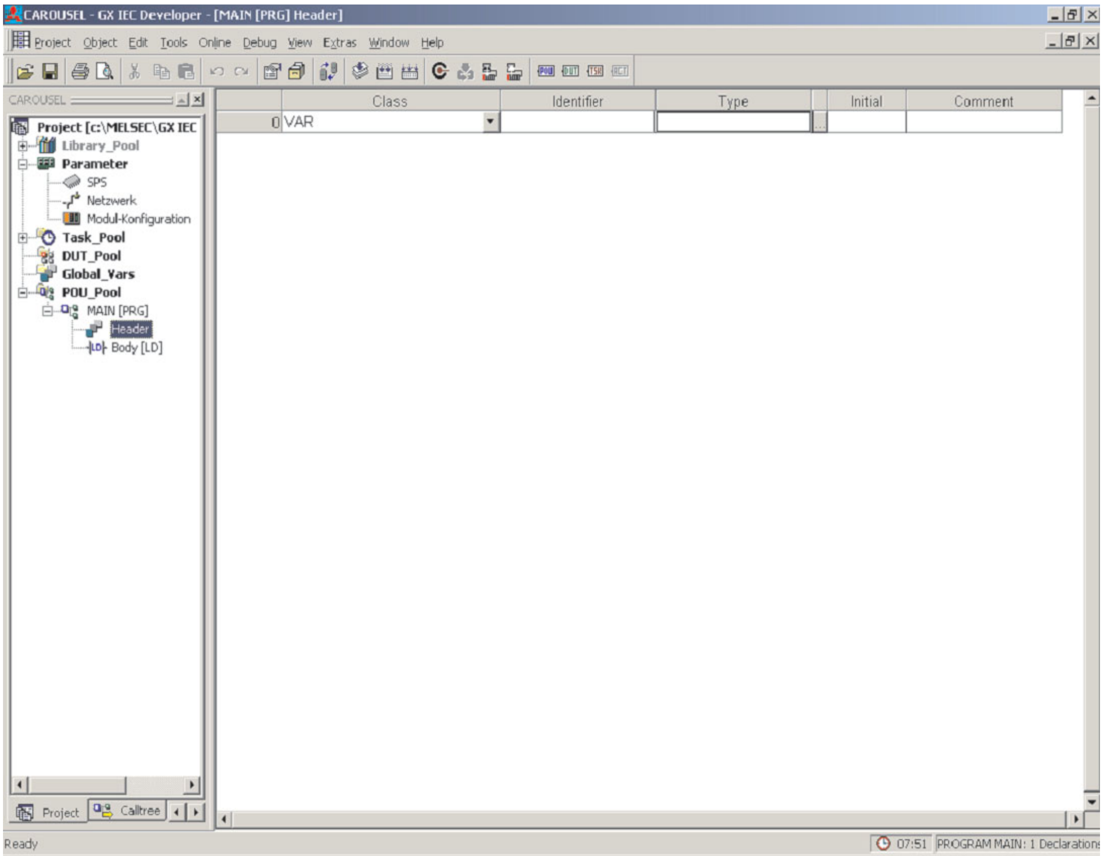
□ Minimize Dialog after show

Opening the POU Header

From the Project Navigation window, double click on the **Header** on the POU ***MAIN***.



The following screen will be displayed:

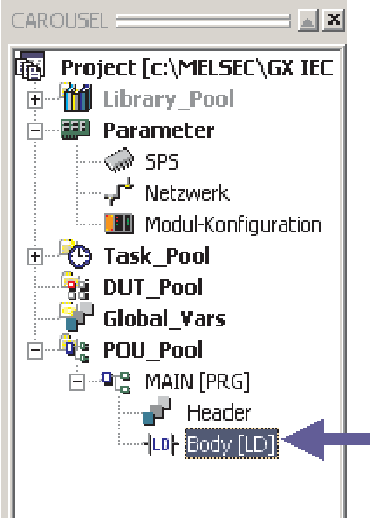

CAROUSEL - GX IEC Developer - [MAIN [PRG] Header]
Project Object Edit Tools Online Debug View Extras Window Help
CAROUSEL
Project [c:\MELSEC\GX IEC
Library_Pool
Parameter
SPS
Netzwerk
Modul-Konfiguration
Task_Pool
DUT_Pool
Global_Vars
POU_Pool
MAIN [PRG]
Header
Body [LD]
Project
Calltree
Ready
07:51
PROGRAM MAIN: 1 Declarations

	Class	Identifier	Type	Initial	Comment
0	VAR				

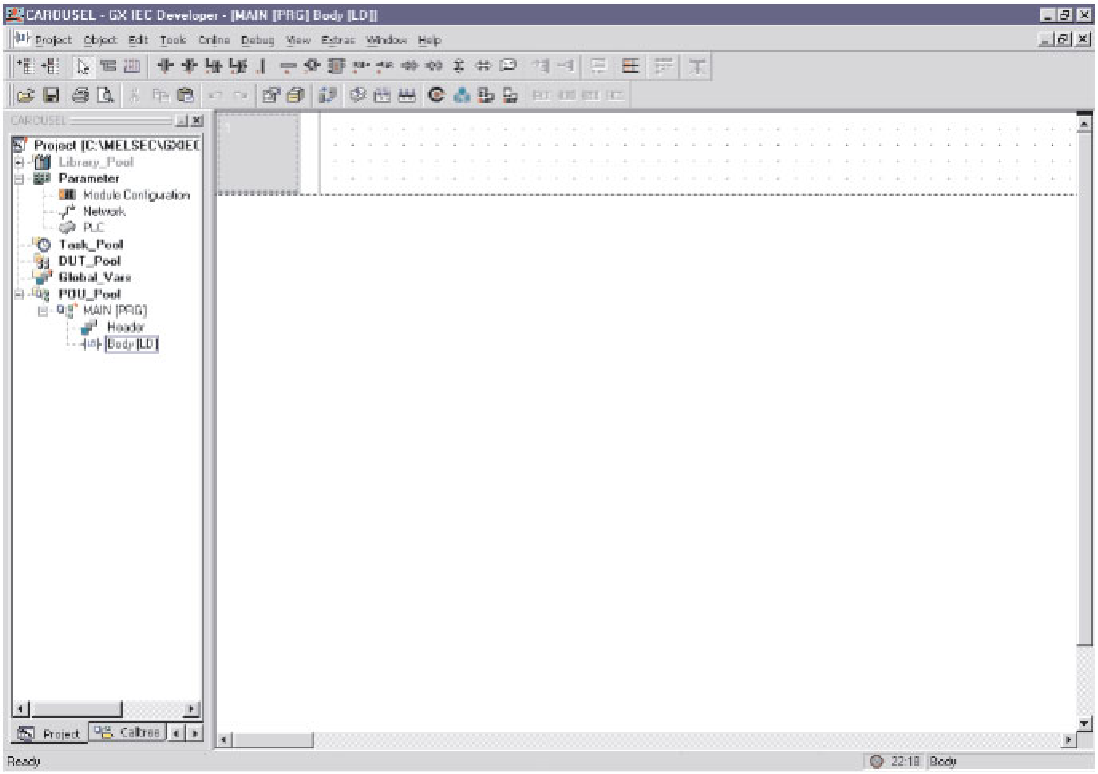
Close this POU Header display.

4.2.5 Programming the POU Body

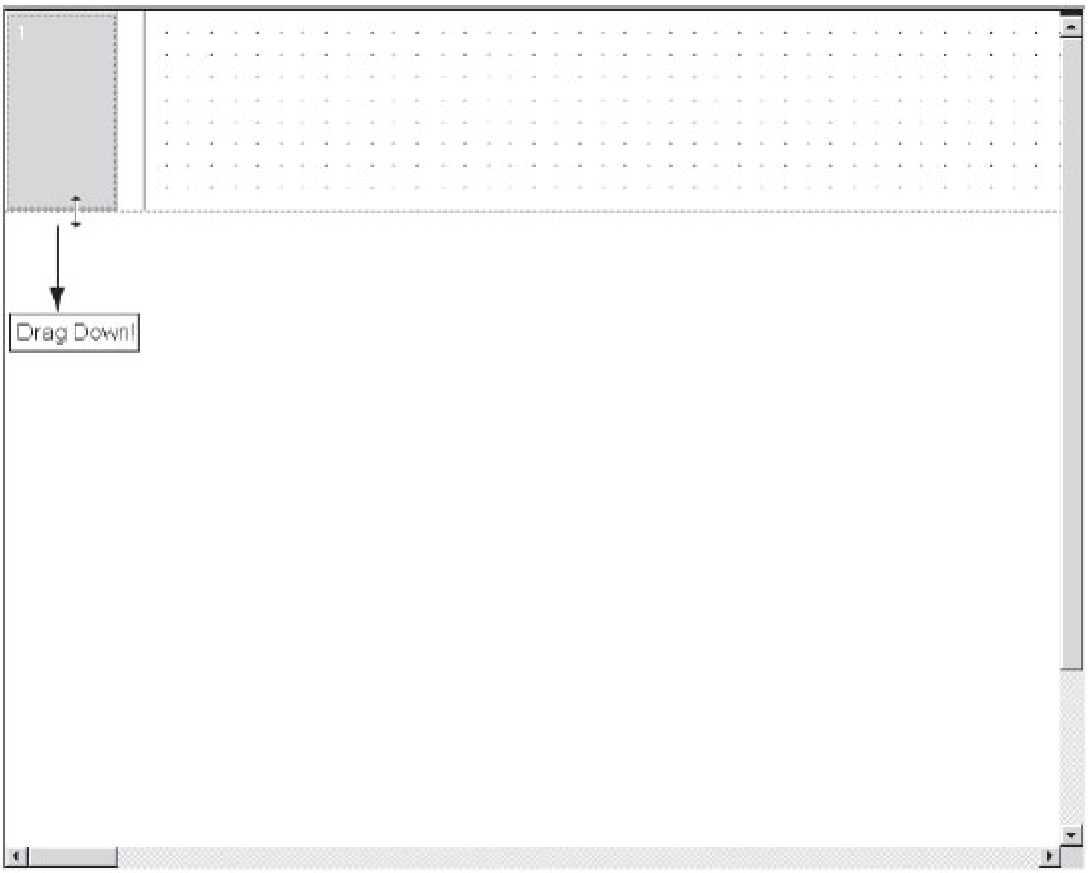
① To open the Ladder diagram editor, double click on the Body selection under the POU pool in the project navigation window:



The following window is displayed:

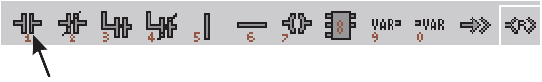


② With the pointer over the window boundary, click and drag downwards to increase the vertical size of the network:

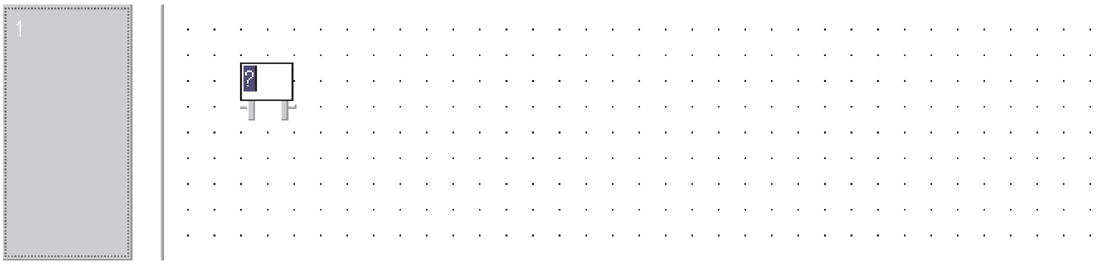


Using the Toolbar Ladder Symbol Selection

③ With the editor in “Selection Mode”, select the ‘Normally Open’ contact from the toolbar:

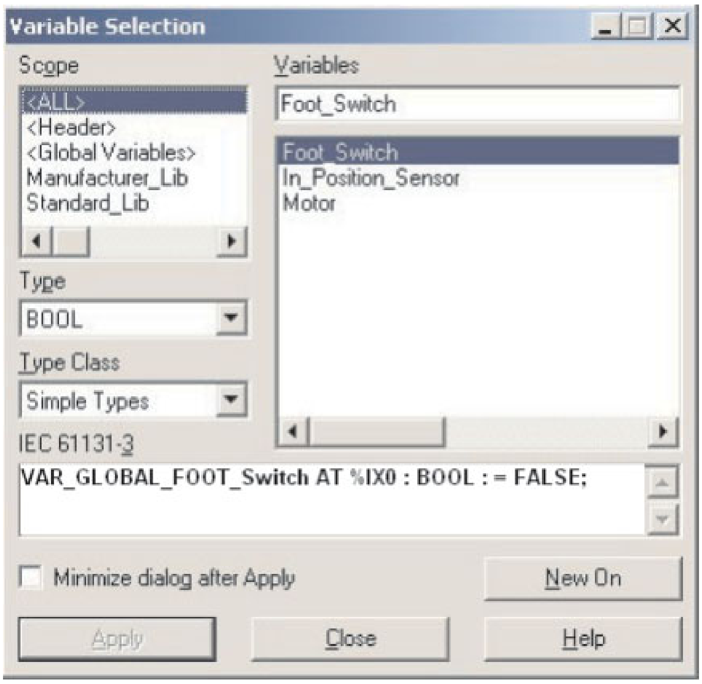


④ Move the mouse pointer over the work area and click to fix the drop position on the window:



Selecting variables from the POU Header

① Press the “F2” button on the keyboard or click on the button on the tool bar to call up the variables selection window and the display will be as shown below:



Variables: Foot_Switch

Type: BOOL

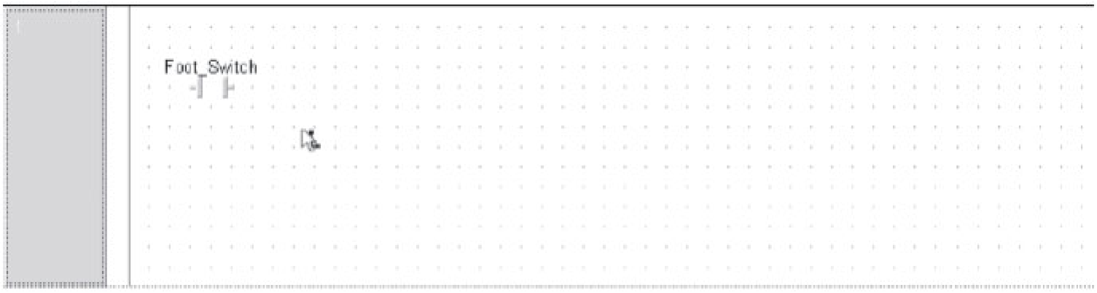
Type Class: Simple Types

IEC 61131-3

```
VAR_GLOBAL_FOOT_Switch AT %IX0 : BOOL : = FALSE;
```

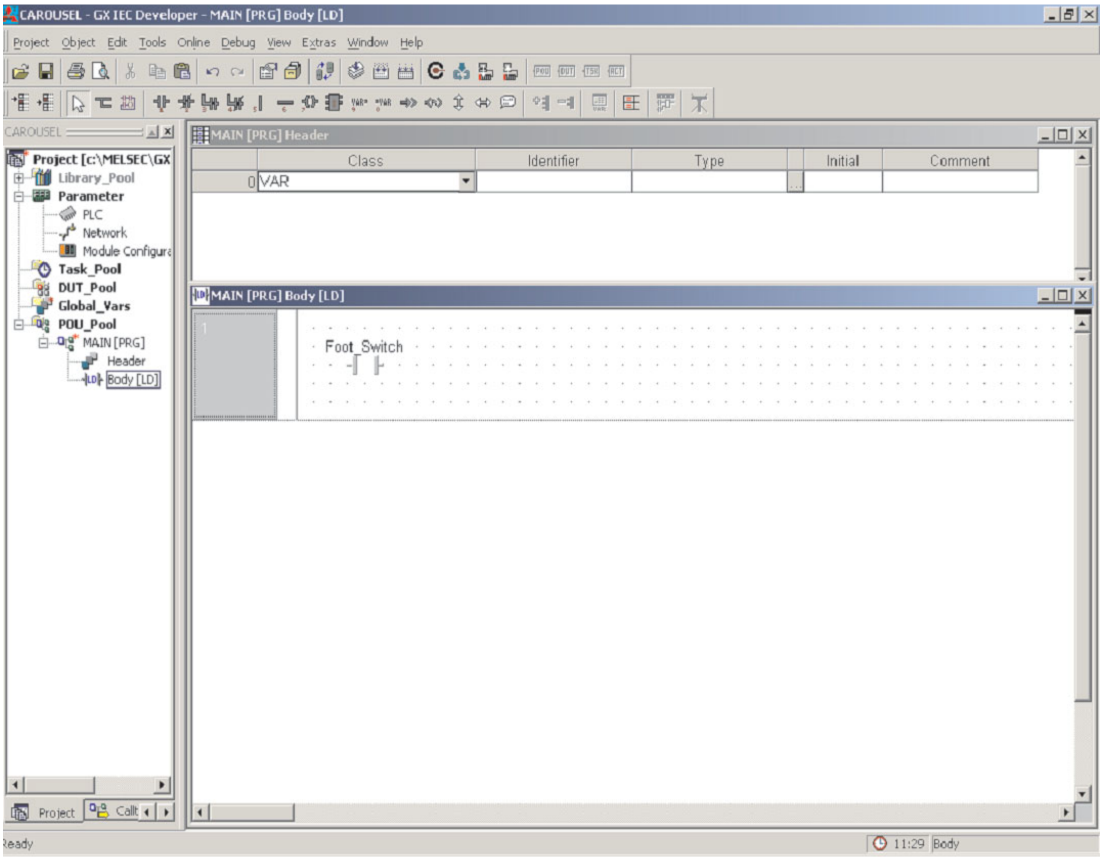
Note that the current ‘Header’ should be selected under the ***Scope*** dialogue area.

② Click “Foot_Switch” to highlight that variable and click the ***Apply*** button. Then close the Variable Selection box.



Alternative Variable Specification Method: Editing in Split Screen

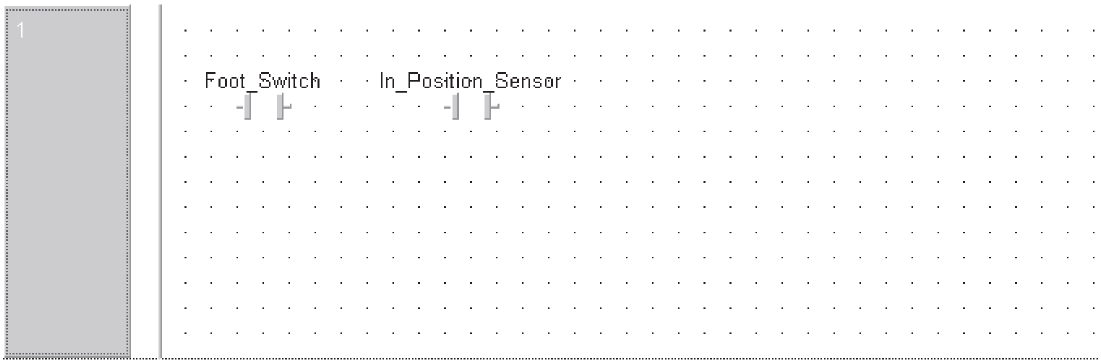
Split screen viewing of POU Ladder diagram and Header is possible by opening both the header and the ladder and selecting “Tile Horizontally.”



	Class	Identifier	Type	Initial	Comment
0	VAR				

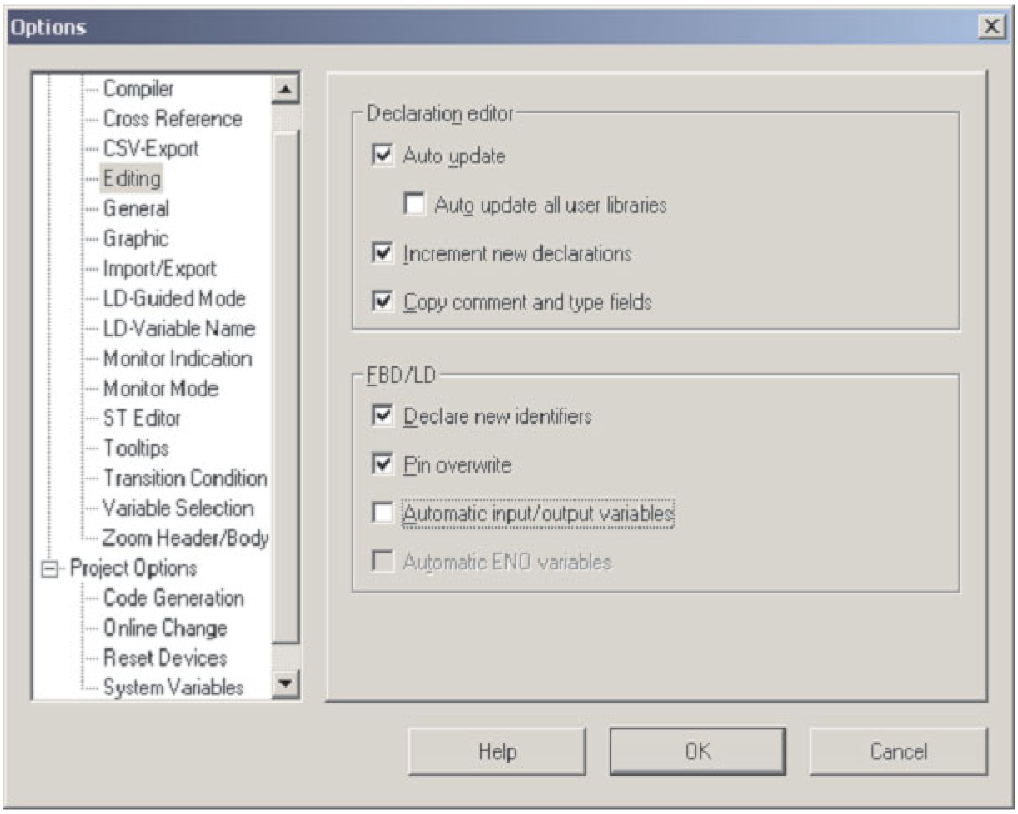
Continue editing Project ‘Carousel’

Enter the normally open contact of the “In_Position_Sensor" in the position shown on the current screen in the same manner, as shown below:



Entering a Function Block command into the Ladder program

Before continuing, it is recommended for the remainder of this course, that the ***Automatic input/output variables*** facility be "**Disabled**" by de-selecting this option. This facility is found under the ***Extras*** menu using the ***Options*** selection and selecting ***Editing***, as shown below:



Options

- Compiler
- Cross Reference
- CSV-Export
- Editing
- General
- Graphic
- Import/Export
- LD-Guided Mode
- LD-Variable Name
- Monitor Indication
- Monitor Mode
- ST Editor
- Tooltips
- Transition Condition
- Variable Selection
- Zoom Header/Body
- Project Options
  - Code Generation
  - Online Change
  - Reset Devices
  - System Variables

Declaration editor

☑ Auto update

□ Auto update all user libraries

☑ Increment new declarations

☑ Copy comment and type fields

FBD/LD

☑ Declare new identifiers

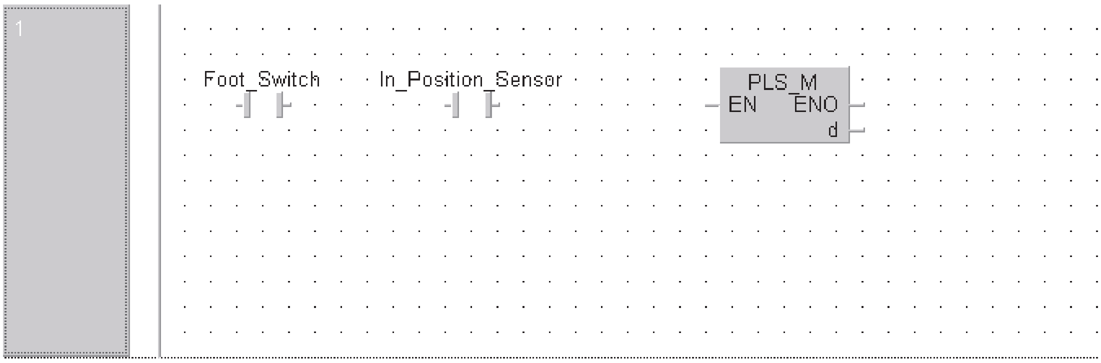
☑ Pin overwrite

□ Automatic input/output variables

□ Automatic ENO variables

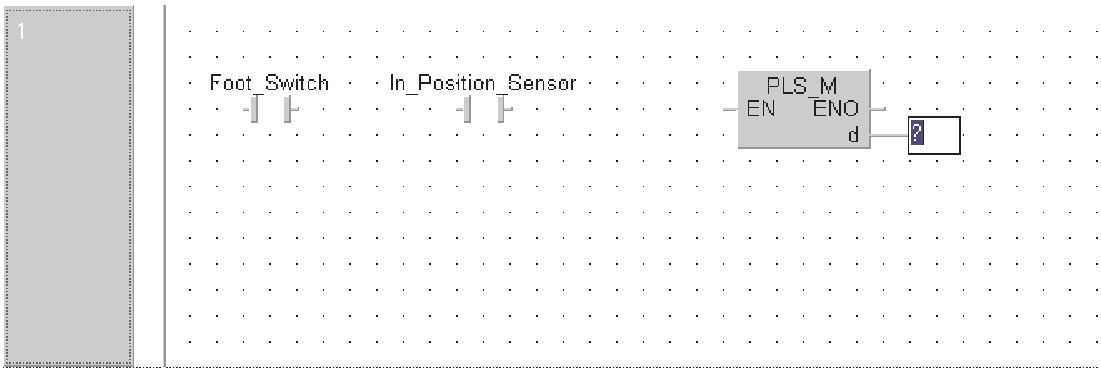
The MELSEC Function Block command, 'PLS_M' will be added to the program as the output function.

① Click on the Function / Function block selection button on the tool bar. On the ***Operator type*** click ***Functions*** and type "PLS_M" into the ***Operators*** prompt box thus:



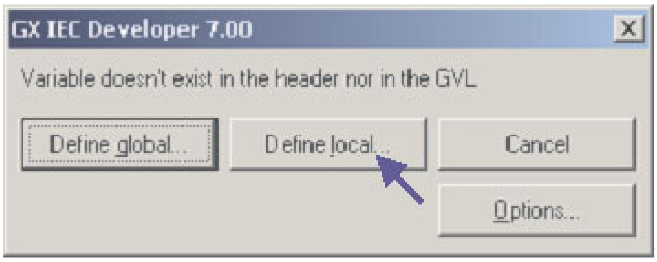
Assigning a Variable to an Instruction

② Click on the output variable prompt from the toolbar. Click on the ‘d’ destination, output function from the PLS_M to drop the variable prompt field.



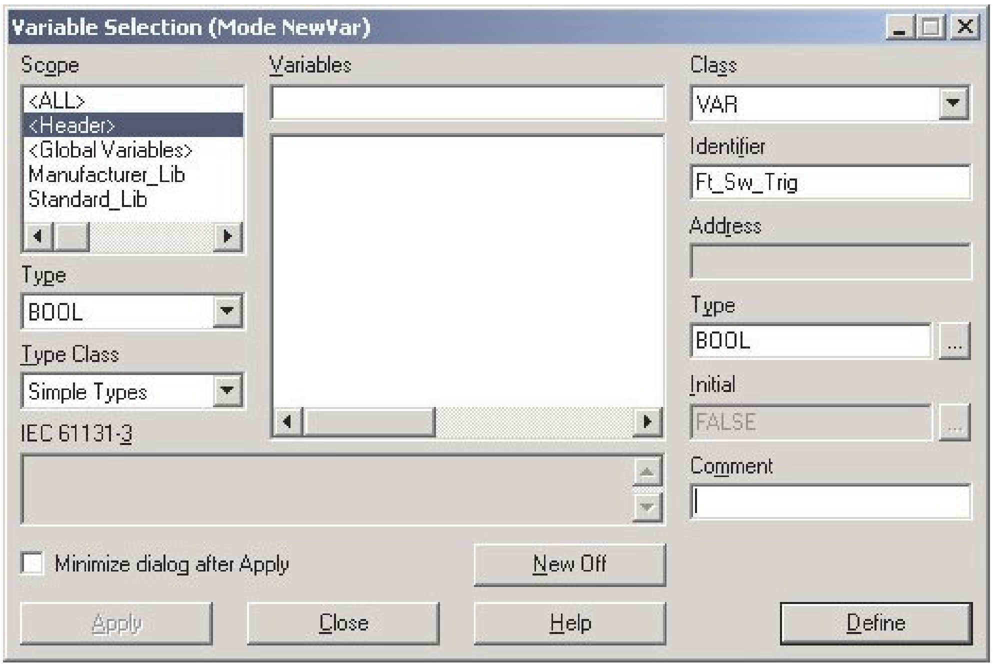
③ Enter the variable name Ft_Sw_Trig into the empty ‘?’ box.

The following prompt is displayed if the variable does not exist in the Local Variable List ‘LVL’ (Local Header) or the Global Variable List ‘GVL’:


GX IEC Developer 7.00

Variable doesn't exist in the header nor in the GVL

④ Click on ***Define Local*** to define a new Local Variable ‘LVL’. The ***Variable Selection*** window is displayed, prompting a new variable to be defined:


Variable Selection (Mode NewVar)

Scope
<ALL>
<Header>
<Global Variables>
Manufacturer_Lib
Standard_Lib

Type: BOOL

Type Class: Simple Types

IEC 61131-3

Class: VAR

Identifier: Ft_Sw_Trig

Address:

Type: BOOL

Initial: FALSE

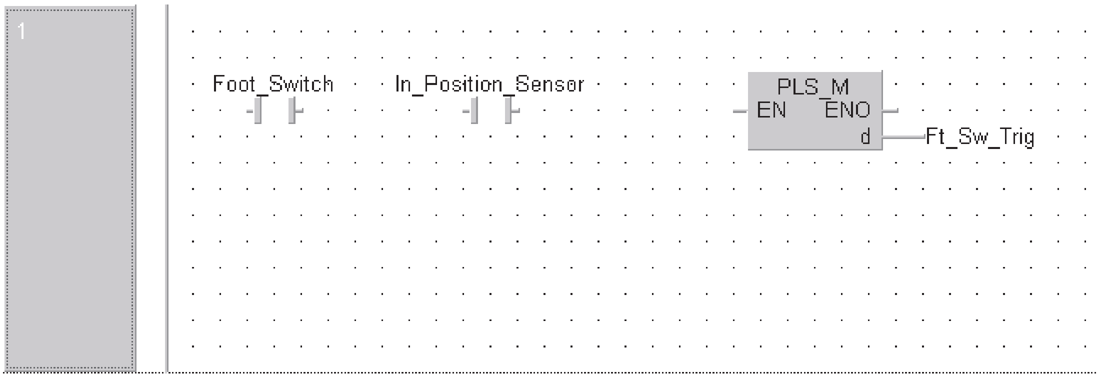
Comment:

□ Minimize dialog after Apply

⑤ Click ***Define*** to enter the new variable into the LVL (Local Header).

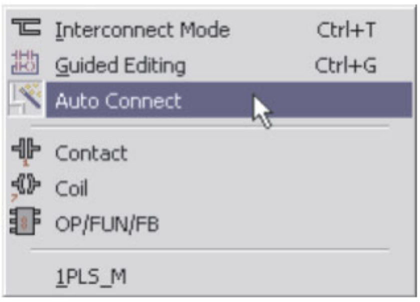
NOTE To confirm the above operation, check the local header!!

The display should be as follows:

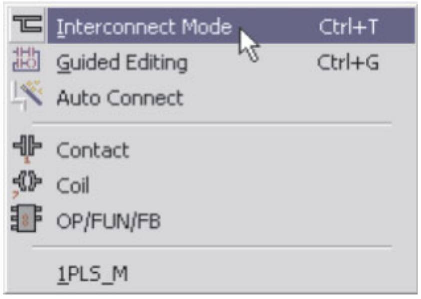


Finally, the ladder network must be finalised by connecting up the elements as follows.

⑥ Right click the mouse anywhere in the edit window area and de-select the ***Auto connect*** function.

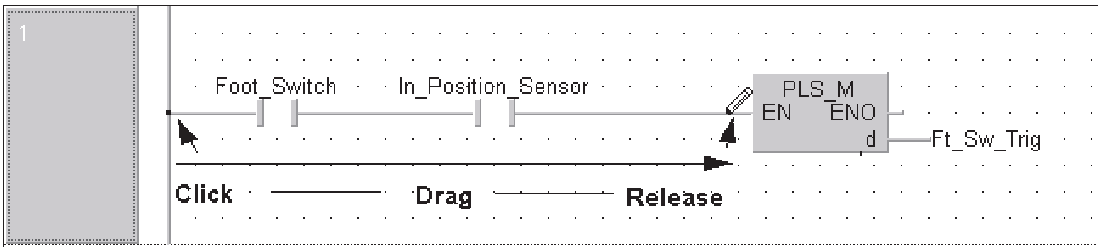


⑦ In the same manner, click to select ***Interconnect Mode***.



Note that the Pointer now changes to a small pencil icon.

⑧ On the Ladder diagram click on the left point on the ladder diagram and “Click – Drag” across the diagram and release on the ‘EN’ input on the ‘PLS_M’ function as shown below:



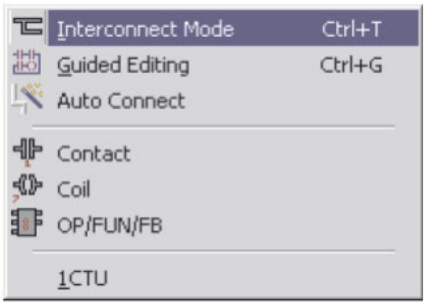
The circuit is now complete.

Changing the cursor mode

Before continuing with the worked example, it is necessary to understand the operation of the cursor control and the various edit modes that are available.

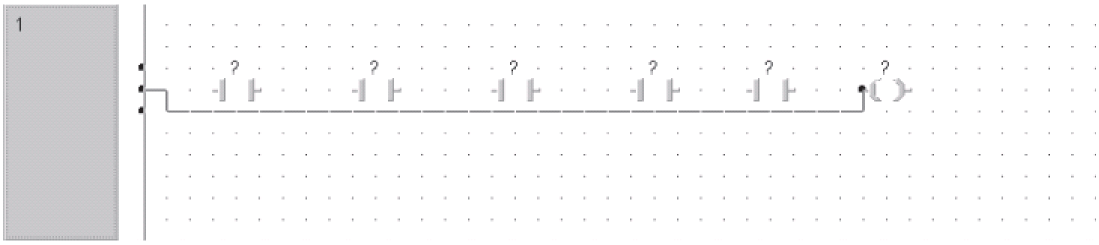
The following text is for illustration purposes only:

While in the ladder edit screen, Right clicking the mouse button pops up a small selection window as shown below. Clicking on ***Auto Connect*** toggles this feature on/off; it is also the method for switching between pen and arrow, other than via toolbar icons.

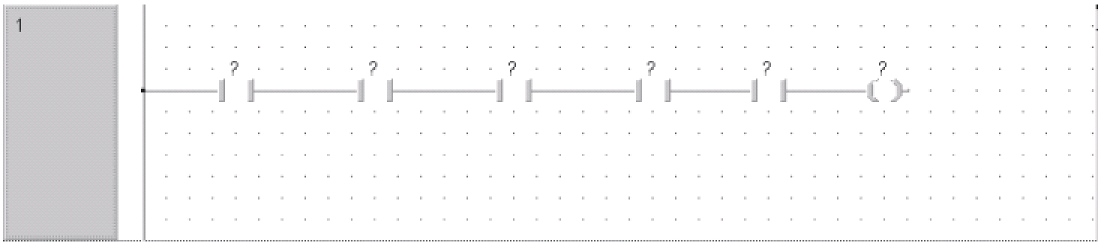


Precautions when using the Ladder Editor

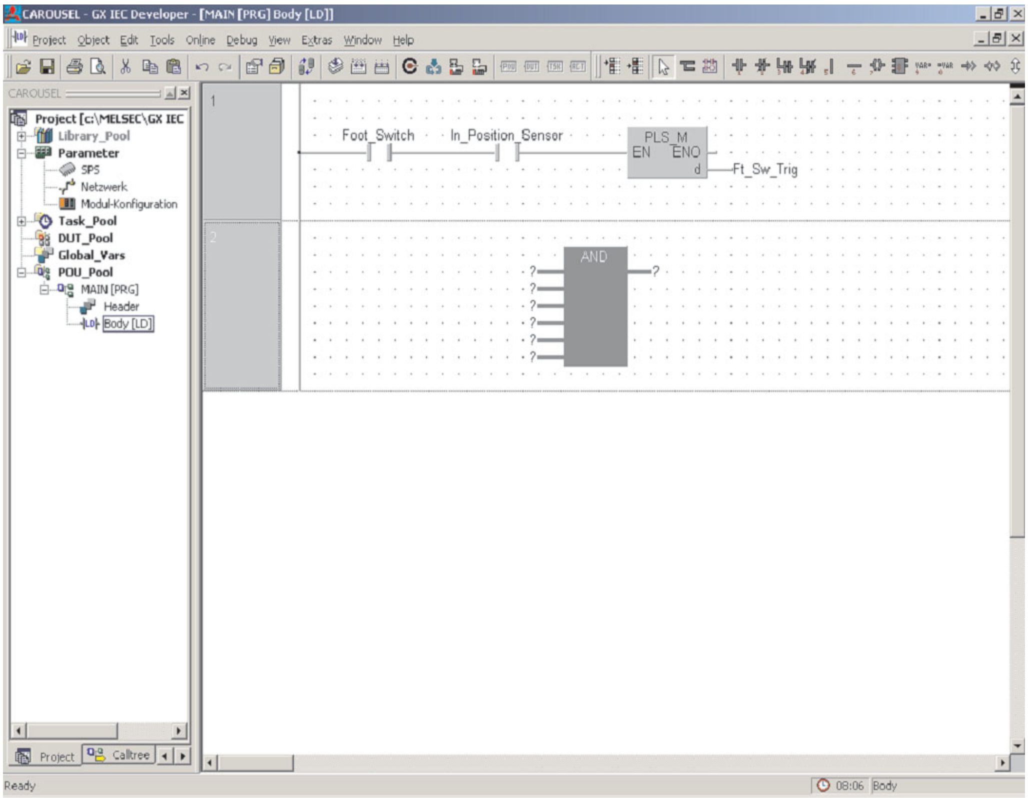
As can be seen from the screen below, because ***Auto Connect*** connects between two points, for a row of contacts the line tries to connect as shown. With ***Auto Connect*** on, the only way to connect these contacts is to connect between each individual pair:



The pen can then strike through all contacts, from the bus bar, to the coil. In the Ladder Editor the suggestion is to invoke the ***Auto Connect*** feature when dropping elements onto the POU body or connecting parallel elements. It should however be disabled when connecting a row of contacts as shown in the following screen, or inserting a contact into an existing network.

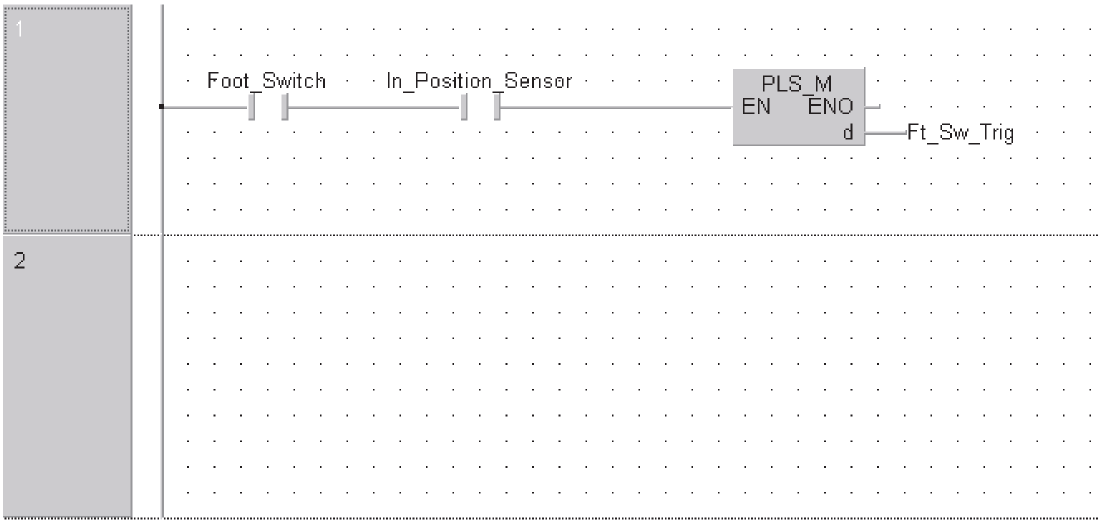


When using multi-legged or ‘pinned’ functions such as MUL, the number of input parameter legs, can be incremented/decremented by using the special toolbar, icons shown. This can also be achieved by placing the cursor at the bottom edge of the function, holding down the left hand mouse button and then dragging away as shown below:

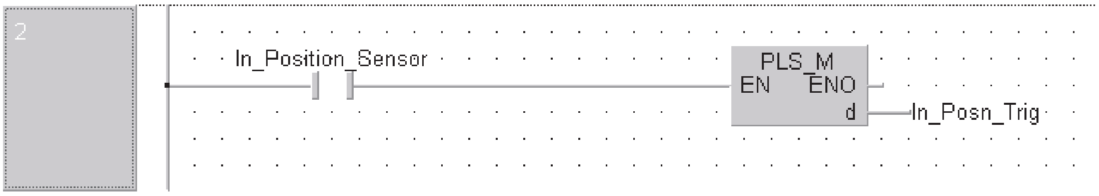


Creating a new Program Network

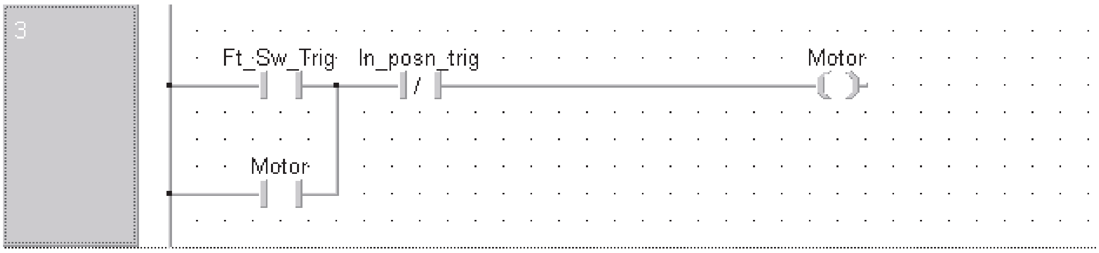
① To create a network below the current one, click the ‘insert after’ button. A blank network space will appear:



② Enter the second network in the same format as previously described with the following attributes:

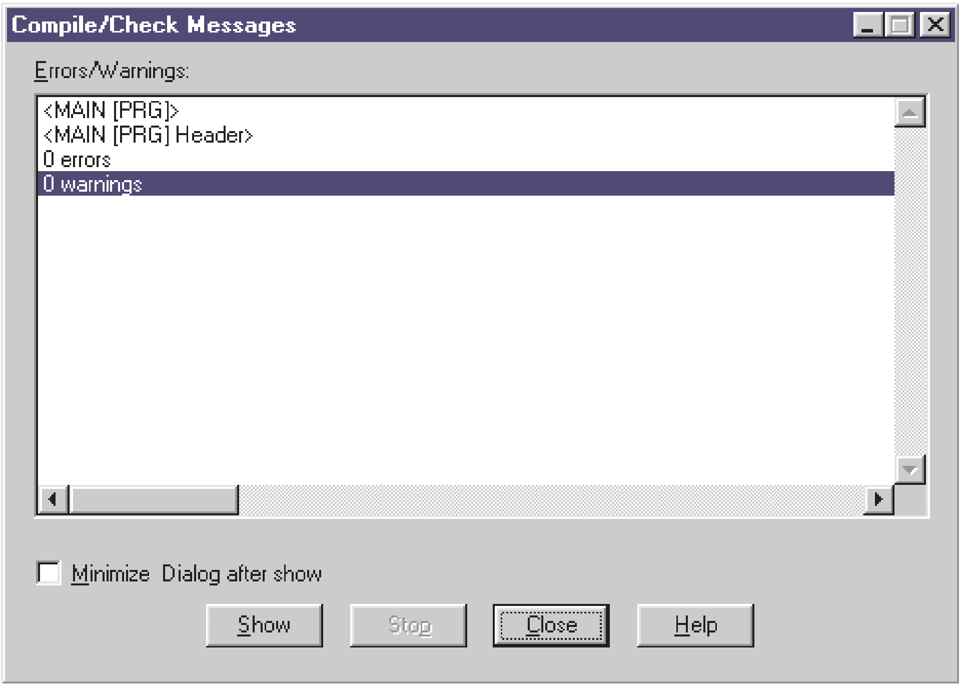


③ Finally, enter the following network as shown:



Checking the entered Program

When the three networks have been entered, complete click the Check button and if all is well, the following dialogue is displayed:



<MAIN [PRG]>
<MAIN [PRG] Header>
0 errors
0 warnings

□ Minimize Dialog after show

Adding new POU’s – Counters and Timers

Continuing with the Carousel example; Additional routines will now be added to illustrate the use of timing and counting functions.

– Counting number of operations (Product Batch Counter)

– Create an additional POU to provide a batch counting function.

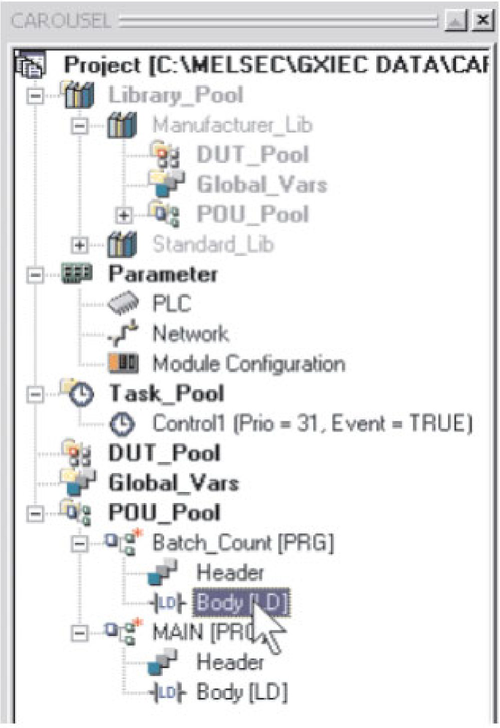
Task:

An additional POU will now be added to the project in order to count the number of times the motor is activated, i.e. product batch counter.

When ten products have been counted, the PLC will flash an output at a 1 Second ‘time-base’ until a button is operated to reset the batch counter.

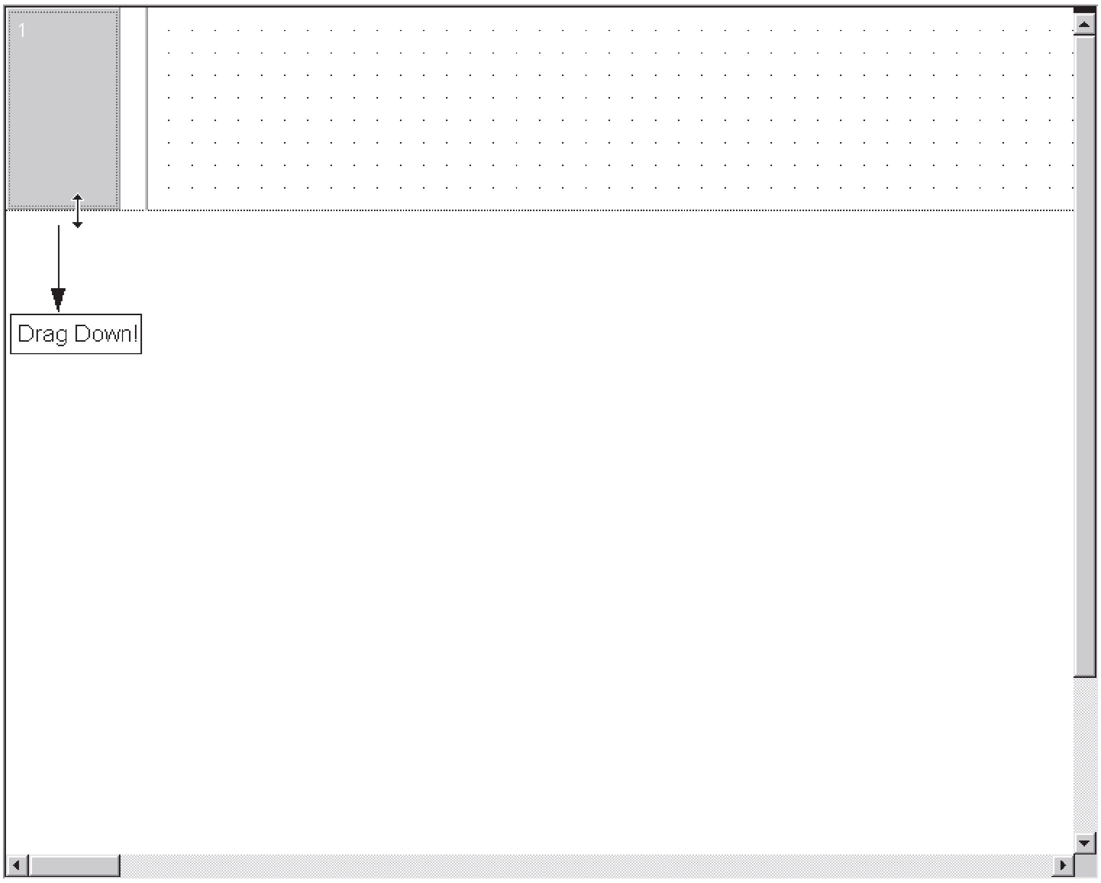
Enter the following POU ladder routine, using the ‘free-form’ editors as shown:

① Create a new POU by clicking on the POU button.



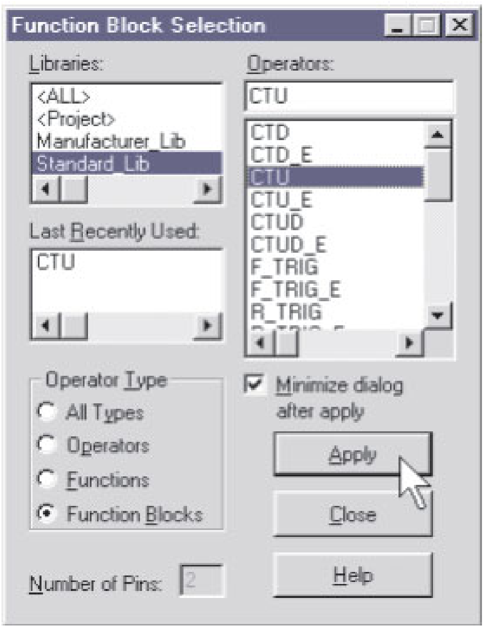
② Select the Body of the new POU by opening the newly created entry in the Project Navigation Window.

As discussed previously, the ladder network may be re-sized by moving the mouse pointer to the lower boundary of the network header and ‘click-hold’ dragging downward to increase the vertical size:



Counting function

Using the editor in “select” mode, enter the instruction CTU (Count Up) into the ladder network:


Function Block Selection

Libraries: <ALL>, <Project>, Manufacturer_Lib, Standard_Lib

Operators: CTU

CTD, CTD_E, CTU, CTU_E, CTUD, CTUD_E, F_TRIG, F_TRIG_E, R_TRIG

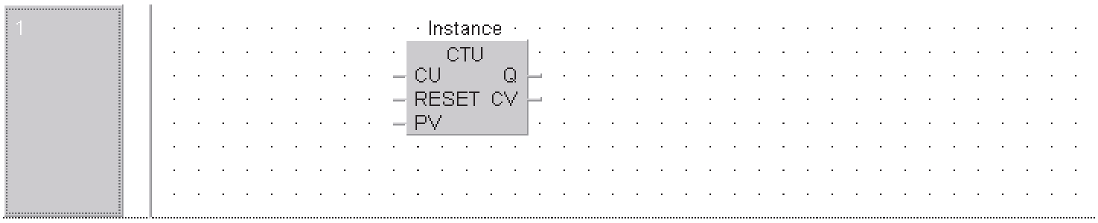
Last Recently Used: CTU

Operator Type: ○ All Types ○ Operators ○ Functions ● Function Blocks

☑ Minimize dialog after apply

Number of Pins: 2

Drop the IEC Function Block onto the empty Ladder network:

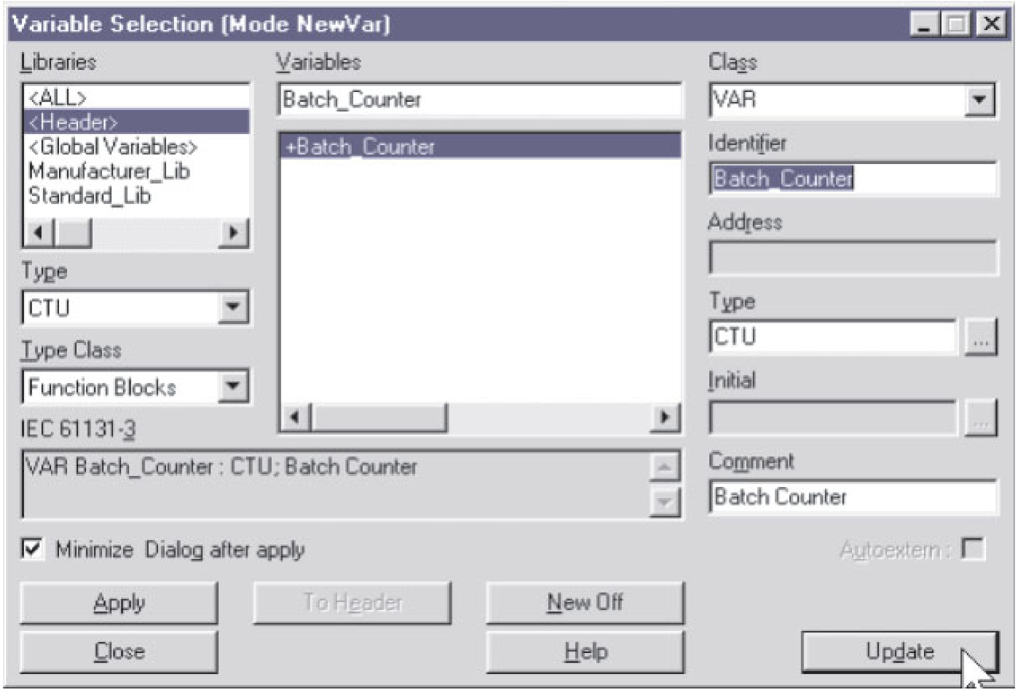


Instances of Function Blocks

Function Blocks can only be called as “**Instances.**” The process of “Instancing,” or making a copy of a function block, is performed in the header of the POU in which the instance is to be used. In this header the function block will be declared as a variable and the resulting instance is given a name. It is possible to declare multiple instances with different names from one and the same function block within the same POU. The instances are then called in the body of the POU and the ‘**Actual**’ parameters are passed to the ‘**Formal**’ parameters. Each instance can be used more than once.

Entering IEC Function Block CTU

① To create a new name for this instance of the CTU Function Block in this POU, click on the variable name ***Instance*** above the CTU function block. And press F2 to bring up the ***Variable selection*** dialogue. Fill in the resulting window as shown on the next page.



Variable Selection (Mode NewVar)

Libraries

<ALL>
<Header>
<Global Variables>
Manufacturer_Lib
Standard_Lib

Variables: Batch_Counter

+Batch_Counter

Class: VAR

Identifier: Batch_Counter

Address:

Type: CTU

Initial:

Comment: Batch Counter

Type: CTU

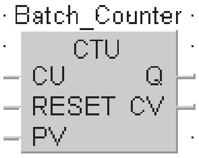
Type Class: Function Blocks

IEC 61131-3

VAR Batch_Counter : CTU; Batch Counter

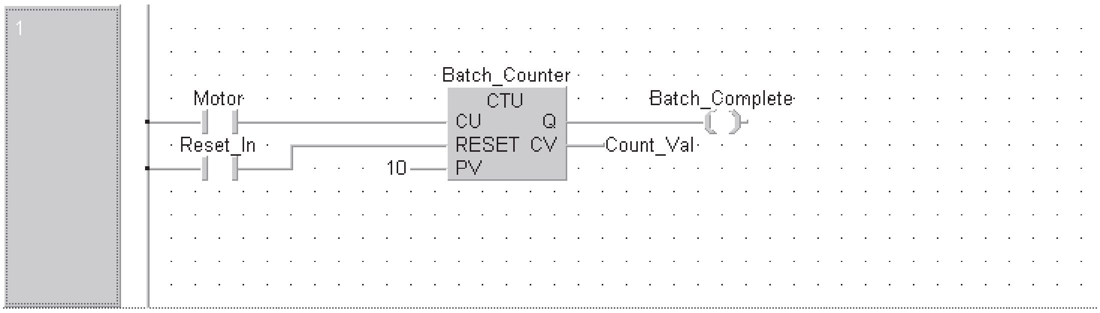
☑ Minimize Dialog after apply

Autoextern: □



② Click on ***Apply***, then ***Update*** and the variable name will change as shown on the left.

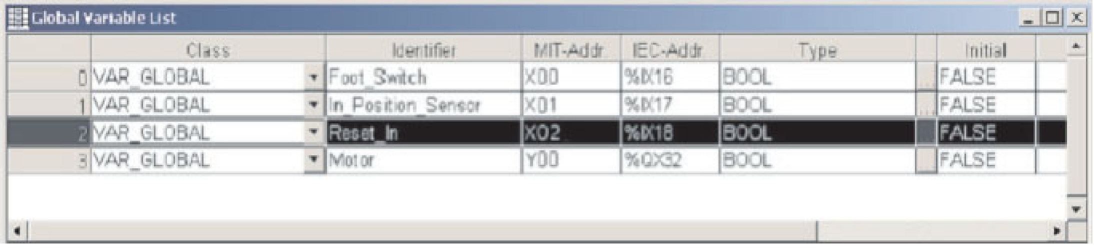
③ Continue to enter the program as previously described so that the following display is achieved:



When entering the PV and CV values, use the variable VAR= =VAR buttons respectively.

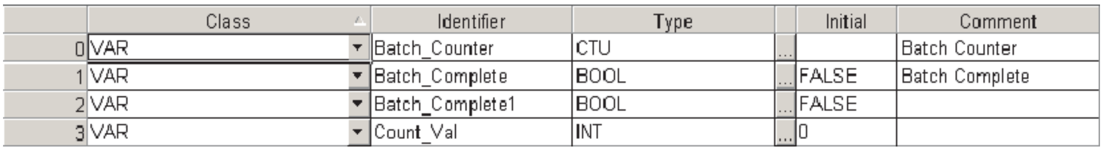
Adding entries to the GVL

Note, in particular: “Reset_In” (Global) - is a new Input mapped from the MELSEC boolean address X02 or IEC %IX2. This requires a new entry into the GVL as follows:



Global Variable List

	Class	Identifier	MIT-Addr	IEC-Addr	Type	Initial
0	VAR_GLOBAL	Foot_Switch	X00	%IX16	BOOL	FALSE
1	VAR_GLOBAL	In_Position_Sensor	X01	%IX17	BOOL	FALSE
2	VAR_GLOBAL	Reset_In	X02	%IX18	BOOL	FALSE
3	VAR_GLOBAL	Motor	Y00	%QX32	BOOL	FALSE

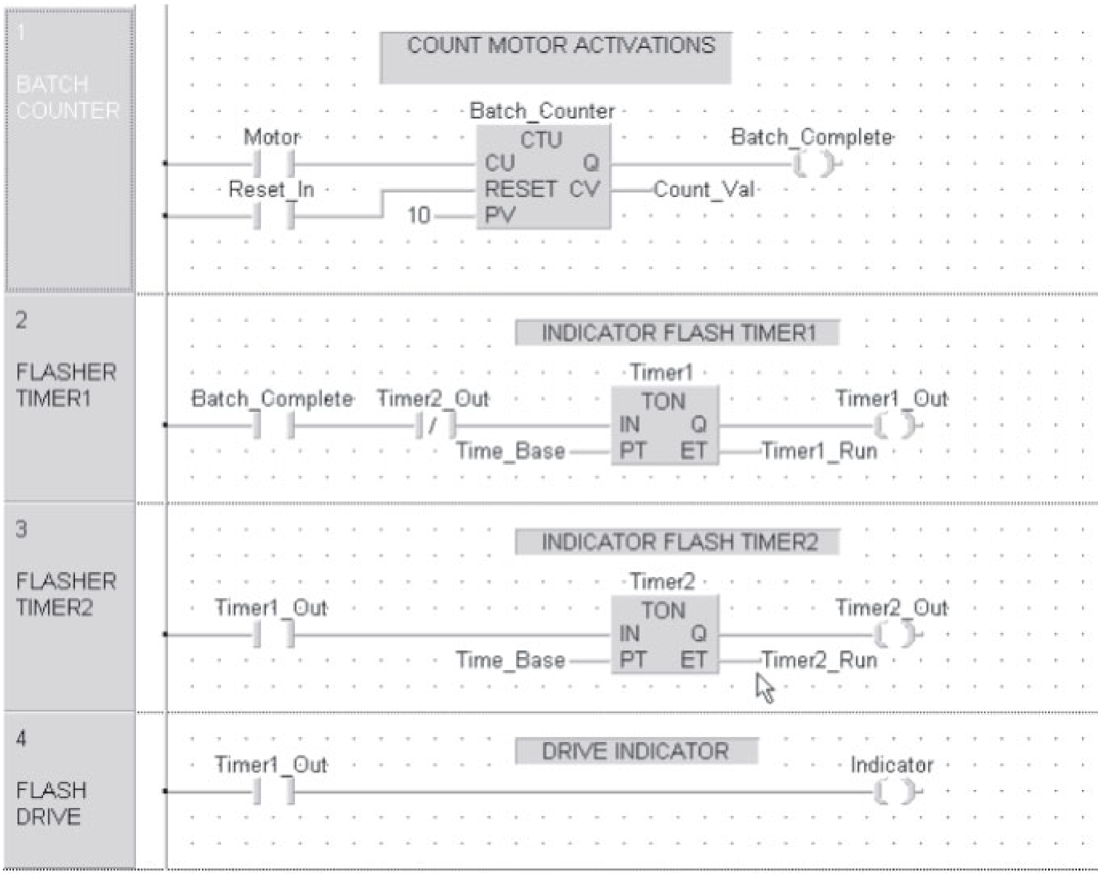


	Class	Identifier	Type	Initial	Comment
0	VAR	Batch_Counter	CTU		Batch Counter
1	VAR	Batch_Complete	BOOL	FALSE	Batch Complete
2	VAR	Batch_Complete1	BOOL	FALSE	
3	VAR	Count_Val	INT	0	

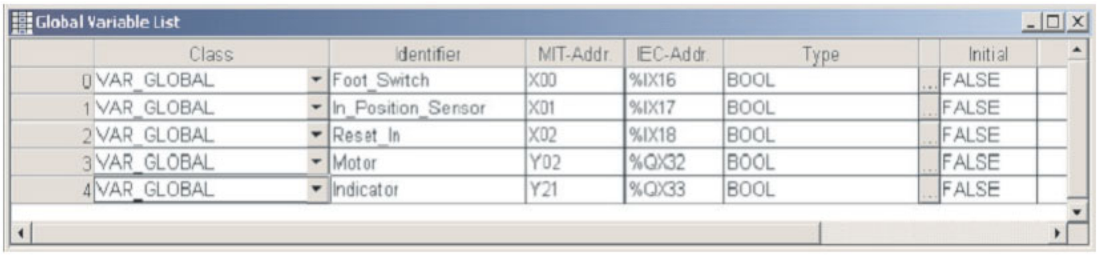
When all new entries are complete, click the check button then the ‘Rebuild All’ button to check and assemble the project.

Timing Function

Create the following Ladder Networks below the batch counting routine in the Batch_Count POU as shown:



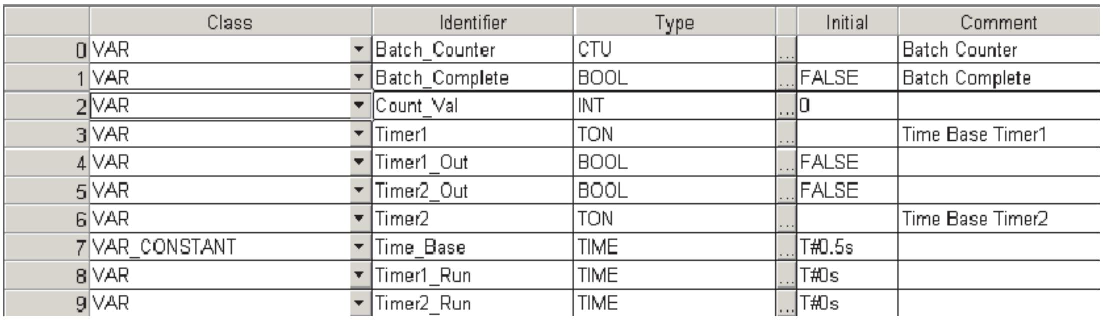
When the editing task has been completed, the GVL should appear thus:



Global Variable List

	Class	Identifier	MIT-Addr.	IEC-Addr.	Type	Initial
0	VAR_GLOBAL	Foot_Switch	X00	%IX16	BOOL	FALSE
1	VAR_GLOBAL	In_Position_Sensor	X01	%IX17	BOOL	FALSE
2	VAR_GLOBAL	Reset_In	X02	%IX18	BOOL	FALSE
3	VAR_GLOBAL	Motor	Y02	%QX32	BOOL	FALSE
4	VAR_GLOBAL	Indicator	Y21	%QX33	BOOL	FALSE

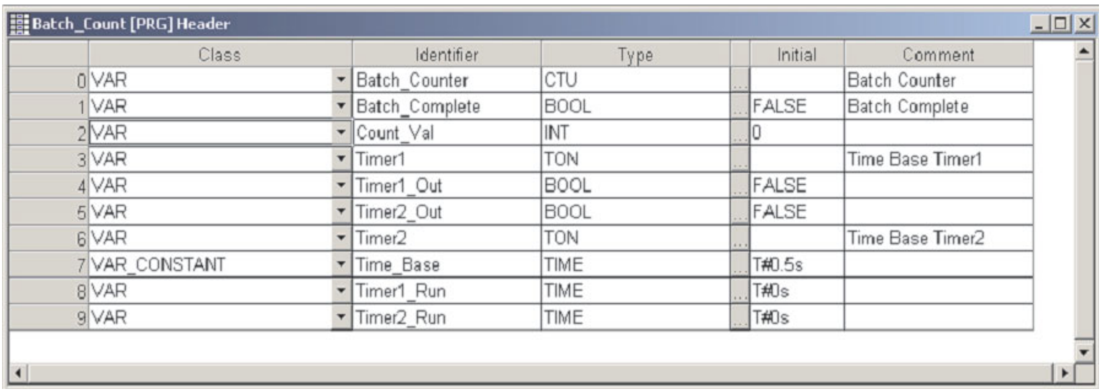
The header (LVL) for the above program “Batch_Count” should now appear as shown:



	Class	Identifier	Type	Initial	Comment
0	VAR	Batch_Counter	CTU		Batch Counter
1	VAR	Batch_Complete	BOOL	FALSE	Batch Complete
2	VAR	Count_Val	INT	0	
3	VAR	Timer1	TON		Time Base Timer1
4	VAR	Timer1_Out	BOOL	FALSE	
5	VAR	Timer2_Out	BOOL	FALSE	
6	VAR	Timer2	TON		Time Base Timer2
7	VAR_CONSTANT	Time_Base	TIME	T#0.5s	
8	VAR	Timer1_Run	TIME	T#0s	
9	VAR	Timer2_Run	TIME	T#0s	

When all new entries are complete, click the check button then the ‘Rebuild All’ button to check and assemble the project.

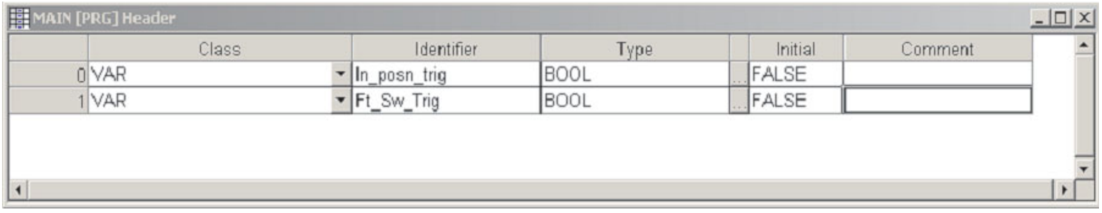
For the POU, “Batch_Count” header



Batch_Count [PRG] Header

	Class	Identifier	Type	Initial	Comment
0	VAR	Batch_Counter	CTU		Batch Counter
1	VAR	Batch_Complete	BOOL	FALSE	Batch Complete
2	VAR	Count_Val	INT	0	
3	VAR	Timer1	TON		Time Base Timer1
4	VAR	Timer1_Out	BOOL	FALSE	
5	VAR	Timer2_Out	BOOL	FALSE	
6	VAR	Timer2	TON		Time Base Timer2
7	VAR_CONSTANT	Time_Base	TIME	T#0.5s	
8	VAR	Timer1_Run	TIME	T#0s	
9	VAR	Timer2_Run	TIME	T#0s	

For the POU, “MAIN” header:



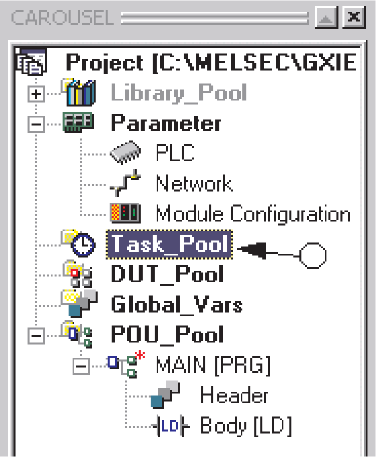
MAIN [PRG] Header

	Class	Identifier	Type	Initial	Comment
0	VAR	In_posn_trig	BOOL	FALSE	
1	VAR	Ft_Sw_Trig	BOOL	FALSE	

4.2.6 Creating a new Task

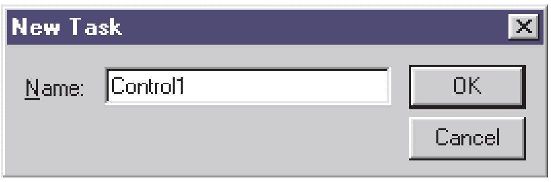
In order for the POUs “MAIN” and "Batch_Count" to be assembled and executed in the PLC, they must be specified as valid tasks in the ***Task Pool***.

① Click once to highlight the ***TASK_Pool*** icon in the Project Navigation area.



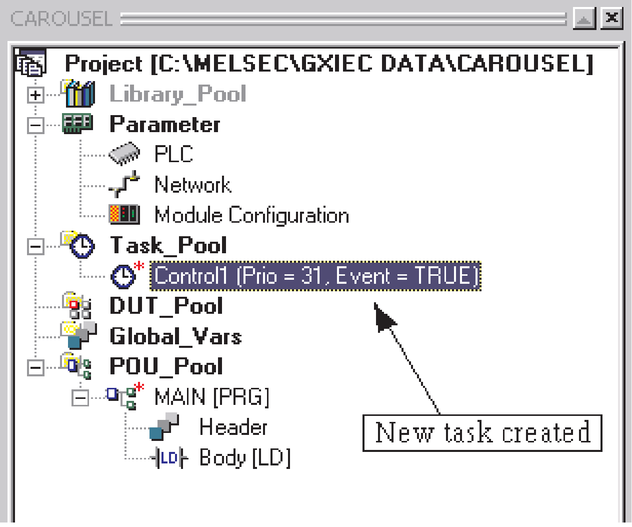
② Then click on the Task button TSK on the Toolbar. Alternatively, ‘Right Click’ the task pool icon in the Project navigation window and select the ***New Task*** option from the menu.

③ Enter the name of the new task ("Control1") in the prompt window.


New Task

Name: Control1

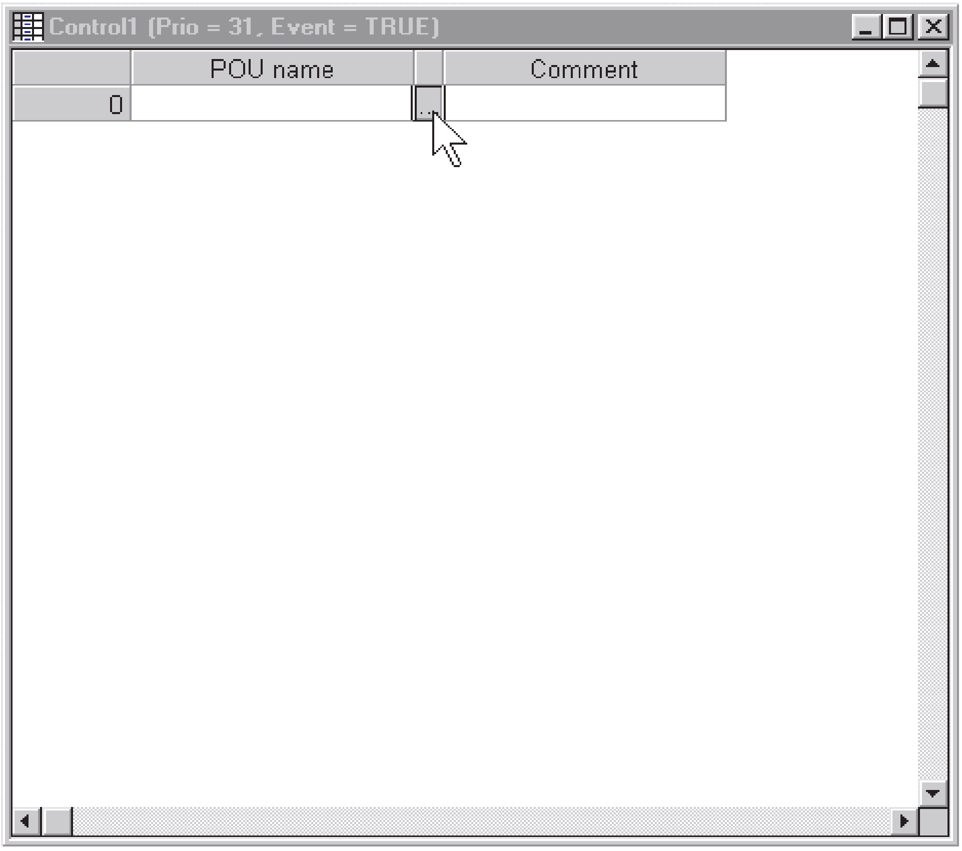
④ Click ***OK*** and the Project Navigation window now shows the newly created task called “Control1”:



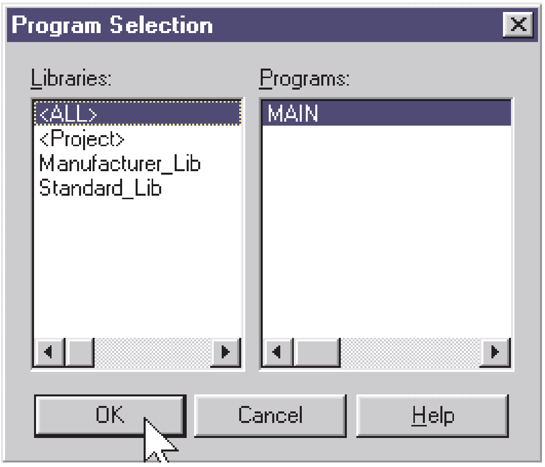
Assigning the POU to Task

The newly created task “Control1” must now reference a POU.

① Double click the ***Control1*** Task icon in the Project Navigation Window; the ‘task event list’ window will be displayed:



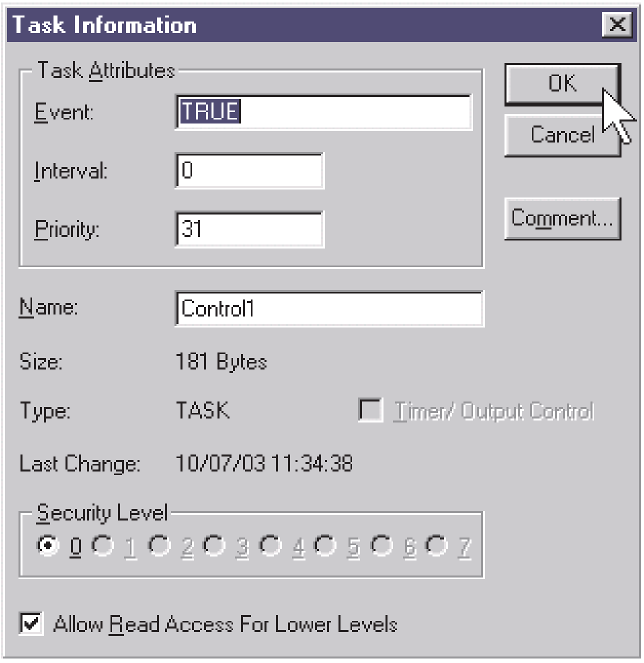
② Click on the centre ‘choice browse’ ellipsis as shown above. The following prompt dialogue is displayed:



③ Choose MAIN and click ***OK*** to complete the assignment operation.

Task Properties

The properties for the task can be displayed by right clicking the mouse on the required task pool entry (i.e. Control1) and selecting ***Properties*** from the menu. The following task settings window is displayed:



Task Information

Task Attributes

Event: TRUE

Interval: 0

Priority: 31

Name: Control1

Size: 181 Bytes

Type: TASK □ Timer/ Output Control

Last Change: 10/07/03 11:34:38

Security Level

● 0 ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7

☑ Allow Read Access For Lower Levels

- Task Attributes
  - Event = TRUE: Always execute
  - Interval = 0: Set to zero because ***Event*** is always true.
  - Priority = 31: 31 is lowest priority i.e. is scanned last.

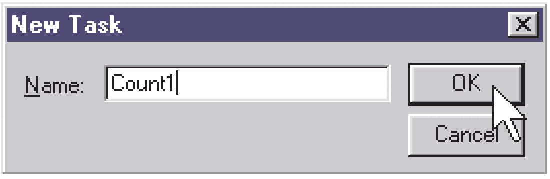
Before continuing, it is a good idea to “SAVE” the project; click on the Save 💾 Button.

Creation of a new task for the POU "Batch-Count"

The POU "Batch-Count" needs also to be referenced (called) by a task in the ‘Task Pool’.

① To create a new task, Right Click on the ‘Task_Pool’ icon on the Project Navigation Window (PNW) and select ***New Task*** from the presented menu. Alternatively, follow the previous procedure, clicking once on the Task_Pool Icon to highlight it on the PNW and click the ‘New Task’ TSK icon on the toolbar.

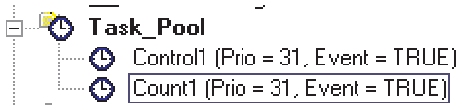
② Enter the name "Count1" into the prompt window as illustrated:



New Task

Name: Count1

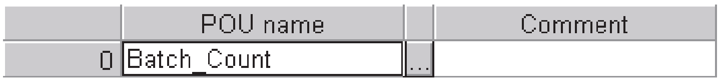
The new task will appear under the previous Task “Control1” in the task Pool:



Task_Pool
- Control1 (Prio = 31, Event = TRUE)
- Count1 (Prio = 31, Event = TRUE)

③ Double click on the new task icon, ‘Count1’ in the PNW.

④ Assign the remaining POU to this task:



	POU name		Comment
0	Batch_Count	...	

When complete, click the check button then the ‘Rebuild All’ button to check and assemble the project.

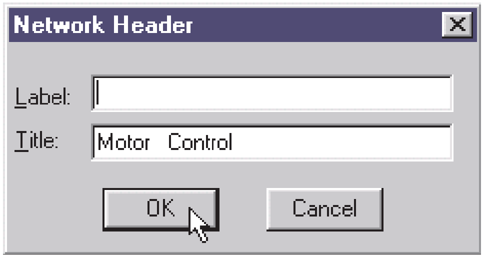
Save the project using the save button. The project is now complete and must therefore be transferred to the PLC.

4.2.7 Program Documentation

Network Header

Titling the network header is optional and provides a means to identify the program network with a descriptive title of up to 22 characters. This can assist handling projects where large numbers of networks are present.

① With Network 1 selected, click the ***Network Header*** button or double click the mouse pointer over the network header area and enter the following data into the Title field **ONLY** – leave the ***Label*** field **Blank** as this has another function:

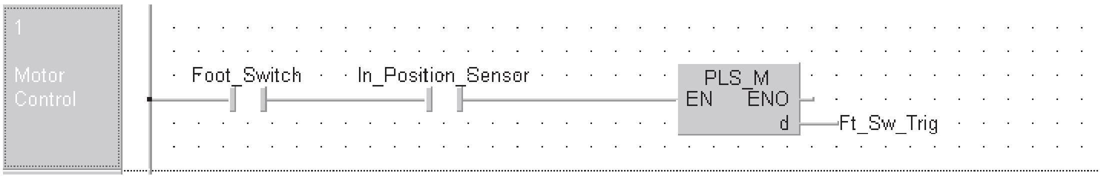


Network Header

Label:

Title: Motor Control

② Click ***OK*** and the network header will be displayed on the left hand side of the screen:



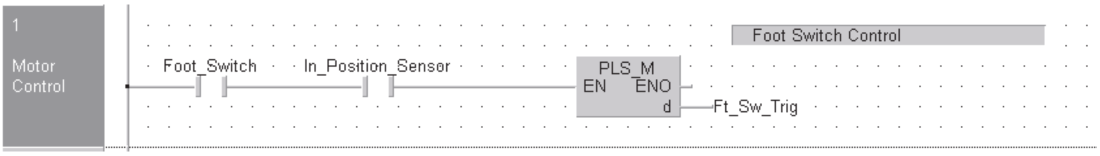
Note that the title may require pre-formatting (Padding with spaces), depending on the screen resolution set, to read correctly as the text auto wraps to fit into the horizontal space available (22 characters max).

Network Comments

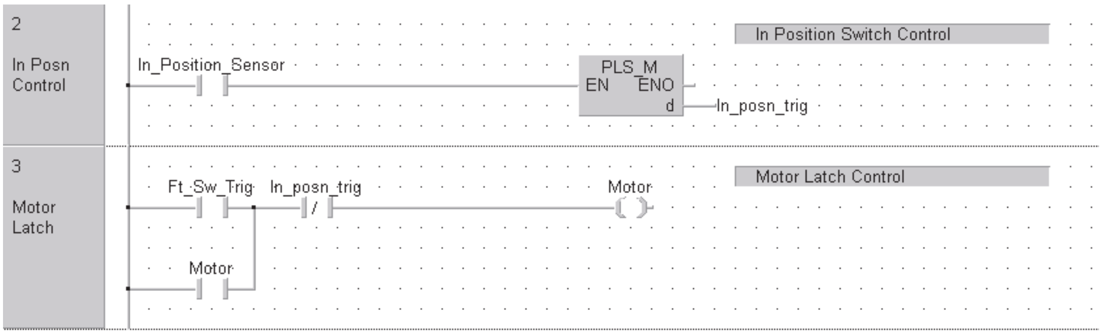
Comments enable virtually freehand text descriptors to be added anywhere inside the ladder network area. This is vital to provide descriptions of the operation of the program.

① To create a comment, press the 'Comment Button' on the toolbar.

② The mouse pointer changes to , click the left mouse button wherever the comment is to be placed and type the required text and press <Enter>:



Continue to complete the program documentation as follows:



Moving the position of a comment

With the cursor in ‘Select Mode’, it is possible to grab and move the comments around the ladder network area. To achieve this, click and hold on the left part of the comment dialogue area. Drag the comment anywhere on the screen and release the mouse button.

Deleting a comment

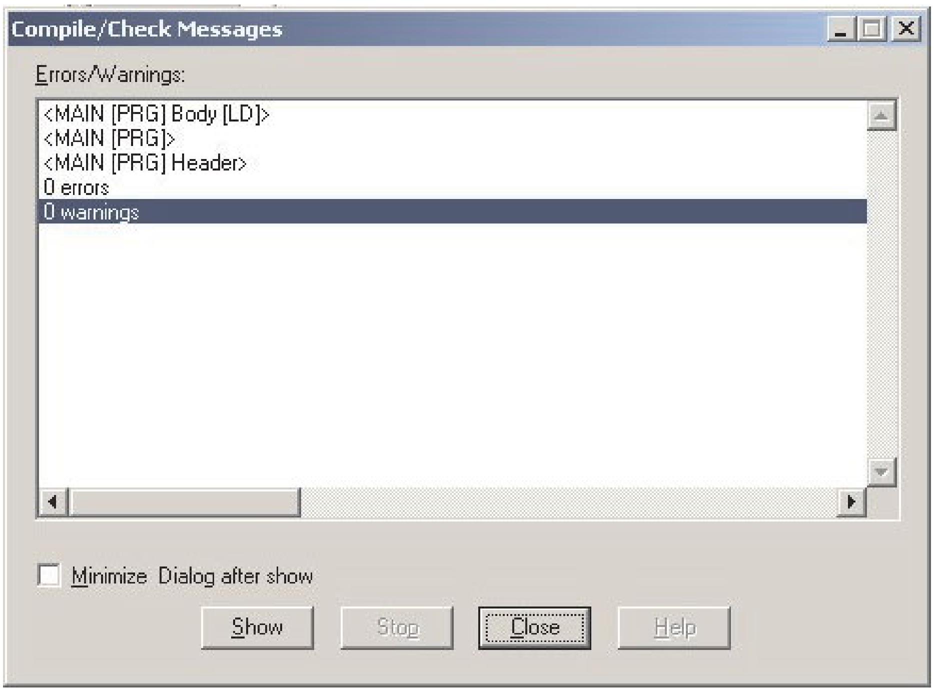
Click once on the comment to highlight and press the <Delete> key on the keyboard.

Cutting / Copying a comment

Duplication of comments is achieved by clicking on the left hand end of the source comment to highlight it. Use windows cut/copy – paste procedure and click the mouse once again to set position of destination comment in another network.

4.2.8 Checking and Building the Project Code

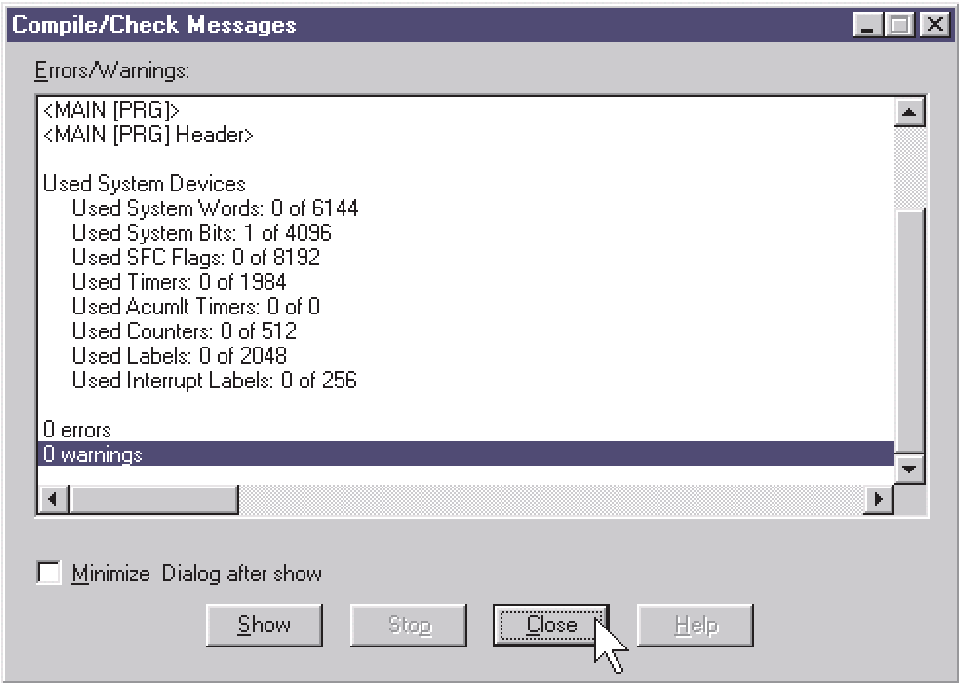
① When the Ladder Diagram is complete and task has been specified in the Task Pool, once again press the “Check” button on the tool bar to check the program for errors; the following dialogue should be displayed:


Compile/Check Messages

Errors/Warnings:

```
<MAIN [PRG] Body [LD]>
<MAIN [PRG]>
<MAIN [PRG] Header>
0 errors
0 warnings
```

② Click either the ‘Build’ button or the ‘Rebuild All’ button on the toolbar and if all is well, the following compiler messages are reported:



Compile/Check Messages

Errors/Warnings:

```
<MAIN [PRG]>
<MAIN [PRG] Header>

Used System Devices
    Used System Words: 0 of 6144
    Used System Bits: 1 of 4096
    Used SFC Flags: 0 of 8192
    Used Timers: 0 of 1984
    Used Acumlt Timers: 0 of 0
    Used Counters: 0 of 512
    Used Labels: 0 of 2048
    Used Interrupt Labels: 0 of 256

0 errors
0 warnings
```

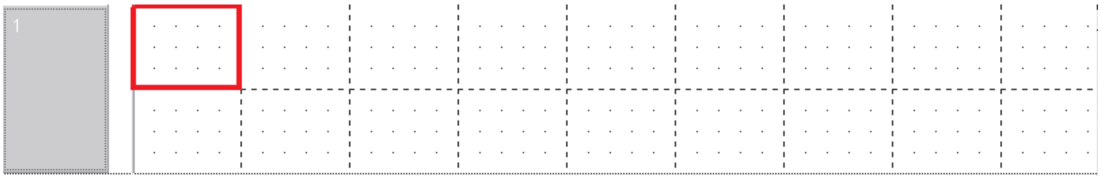
□ Minimize Dialog after show

③ Click ***Close*** to exit this display.

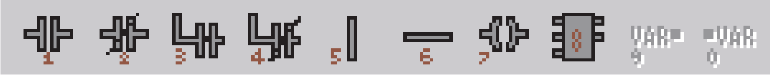
4.2.9 Illustration: Guided Ladder Entry Mode

In addition to the freehand ladder entry methods, GX IEC Developer Version 6 onward features a ***Guided Ladder Entry*** Monitor method which may be used to aid Ladder program entry. This entry method may prove to be helpful to those wishing to make the transition to GX-IEC Developer who have had previous familiarity with Mitsubishi's MEDOC package and GX-Developer.

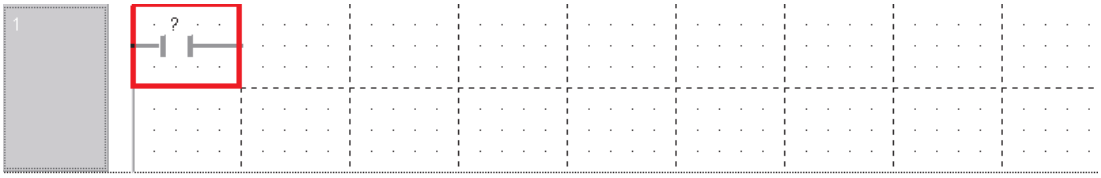
① Enter the ***Guided Entry Monitor*** mode by pressing the button on the tool bar. The following matrix is placed into the edit area:



② Use the following buttons on the toolbar to select the ladder symbols. The corresponding number may be pressed to select the appropriate symbol from the keyboard, thus eliminating the need to use the mouse:



③ Select the 'Normally Open' Contact symbol "1" and the following will be displayed:

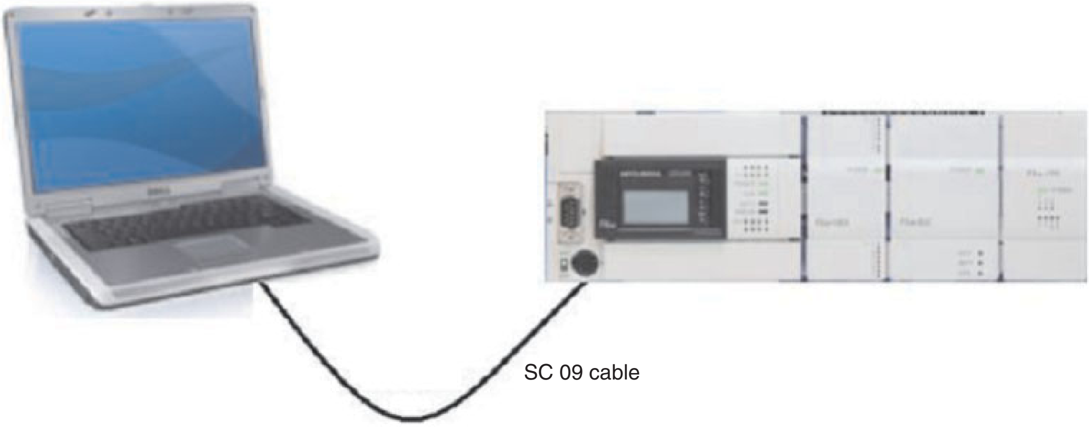


The program may continue to be entered using the "F2" button on the keyboard or click on the button on the tool bar to call up the variables selection window as previously described.

4.3 Project Download Procedures

4.3.1 Connection with Peripheral Devices

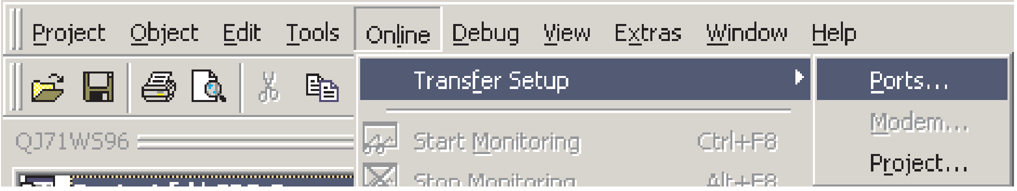
The following notes describe how the project is downloaded to a FX PLC. To connect a controller of the FX family and a PC, the SC 09 converter is used to convert the RS232 common mode serial signals 'to and from' the computer to the RS 422 serial-differential format required by the PLC.



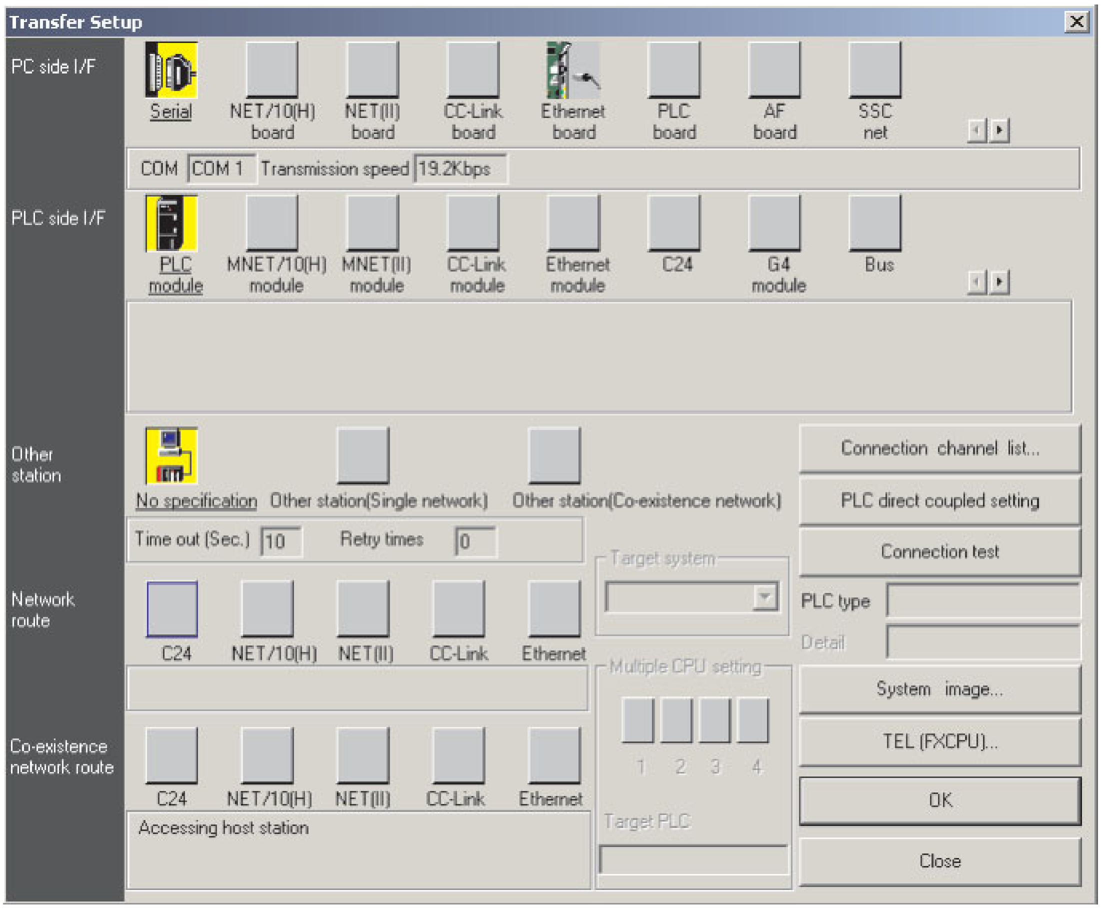
4.3.2 Communications Port Setup

Before the project can be downloaded into the PLC CPU for the first time, the communication and download settings must be configured.

① From the ***Online*** menu, select ***Transfer Setup*** and then ***Ports***:



The ***Connection Setup*** window shown on the next page will be displayed.

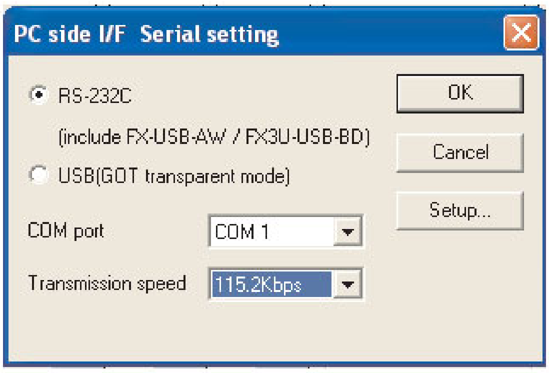


Transfer Setup

COM COM 1 Transmission speed 19.2Kbps

Time out (Sec.) 10 Retry times 0

② Double click the mouse on the yellow ***PC side I/F*** – ***Serial*** button and the following dialogue window is displayed:



PC side I/F Serial setting

● RS-232C
(include FX-USB-AW / FX3U-USB-BD)

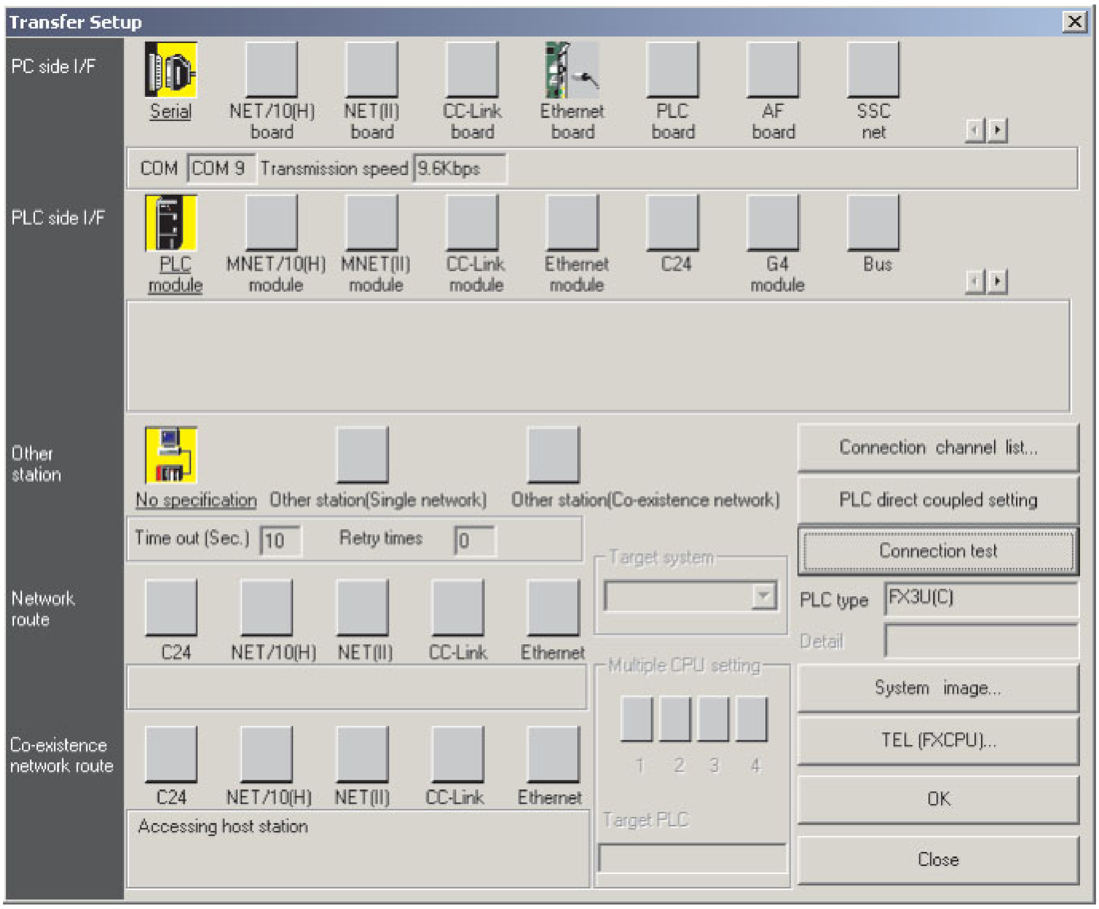
○ USB(GOT transparent mode)

COM port COM 1

Transmission speed 115.2Kbps

③ Select ***RS232C*** as shown above and click ***OK***.

④ Click on the ***Connection Test*** button to check PC-PLC communications are ok:



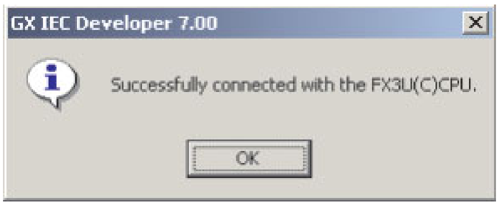
Transfer Setup

COM COM 9 Transmission speed 9.6Kbps

Time out (Sec.) 10 Retry times 0

PLC type FX3U(C)

⑤ The following message should be displayed:



GX IEC Developer 7.00

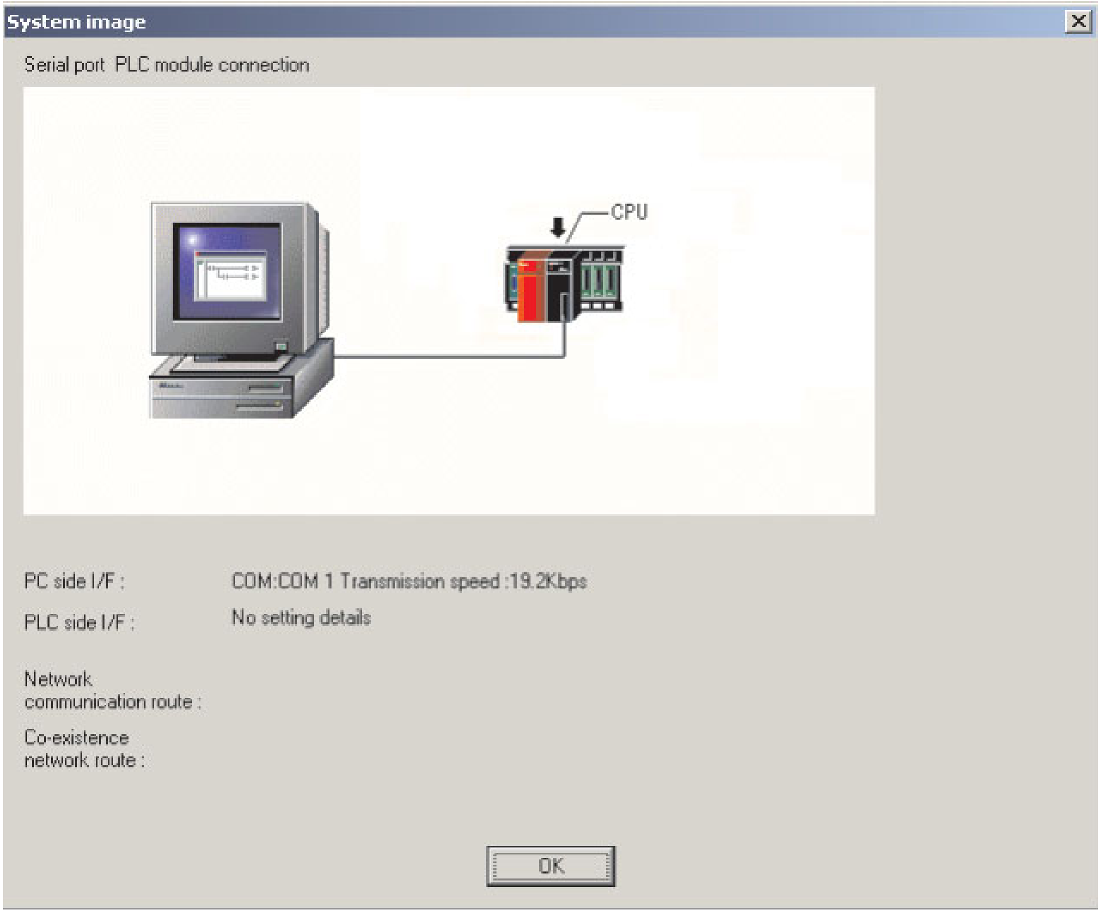
Successfully connected with the FX3U(C)CPU.

⑥ Click ***OK*** to close this message.

If an error message is displayed, check connections and settings with the PLC.

Connection Setup Route

① To obtain a pictorial view of the Connection setup route, select the ***System Image*** button



Serial port PLC module connection

PC side I/F : COM:COM 1 Transmission speed :19.2Kbps

PLC side I/F : No setting details

Network communication route :

Co-existence network route :

② Click ***OK*** to clear the display.

NOTE

When using a standard RS232 Serial Port to communicate with the PLC, if another device is already connected to the selected COM (n) interface, for example a serial mouse; Select another free serial port.

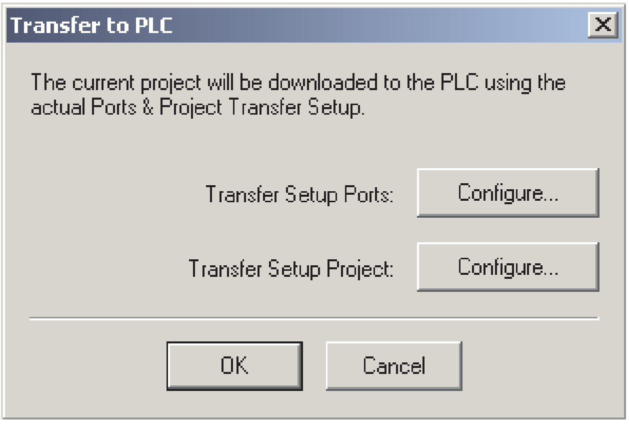
③ Select ***OK*** to close the ***System image*** display and return to the ***Connection setup*** display. Than click the ***OK*** button to close the ***Connection Setup*** window. If you leave the ***Connection Setup*** window using the ***Close*** button, the settings are not saved.

4.3.3 Downloading the project

① Once the setting up procedures is complete, click on the “Download Project” icon on the toolbar.

Transfer Setup

② Click the ***Configure*** button to setup the “Transfer parameters” for the project.

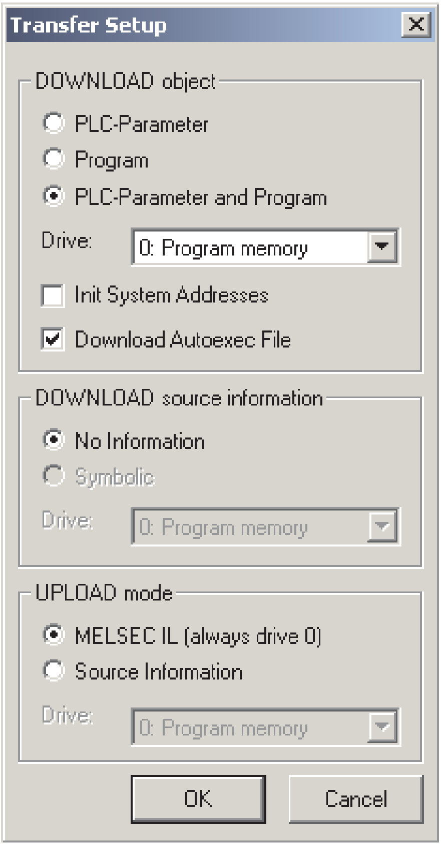


Transfer to PLC

The current project will be downloaded to the PLC using the actual Ports & Project Transfer Setup.

Transfer Setup Ports: Configure...

Transfer Setup Project: Configure...



Transfer Setup

DOWNLOAD object

○ PLC-Parameter
○ Program
● PLC-Parameter and Program

Drive: 0: Program memory

□ Init System Addresses
☑ Download Autoexec File

DOWNLOAD source information

● No Information
○ Symbolic

Drive: 0: Program memory

UPLOAD mode

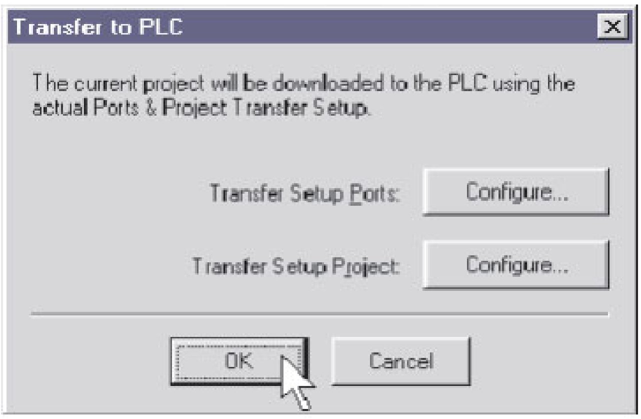
● MELSEC IL (always drive 0)
○ Source Information

Drive: 0: Program memory

② Click on ***PLC-Parameter and Program***

③ Click on ***OK*** to confirm the selection.

④ To send the project to the PLC, click the ***OK*** button to execute the transfer.



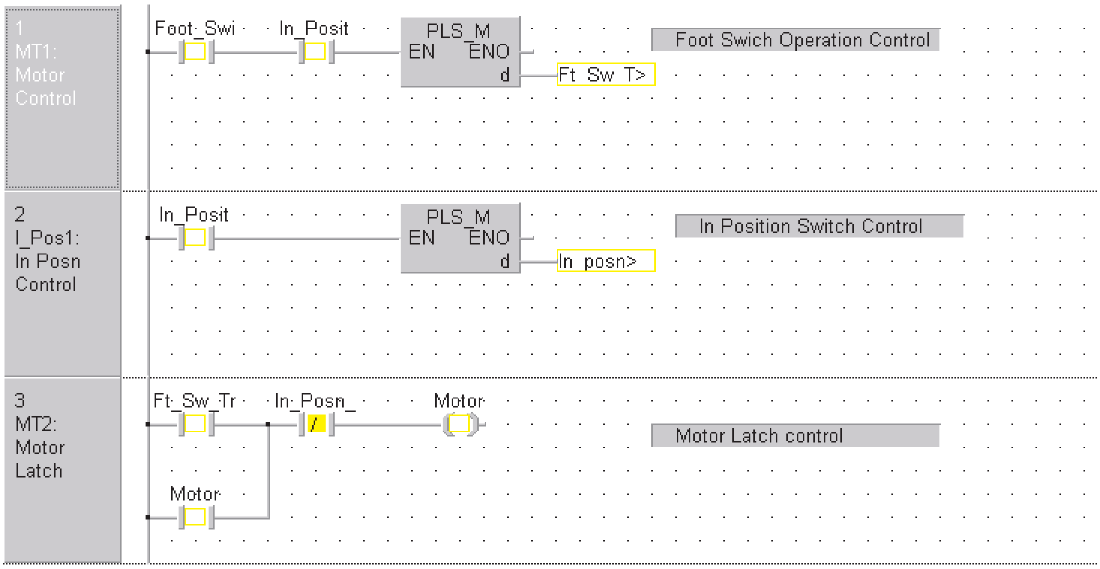
The current project will be downloaded to the PLC using the actual Ports & Project Transfer Setup.

4.4 Monitoring the Project

Ensure that the PLC is switched to RUN and no errors are present.

Display the body of the MAIN ladder program.

Click on the Monitor Mode Icon on the toolbar and observe the ladder display:



NOTE Depending on the colour attributes set, monitored variables will be displayed with a coloured surround (Default: Yellow). Values of any analogue variable will be displayed on the monitored networks as appropriate.

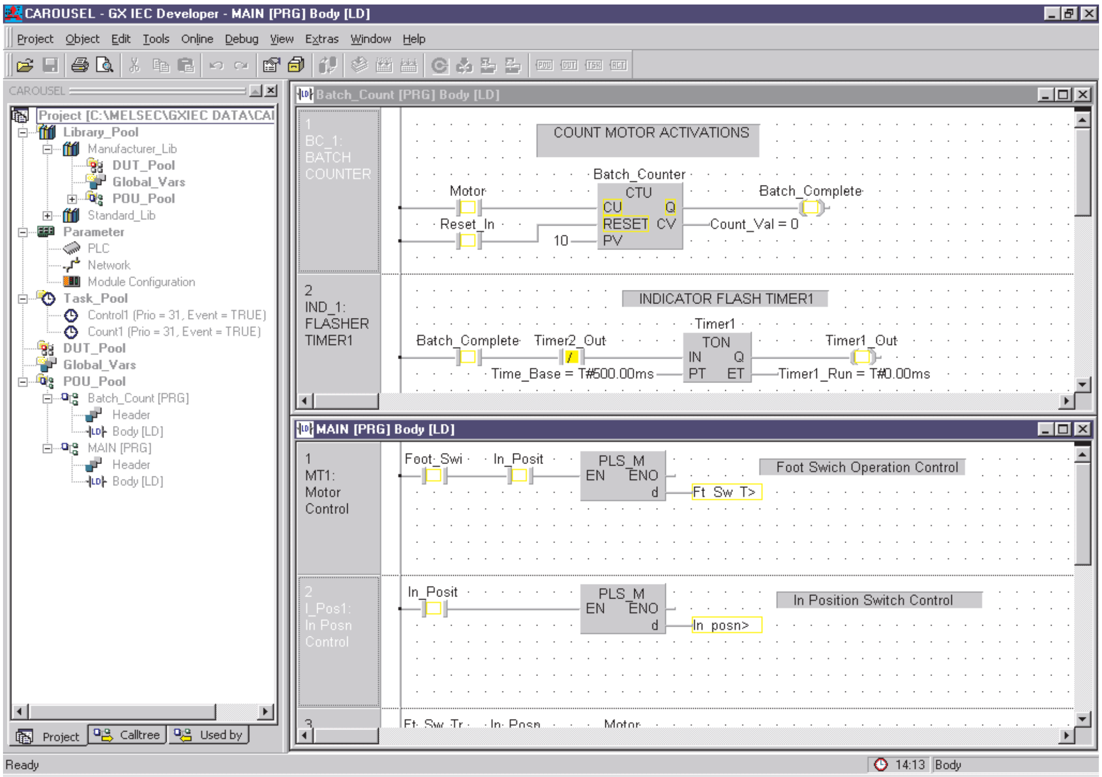
4.4.1 Split / Multi Window Monitoring

To monitor both of the project' POU's simultaneously, open both POU bodies and select ***Tile Horizontally*** from the ***Window*** menu.

NOTE

Important: It should be noted that when initially entering monitor mode with , only the screen in focus will be monitored. This is to avoid unneeded communication traffic occurring from other screens that have been opened but are not necessarily in the focus (i.e. opened but behind).

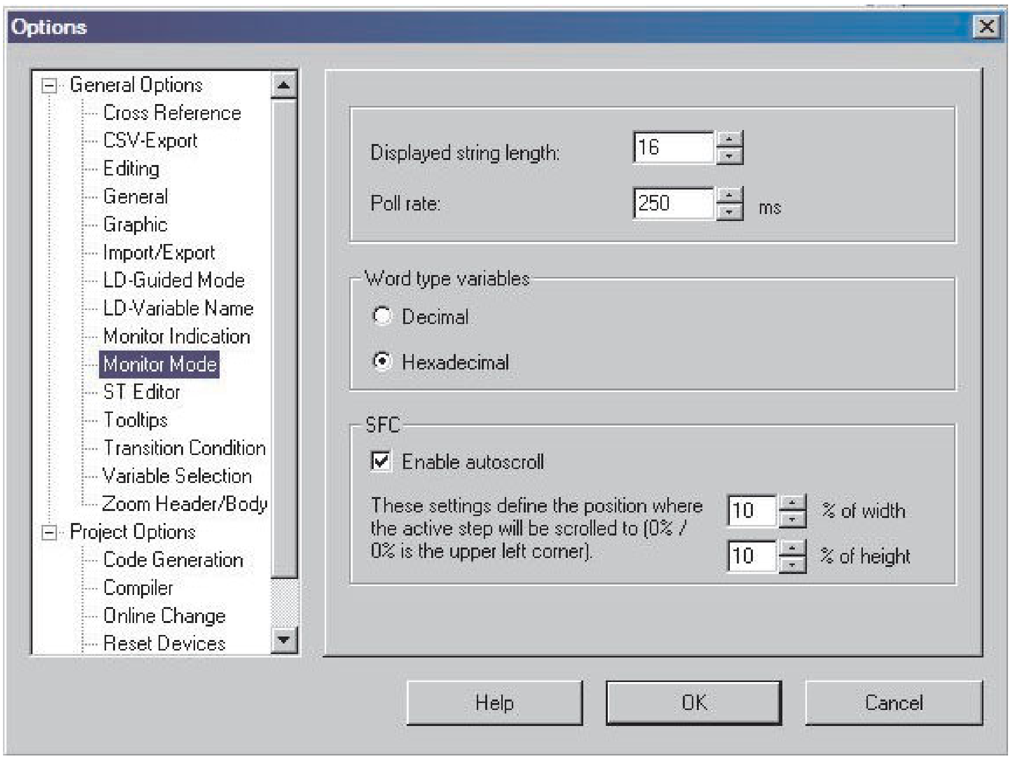
To begin monitoring the content of additional windows, click inside that window and select ***Start Monitoring*** from the ***Online*** Menu:



NOTE

Due to the serial communications handshake, be prepared to wait a few seconds for the monitor information to be registered between GX IEC Developer and the PLC.

The rate of communication polling from GX IEC Developer to the PLC may be increased by adjusting the poll rate setting. Select ***Monitor Mode*** from the ***Extras/Options*** menu and enter a new value for the ***Poll rate***.



Displayed string length: 16

Poll rate: 250 ms

Word type variables

○ Decimal

● Hexadecimal

SFC

☑ Enable autoscroll

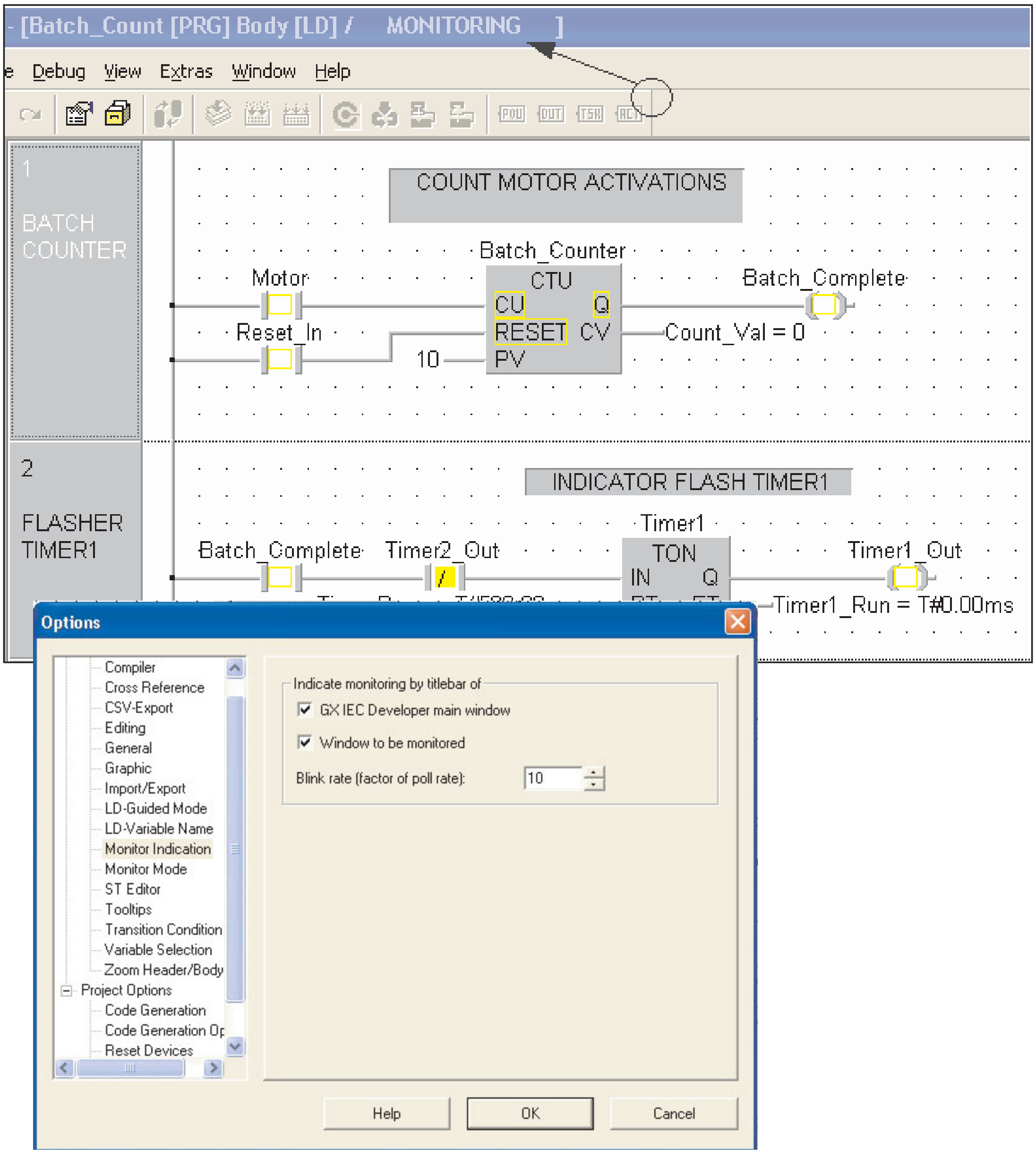
These settings define the position where the active step will be scrolled to (0% / 0% is the upper left corner).

10 % of width

10 % of height

4.4.2 Adjusting Monitor Visibility

To adjust the visibility of the monitor mode, select ‘Extras/Options/Monitor Indication’ and a flashing message can be enabled, to appear where chosen. The blink rate of the “Monitoring” banner can be set by the User:



Indicate monitoring by titlebar of

☑ GX IEC Developer main window

☑ Window to be monitored

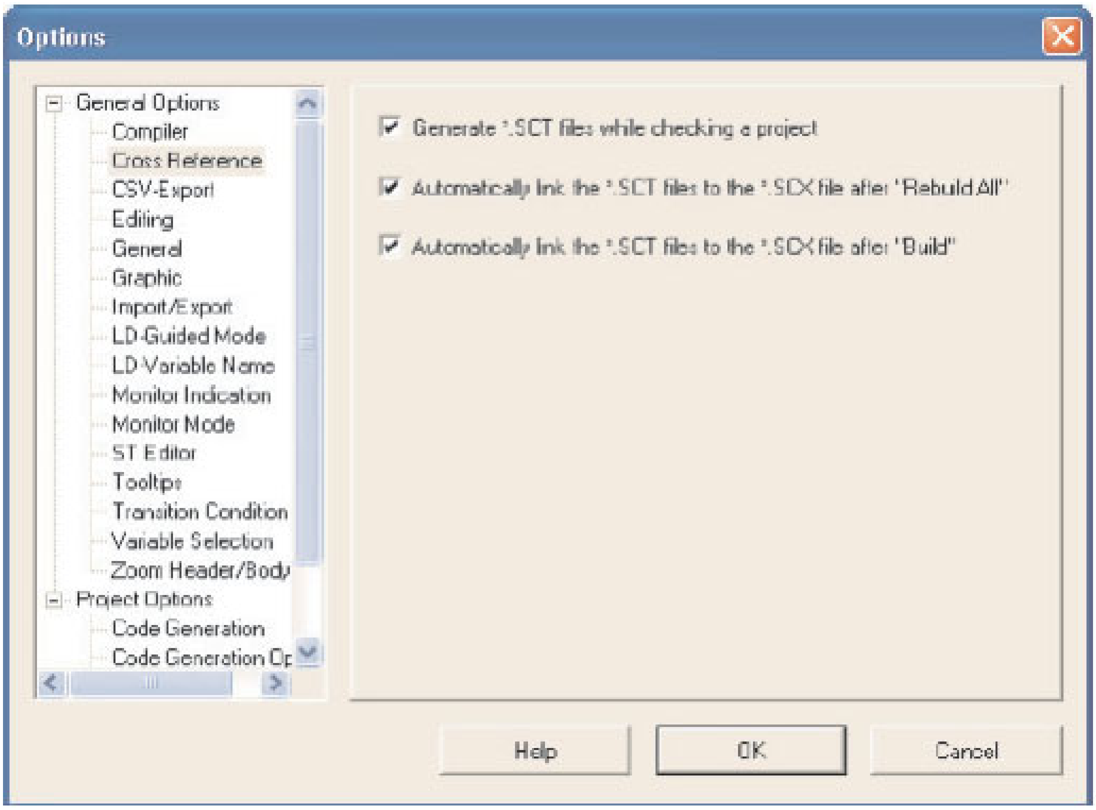
Blink rate (factor of poll rate): 10

4.5 Cross Reference List

To generate a Cross Reference List:

① Open the ***Extras/Options*** Menu and select ***Cross Reference***

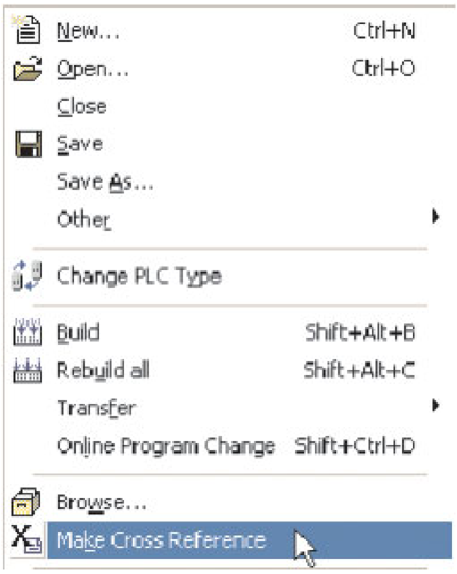
② Check both options shown and re-compile the project.



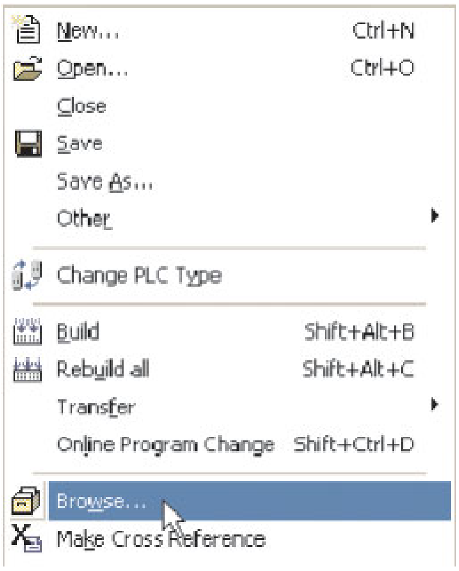
☑ Generate *.SCT files while checking a project

☑ Automatically link the *.SCT files to the *.SCX file after 'Rebuild All'

☑ Automatically link the *.SCT files to the *.SCX file after 'Build'

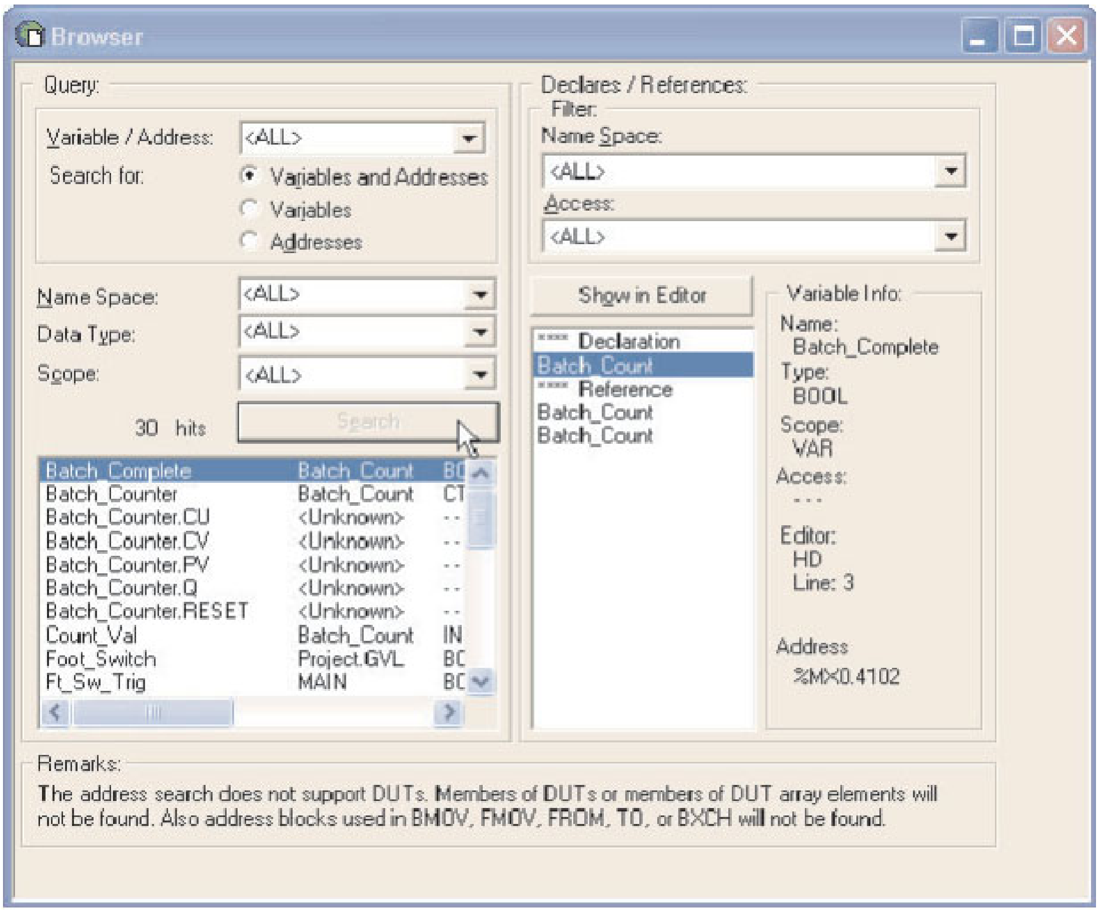


③ Then select ***Make Cross Reference*** from the ***Project*** Menu and the list is generated.



④ Open the Browser, either from the ***Project*** menu, or via the toolbar icon .

⑤ Click on the ***Search*** button and the full list will be displayed.


Browser

Query:
Variable / Address: <ALL>
Search for: ● Variables and Addresses ○ Variables ○ Addresses
Name Space: <ALL>
Data Type: <ALL>
Scope: <ALL>
30 hits Search

```
Batch_Complete        Batch_Count   BC
Batch_Counter         Batch_Count   CT
Batch_Counter.CU      <Unknown>     --
Batch_Counter.CV      <Unknown>     --
Batch_Counter.PV      <Unknown>     --
Batch_Counter.Q       <Unknown>     --
Batch_Counter.RESET   <Unknown>     --
Count_Val             Batch_Count   IN
Foot_Switch           Project.GVL   BC
Ft_Sw_Trig            MAIN          BC
```

Declares / References:
Filter:
Name Space: <ALL>
Access: <ALL>
Show in Editor

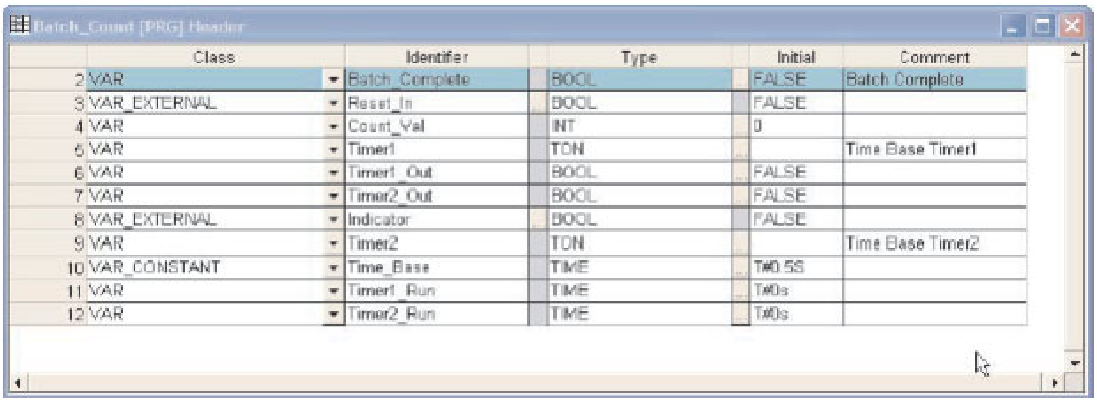
```
**** Declaration
Batch_Count
**** Reference
Batch_Count
Batch_Count
```

Variable Info:
Name: Batch_Complete
Type: BOOL
Scope: VAR
Access: ...
Editor: HD
Line: 3
Address %MX0.4102

Remarks:
The address search does not support DUTs. Members of DUTs or members of DUT array elements will not be found. Also address blocks used in BMOV, FMOV, FROM, TO, or BXCH will not be found.

Specific variables etc. can be searched by using the query selection boxes. Individual details of the highlighted entry are then shown on the right hand side of the window.

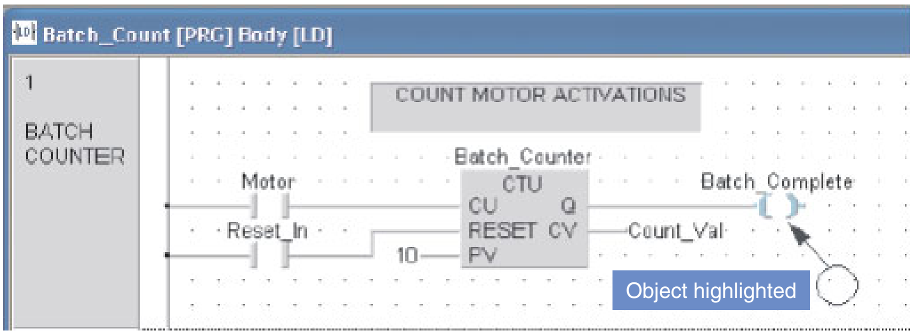
The ***Show in Editor*** button opens the header of the highlighted right hand list element, for example:



Batch_Count [PRG] Header

	Class	Identifier	Type	Initial	Comment
2	VAR	Batch_Complete	BOOL	FALSE	Batch Complete
3	VAR_EXTERNAL	Reset_In	BOOL	FALSE	
4	VAR	Count_Val	INT	0	
5	VAR	Timer1	TON		Time Base Timer1
6	VAR	Timer1_Out	BOOL	FALSE	
7	VAR	Timer2_Out	BOOL	FALSE	
8	VAR_EXTERNAL	Indicator	BOOL	FALSE	
9	VAR	Timer2	TON		Time Base Timer2
10	VAR_CONSTANT	Time_Base	TIME	T#0.5S	
11	VAR	Timer1_Run	TIME	T#0s	
12	VAR	Timer2_Run	TIME	T#0s	

or

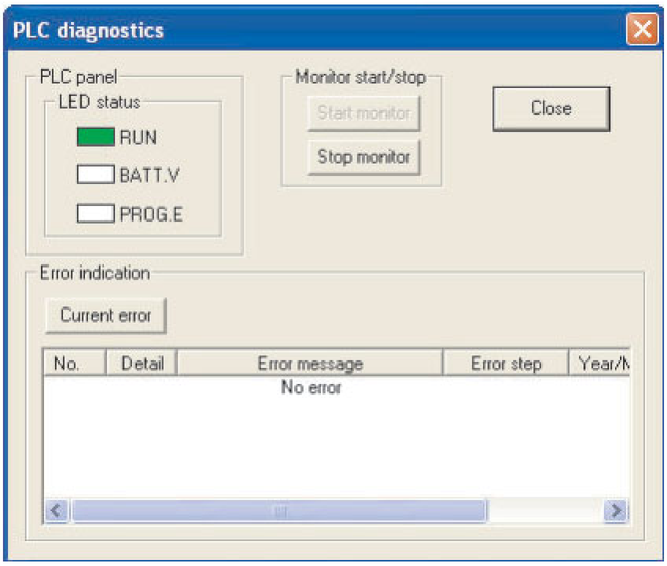


The Cross Reference List may be printed out, using the print facility within GX IEC Developer.

4.6 PLC Diagnostics

In GX IEC Developer various diagnostic functions are available.The functions in the ***Debug*** menu allow to perform precise troubleshooting and error analysis of your application.

Click on ***PLC Diagnostics*** to open the window shown below.



No error

Clear Text Error Message

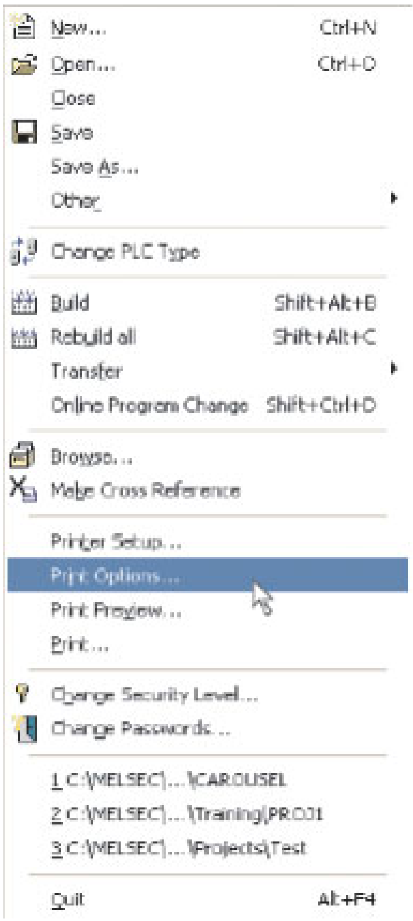
The error data registers of the PLC are evaluated with clear text and respective help texts.

The most important hardware errors such as “Fuse blown” are displayed in a window and evaluated.

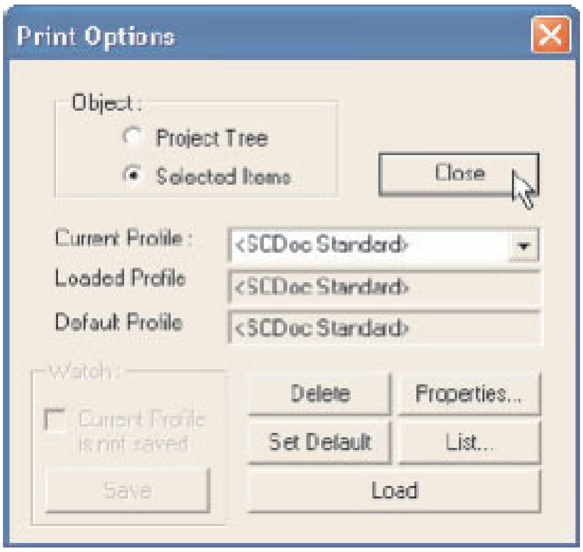
User errors can be determined. These user errors are stored with a self-created text file (USER_ERR.TXT) and allow a quick error correction. The last eight user errors are stored into a FIFO register and only be removed when they no longer occur.

4.7 Project Documentation

Project documentation can be set up using the ***Print Option*** facility from the ***Project*** Menu:



The “Change Configuration” dialogue box can then be seen. Previous project profiles can be retrieved here, or work with the default profile. Either select the ***Project Tree*** for all elements, or ***Selected Items*** for specific highlighted items, open ***Properties***:

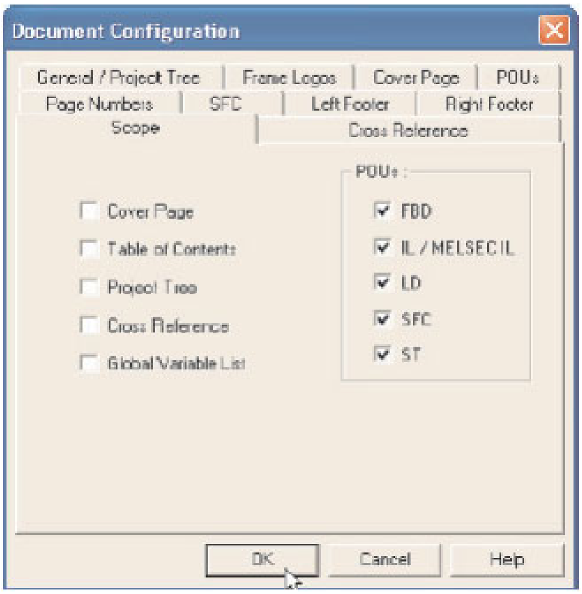

Print Options

Object:
○ Project Tree
● Selected Items

Current Profile: <SCDoc Standard>
Loaded Profile <SCDoc Standard>
Default Profile <SCDoc Standard>

Watch:
□ Current Profile is not saved

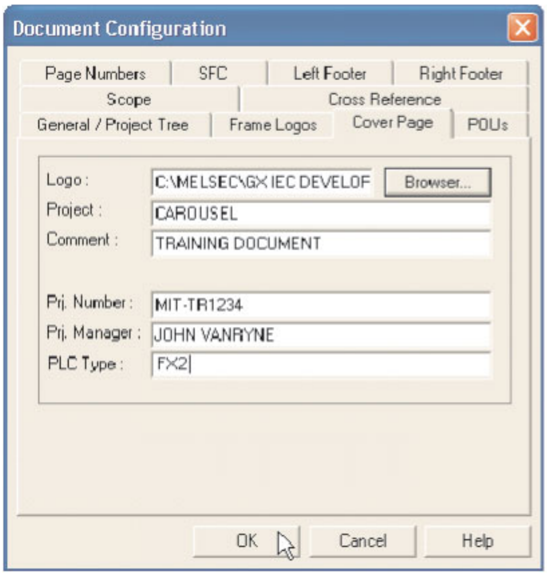
The ***Document Configuration*** folder is shown below. Select the tabs to configure the document as required. In this example, only the COUNTER_FB_CE will be printed, as the ***Selected Items*** option was chosen:



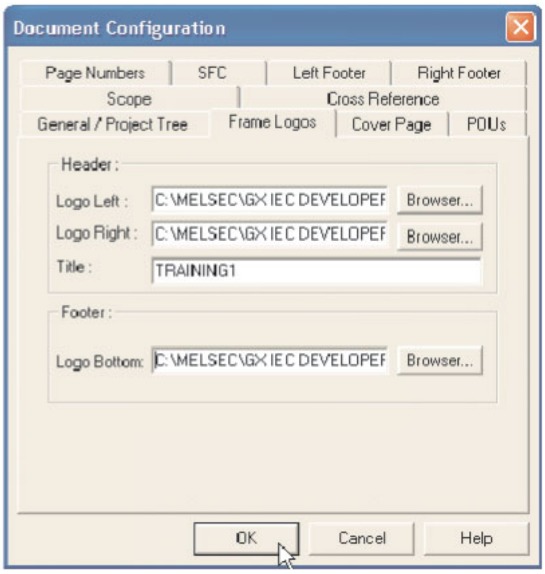
□ Cover Page
□ Table of Contents
□ Project Tree
□ Cross Reference
□ Global Variable List

POUs :
☑ FBD
☑ IL / MELSEC IL
☑ LD
☑ SFC
☑ ST

User defined logos and information can be assigned, in the ***Cover Page*** tab, for the front sheet and for the frame from the ***Frame Logos*** tab:



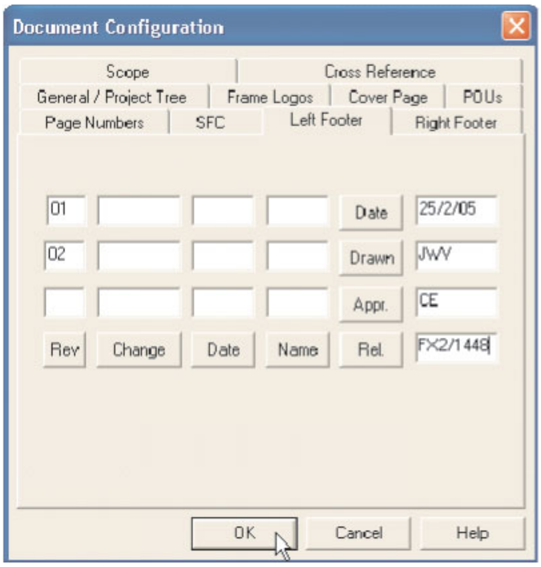
Logo : C:\MELSEC\GX IEC DEVELOF
Project : CAROUSEL
Comment : TRAINING DOCUMENT
Prj. Number : MIT-TR1234
Prj. Manager : JOHN VANRYNE
PLC Type : FX2



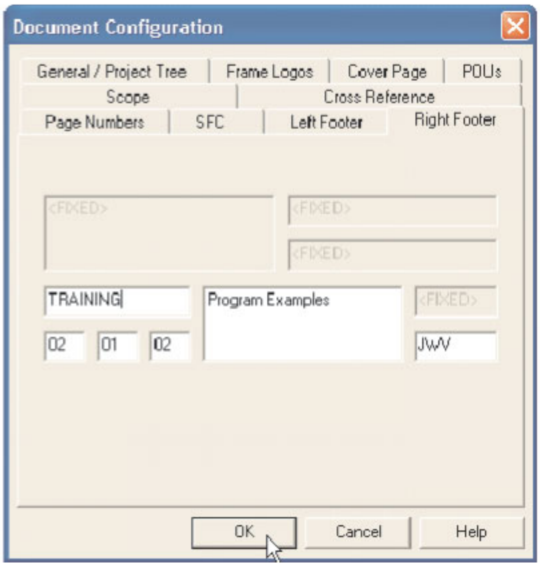
Header :
Logo Left : C:\MELSEC\GX IEC DEVELOPEF
Logo Right : C:\MELSEC\GX IEC DEVELOPEF
Title : TRAINING1

Footer :
Logo Bottom: C:\MELSEC\GX IEC DEVELOPEF

Detailed information can be assigned, to the left and right footers. The field labels in the ***Left Footer*** dialogue can be renamed, by clicking on the name buttons, as required:

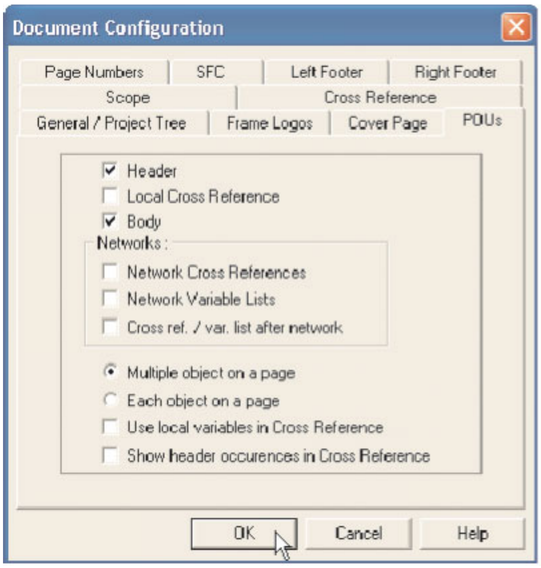


01
02
Date 25/2/05
Drawn JWV
Appr. CE
Rev Change Date Name Rel. FX2/1448

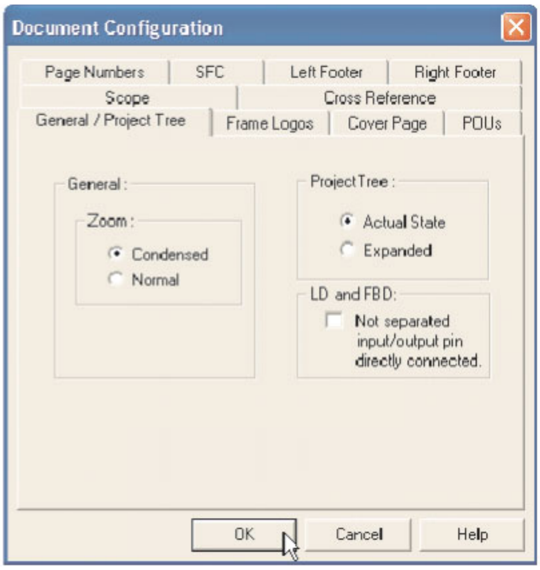


<FIXED> <FIXED>
<FIXED>
TRAINING Program Examples <FIXED>
02 01 02 JWV

Specification for POU appearance and general project specifications are available from the ***POUs*** and ***General/Project Tree*** tabs.



☑ Header
□ Local Cross Reference
☑ Body
Networks :
□ Network Cross References
□ Network Variable Lists
□ Cross ref. / var. list after network
● Multiple object on a page
○ Each object on a page
□ Use local variables in Cross Reference
□ Show header occurences in Cross Reference

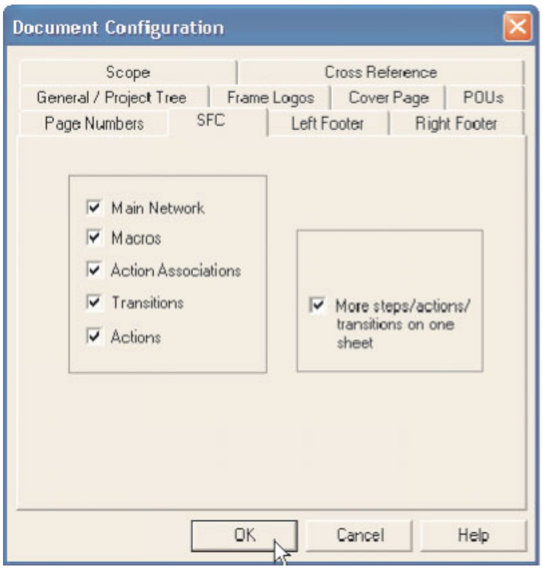


General :
Zoom :
● Condensed
○ Normal

Project Tree :
● Actual State
○ Expanded

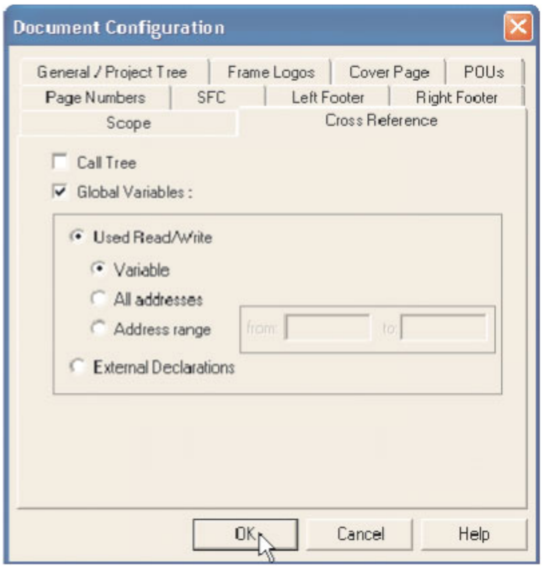
LD and FBD:
□ Not separated input/output pin directly connected.

Specification for SFC appearance and cross reference specifications, are available from the ***SFC*** and ***Cross Reference*** tabs:



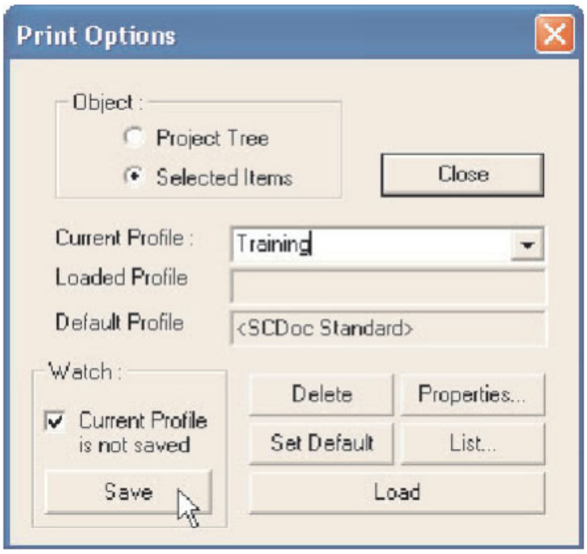
☑ Main Network
☑ Macros
☑ Action Associations
☑ Transitions
☑ Actions

☑ More steps/actions/transitions on one sheet



□ Call Tree
☑ Global Variables :
● Used Read/Write
● Variable
○ All addresses
○ Address range from: to:
○ External Declarations

The configured profile can be saved, by simply naming the ***Current Profile*** field and then clicking the ***Save*** button. It can then be recalled at any time using the selection box:



Object :
○ Project Tree
● Selected Items

Current Profile : Training
Loaded Profile
Default Profile <SCDoc Standard>

Watch :
☑ Current Profile is not saved

5 Program Example

5.1 An Alarm System

Task description

The objective is to create an alarm system with several alarm circuits and a delay function for arming and disarming the system.

- The system will be armed after a delay between turning the switch and activation. This provides time for the user to leave the house without tripping the alarm.
- An n alarm is activated after a delay to make it possible to disarm the system after entering the house.
- The siren will only be sounded for 30 seconds, but the alarm lamp will remain activated until the system is disarmed.

Operation and function of the alarm system

- The system will be armed with a key switch, with a 20-second delay.
- When an alarm is triggered a siren and a blinking alarm lamp are activated after a delay of 10 seconds.
- The key-operated switch will also be used to deactivate the alarm system.

I/O List

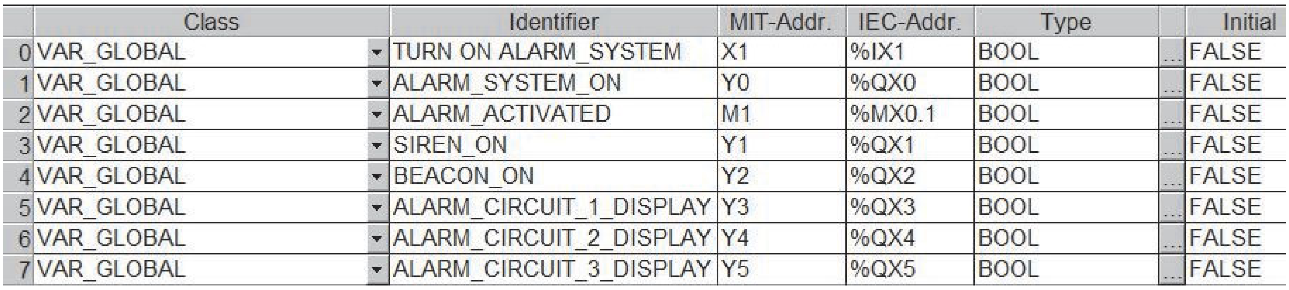
The following table contains an overview of the used inputs, outputs, and timer.The inputs are used to read the status of the alarm circuits. The siren and a blinking alarm lamp are connected to outputs. The timer are used for the required delays.

Function		Adress	Remarks
Input	Arm system	X1	Make contact (key-operated switch)
	Alarm circuit 1	X2	Break contacts (an alarm is triggered when the input has the signal state “0”)
	Alarm circuit 2	X3	
	Alarm circuit 3	X4	
Output	Display “system armed”	Y0	The outputs functions are activated when the corresponding outputs are switched on (set). For example, if Y1 is set the acoustic alarm will sound.
	Acoustic alarm (siren)	Y1	
	Optical alarm (rotating beacon)	Y2	
	Alarm circuit 1 display	Y3	
	Alarm circuit 2 display	Y4	
	Alarm circuit 3 display	Y5	
Timer	Arming delay	T0	Time: 20 seconds
	Alarm triggering delay	T1	Time: 10 seconds
	Siren activation duration	T2	Time: 30 seconds

5.1.1 Method

① Create a new project and name it “Alarm_System”.

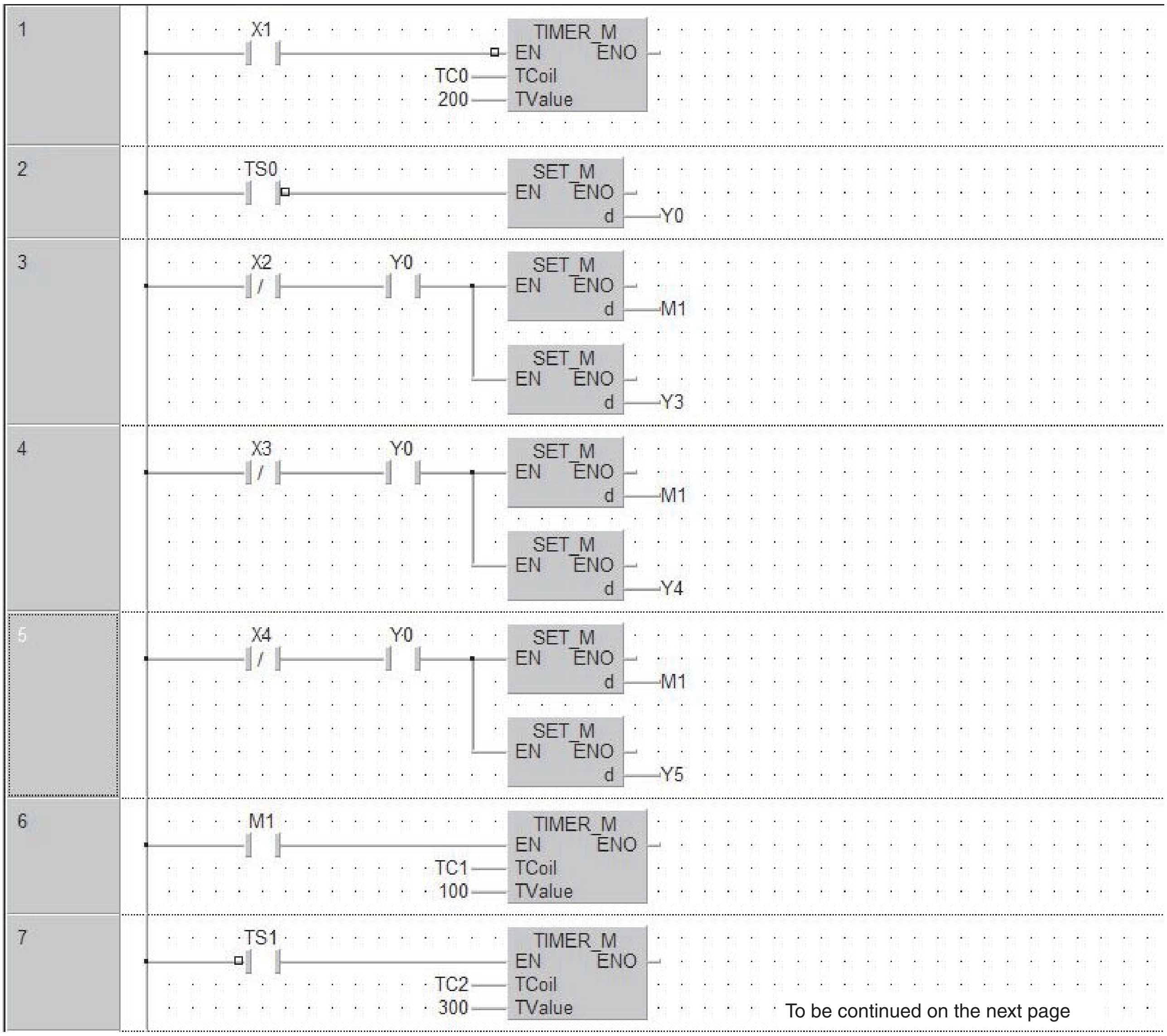
② Enter the following data into the ***Global Variables List***:



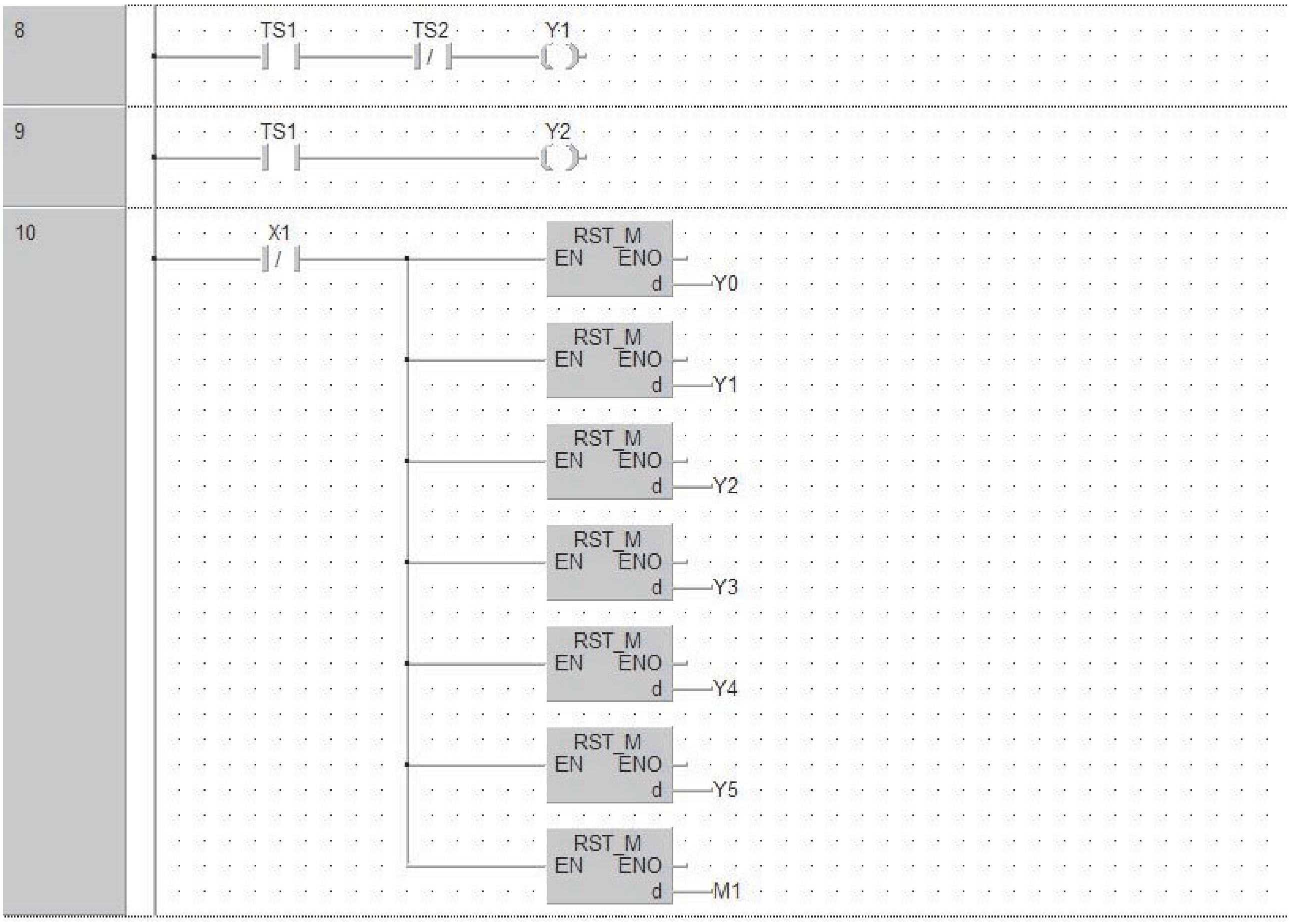
	Class	Identifier	MIT-Addr.	IEC-Addr.	Type	Initial
0	VAR_GLOBAL	TURN ON ALARM_SYSTEM	X1	%IX1	BOOL	FALSE
1	VAR_GLOBAL	ALARM_SYSTEM_ON	Y0	%QX0	BOOL	FALSE
2	VAR_GLOBAL	ALARM_ACTIVATED	M1	%MX0.1	BOOL	FALSE
3	VAR_GLOBAL	SIREN_ON	Y1	%QX1	BOOL	FALSE
4	VAR_GLOBAL	BEACON_ON	Y2	%QX2	BOOL	FALSE
5	VAR_GLOBAL	ALARM_CIRCUIT_1_DISPLAY	Y3	%QX3	BOOL	FALSE
6	VAR_GLOBAL	ALARM_CIRCUIT_2_DISPLAY	Y4	%QX4	BOOL	FALSE
7	VAR_GLOBAL	ALARM_CIRCUIT_3_DISPLAY	Y5	%QX5	BOOL	FALSE

③ Create a new POU of Class ***PRG*** (Program Type) and Language ***Ladder Diagram***.

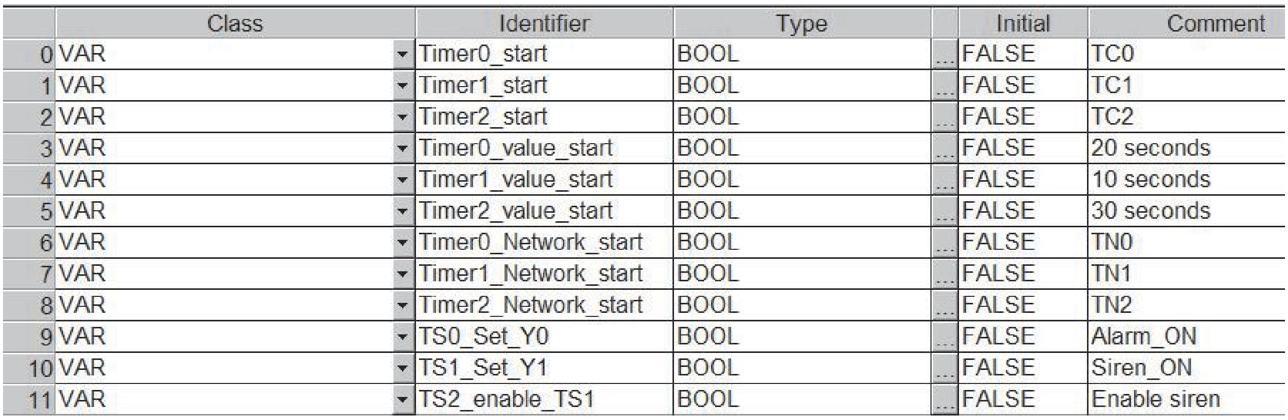
④ Enter the following code into the POU.



To be continued on the next page



The finalised header of the “Main_PRG_LD” POU should read as follows



	Class	Identifier	Type	Initial	Comment
0	VAR	Timer0_start	BOOL	FALSE	TC0
1	VAR	Timer1_start	BOOL	FALSE	TC1
2	VAR	Timer2_start	BOOL	FALSE	TC2
3	VAR	Timer0_value_start	BOOL	FALSE	20 seconds
4	VAR	Timer1_value_start	BOOL	FALSE	10 seconds
5	VAR	Timer2_value_start	BOOL	FALSE	30 seconds
6	VAR	Timer0_Network_start	BOOL	FALSE	TN0
7	VAR	Timer1_Network_start	BOOL	FALSE	TN1
8	VAR	Timer2_Network_start	BOOL	FALSE	TN2
9	VAR	TS0_Set_Y0	BOOL	FALSE	Alarm_ON
10	VAR	TS1_Set_Y1	BOOL	FALSE	Siren_ON
11	VAR	TS2_enable_TS1	BOOL	FALSE	Enable siren

5.1.2 Checking the Example Program "Alarm_System"

① Enter and comment the project. Check the program using the function provided in the tool bar. Build and save the project.

② Download the project to the FX series PLC.

③ Activate the monitor mode to observe the function of the program.

④ Ensure the project is working correctly by monitoring the operation while operating the inputs according to the I/O list shown at the start of this section.

6 Functions and Function Blocks

Below is a table illustrating the comparison between ‘Functions’ and ‘Function Blocks’:

Item	Function Block	Function
Internal variable storage	Storage	No storage
Instancing	Required	Not required
Outputs	– No output – One output – Multiple outputs	One output
Repeated execution with same input values	Does not always deliver the same output value.	Always delivers the same output value.

- Functions are part of the instruction set.
- Functions are included in the standard and manufacturers libraries. i.e. TIMER_M is a function, as is MOV_M, PLUS_M etc. from the Mitsubishi Instruction Set in the Manufacturers Library.
- User defined functions can easily be created out of tested program parts.

  This means that functions can be created i.e. for system/process calculations, and can be stored in libraries and reused many times, with different variable declarations. This would be in the same way that i.e. a MOV instruction would be used but with the advantage of being user specific.

6.1 Functions

Most control programs have some form of maths within them, i.e. for analogue signal conditioning, displaying engineering units etc. These are frequently reused within the program structure.

By using user defined functions, program design time can be dramatically reduced.

6.1.1 Example: Creating a Function

Objective:

Build a Function to change Fahrenheit to Centigrade.

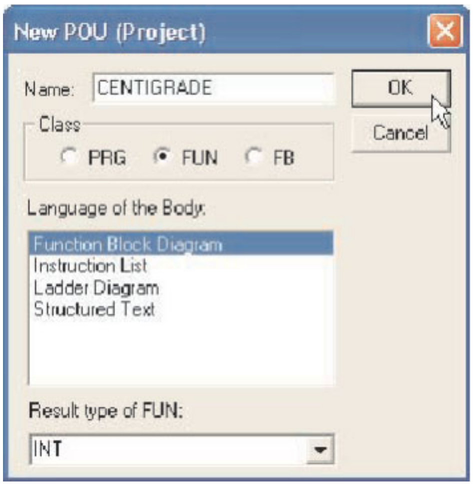
Formula is:

$$Centigrade = \frac{(Fahrenheit - 32) \times 5}{9}$$

The Function will be named “Centigrade” and the input variable will be named “Fahrenheit”.

Procedure

① Select a new POU and name it Centigrade.



New POU (Project)

Name: CENTIGRADE

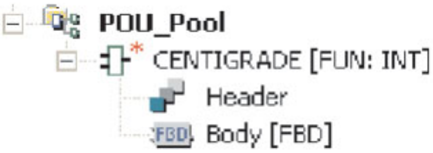
Class: ○ PRG ● FUN ○ FB

Language of the Body: Function Block Diagram

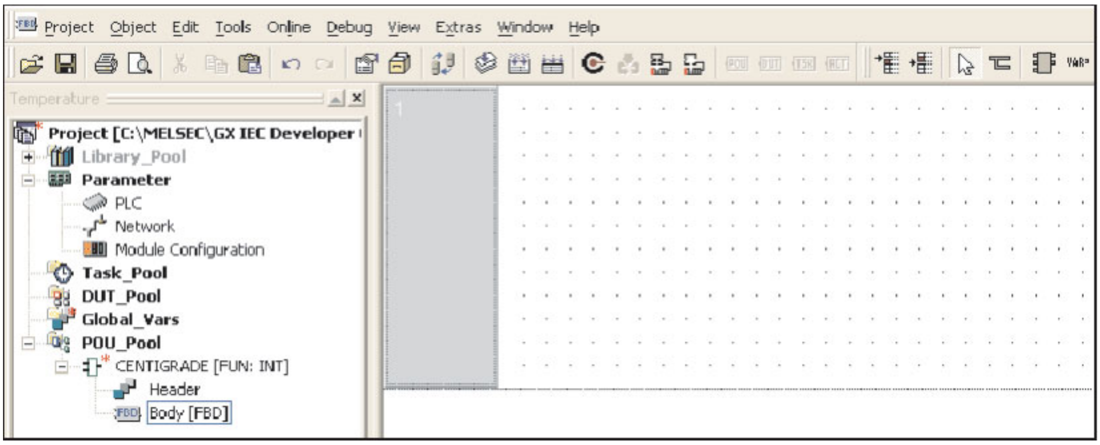
Result type of FUN: INT

This time click the “FUN” option, instead of “PRG.” Select ***Function Block Diagram*** as the editor. The Result Type of FUN should be left as INT (Integer type).

Centigrade will now have appeared on the POU tree:

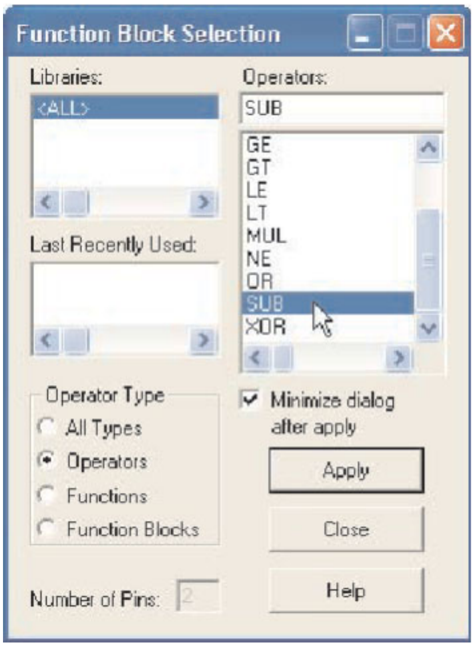


② Double click on the FBD body icon, to open the body network:



Selecting the Function:

① Select the **function block** icon from the toolbar and select ***SUB*** from the operators list:



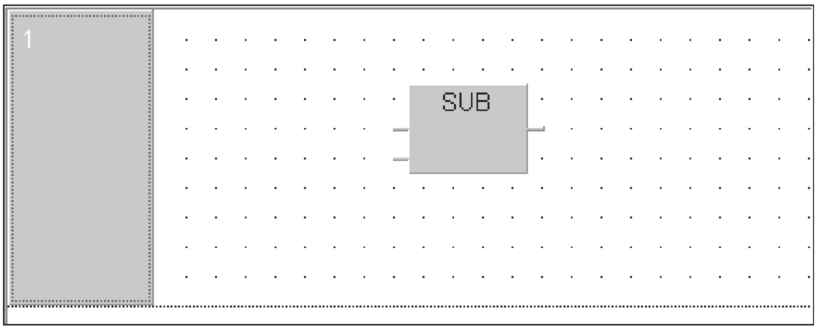
Operators: SUB

○ All Types
● Operators
○ Functions
○ Function Blocks

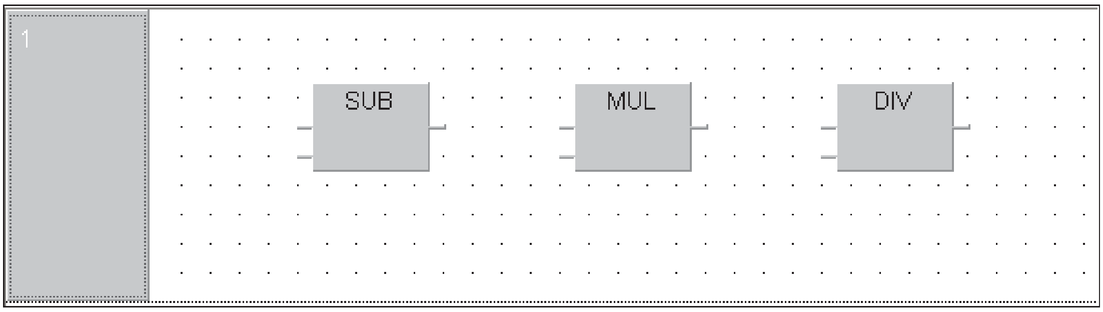
☑ Minimize dialog after apply

Number of Pins: 2

② Using ***Apply*** or double clicking on the selection object, place it on the screen:



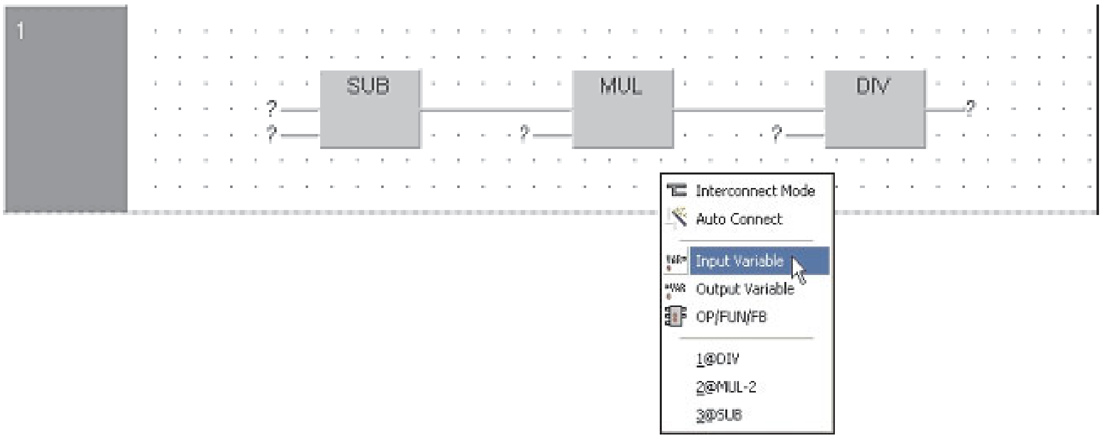
③ Repeat the above process so that the following is visible:



Declaring the Variables

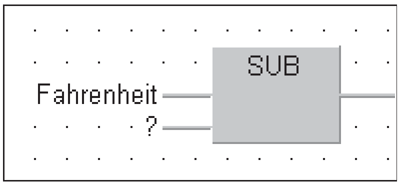
There are a variety of methods available to declare variables. The following procedure illustrates how to declare variables from the body of the FBD:

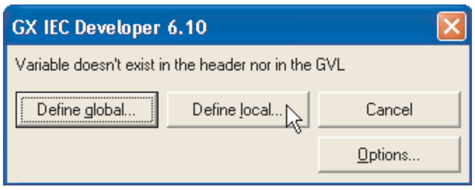
① Place input and output variables by right clicking the mouse in the work area. From the following popup menu, select and place input and output variable tags onto the FBD as shown below:



Alternatively, click on the toolbar button VAR> >VAR.

② Declare the variable "Fahrenheit" by simply typing it into the variable area:

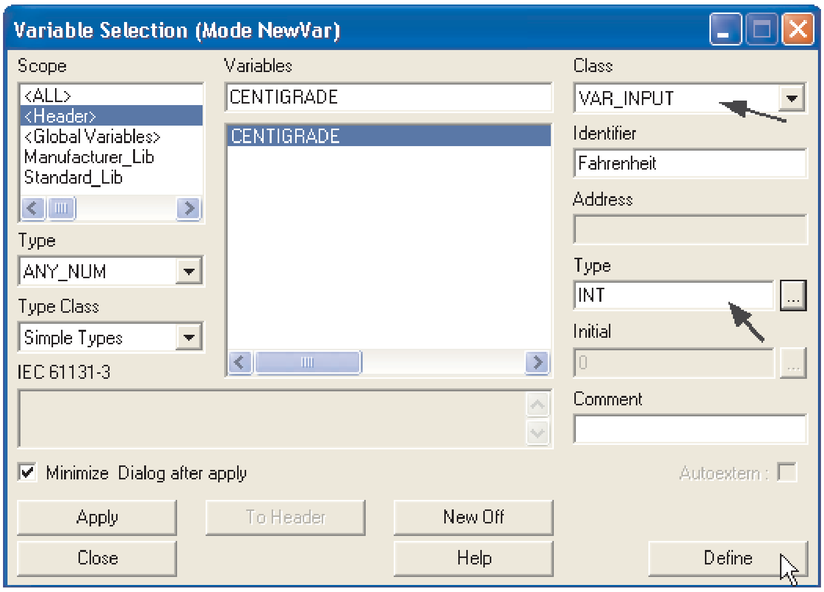



GX IEC Developer 6.10

Variable doesn't exist in the header nor in the GVL

Because this variable name has not yet been defined in the header (LVL), a prompt dialogue will be presented to choose Global or Local variable, click ***Define Local***.

③ Fill out the properties of the variable thus: Class: VAR_INPUT, Type: INT, as shown below:


Variable Selection (Mode NewVar)

Scope

<ALL>
<Header>
<Global Variables>
Manufacturer_Lib
Standard_Lib

Type: ANY_NUM

Type Class: Simple Types

IEC 61131-3

Variables: CENTIGRADE

CENTIGRADE

Class: VAR_INPUT

Identifier: Fahrenheit

Address:

Type: INT

Initial: 0

Comment:

☑ Minimize Dialog after apply

Autoextern : □

NOTES

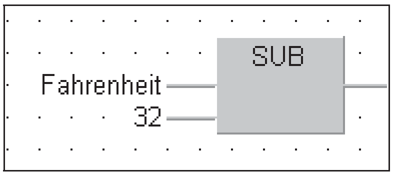
The Class VAR_INPUT is required as this variable enables values to be input into the function when it is connected as part of a program. It will produce a left hand pointing input connection point on the function symbol.

Notice also that the variable CENTIGRADE is automatically listed. This is because the “output variable name” must be the same as the “Function name”.

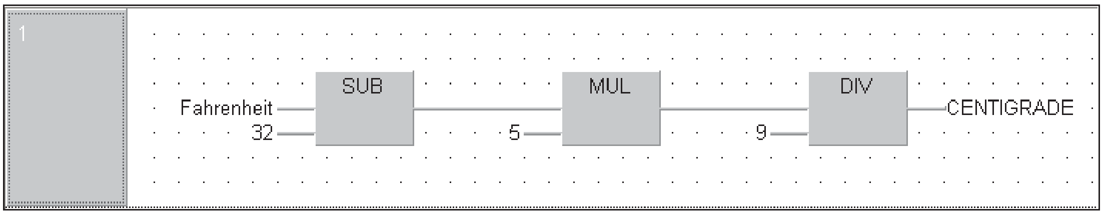
④ Click ‘Define’ and the variable will be written to the header of the Function ‘CENTIGRADE’. You can check it by opening the header.

Declaring Constants

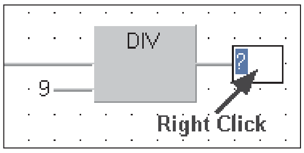
① Declare constant “32” by simply typing the number into the variable box:



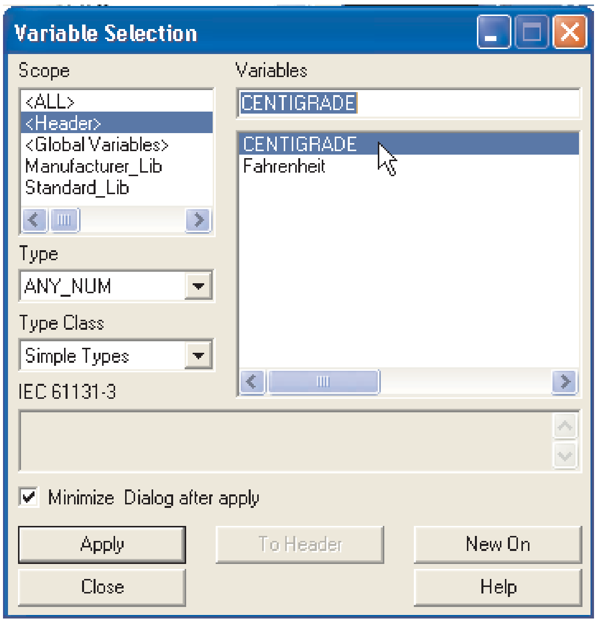
② Complete the circuit of the Function CENTIGRADE as follows:



Hint: When entering the CENTIGRADE variable, it is not necessary to type it, simply right click on the variable box (or press F2).

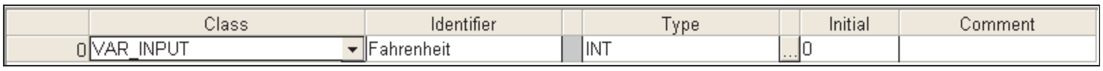


③ In the ***Variable Selection*** window, ‘Double click’ on CENTIGRADE or click to select and press ***Apply***.



CENTIGRADE is automatically placed in the header variable list as it is the name of the function, it must therefore also be specified as the output argument.

If desired, to clarify correct check the Header of the Function ‘CENTIGRADE’; it should appear as follows:



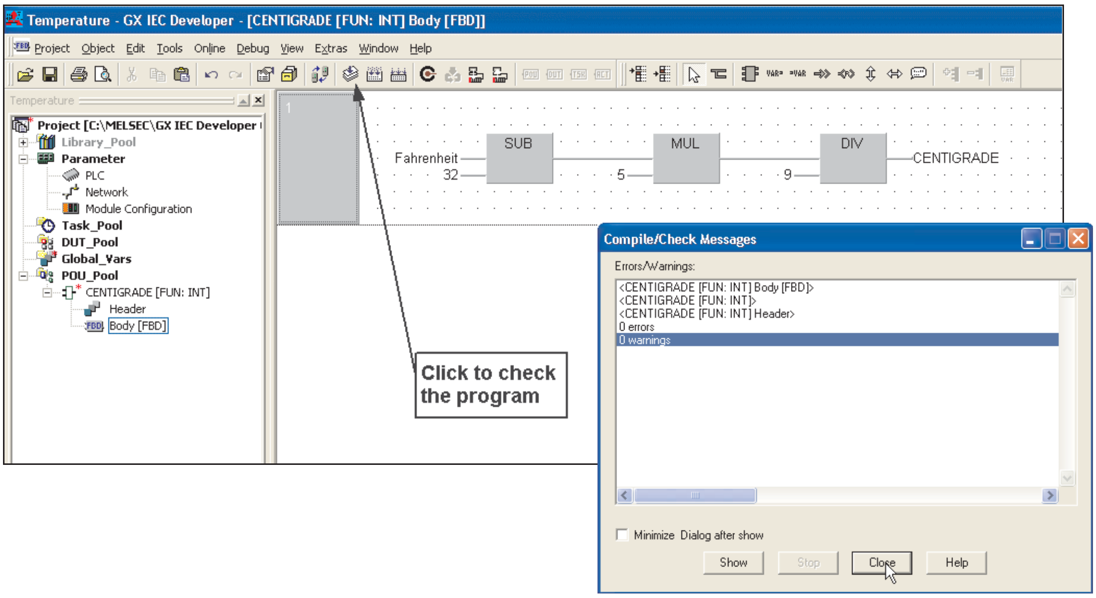
	Class	Identifier	Type	Initial	Comment
0	VAR_INPUT	Fahrenheit	INT	0	

NOTE

Alternatively, the Variable “Fahrenheit” may be entered directly into the Header (as above) and selected (F2 or right click on variable box) at point of entry in the body.

Checking Network Integrity

① Check the Network; you should have no errors and no warnings!



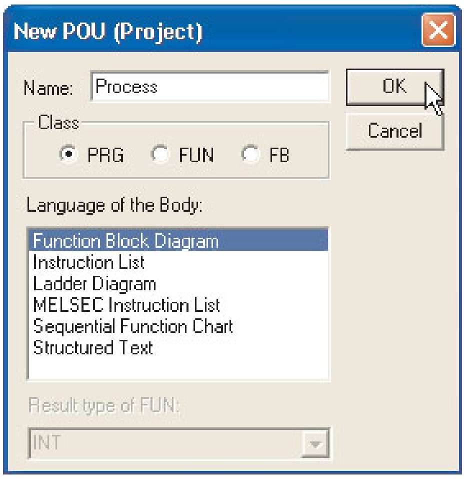
Click to check the program

```
<CENTIGRADE [FUN: INT] Body [FBD]>
<CENTIGRADE [FUN: INT]>
<CENTIGRADE [FUN: INT] Header>
0 errors
0 warnings
```

② Close down all work windows and any dialogues that may be open.

Creating a New Program POU

① Create a new POU called “Process” of Class “PRG” with a language of ***Function Block Diagram*** “FBD”:



Name: Process

Class

● PRG ○ FUN ○ FB

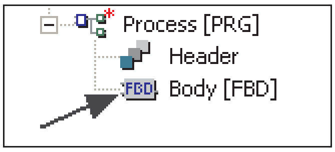
Language of the Body:

- Function Block Diagram
- Instruction List
- Ladder Diagram
- MELSEC Instruction List
- Sequential Function Chart
- Structured Text

Result type of FUN:

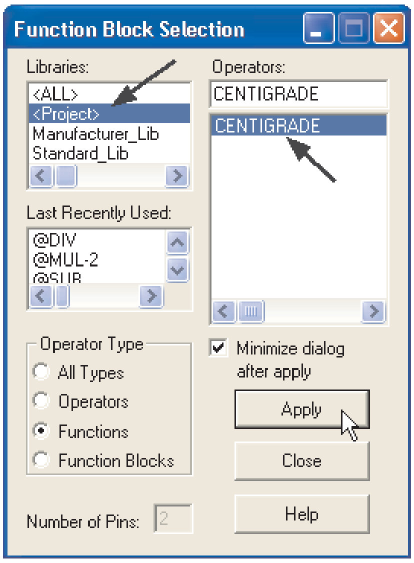
INT

② Open up (Double Click) the body of Ladder POU “Process” in the project POU pool.



Placing a user Function

① Click on the Function Block icon ▤ again, but this time select ***Functions*** and select the ***Project Library***. Notice the newly created function “Centigrade” is now filtered down into the operators list:


Function Block Selection

Libraries: <ALL>, <Project>, Manufacturer_Lib, Standard_Lib

Operators: CENTIGRADE

Last Recently Used: @DIV, @MUL-2, @SUB

Operator Type: ○ All Types ○ Operators ● Functions ○ Function Blocks

☑ Minimize dialog after apply

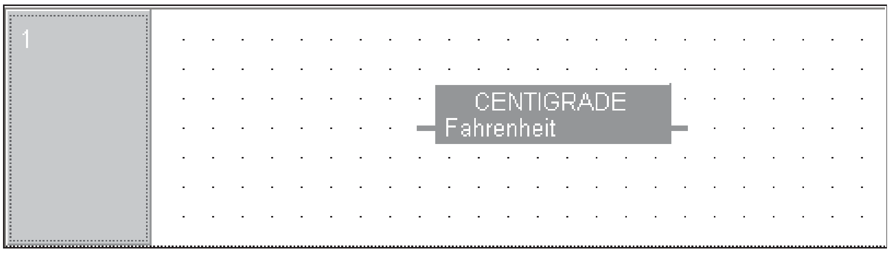
Number of Pins: 2

② Select CENTIGRADE and click ‘Apply’.

NOTE

Depending on preference, it is possible to minimise the ***Function Block selection*** window following ***Apply*** by ticking the selection box as above.

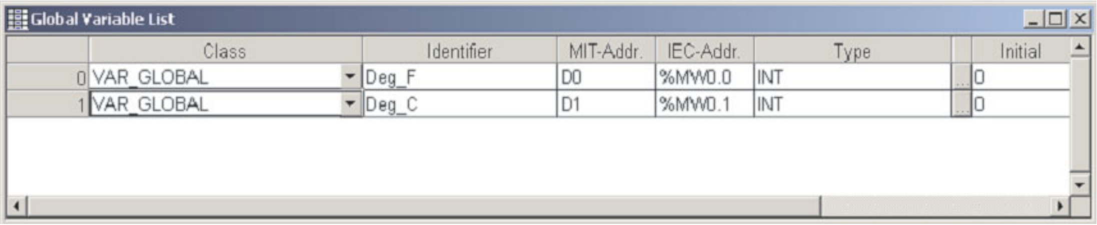
The following will be displayed:



Assigning the Global Variables

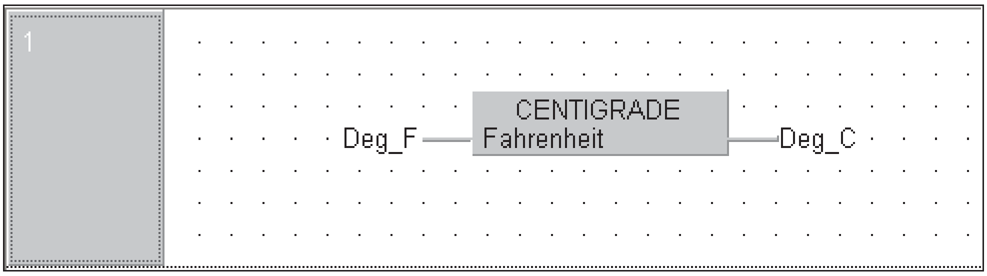
Once the function is placed on the new network assign variables to it.

① Assign Variable names in the Global Variable List as shown:

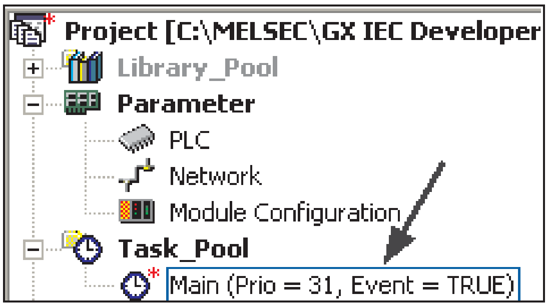

Global Variable List

	Class	Identifier	MIT-Addr.	IEC-Addr.	Type	Initial
0	VAR_GLOBAL	Deg_F	D0	%MW0.0	INT	0
1	VAR_GLOBAL	Deg_C	D1	%MW0.1	INT	0

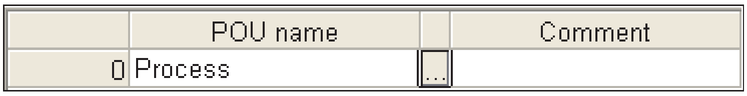
The Body of the POU “Process" should read:



② Create a new task in the ***Task Pool*** named “Main”.


Project [C:\MELSEC\GX IEC Developer
Library_Pool
Parameter
PLC
Network
Module Configuration
Task_Pool
Main (Prio = 31, Event = TRUE)

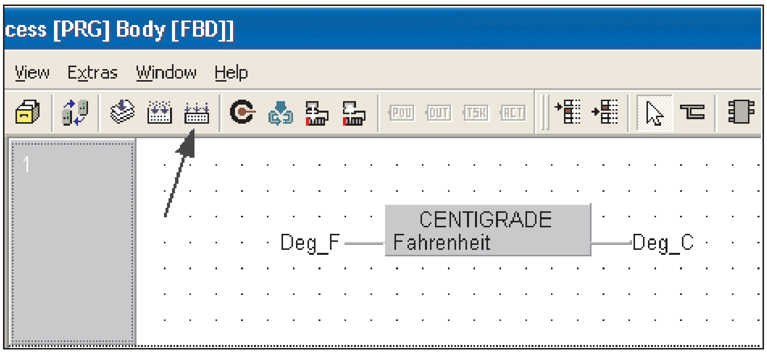
③ Bind the POU “Process” to the Task “Main”:



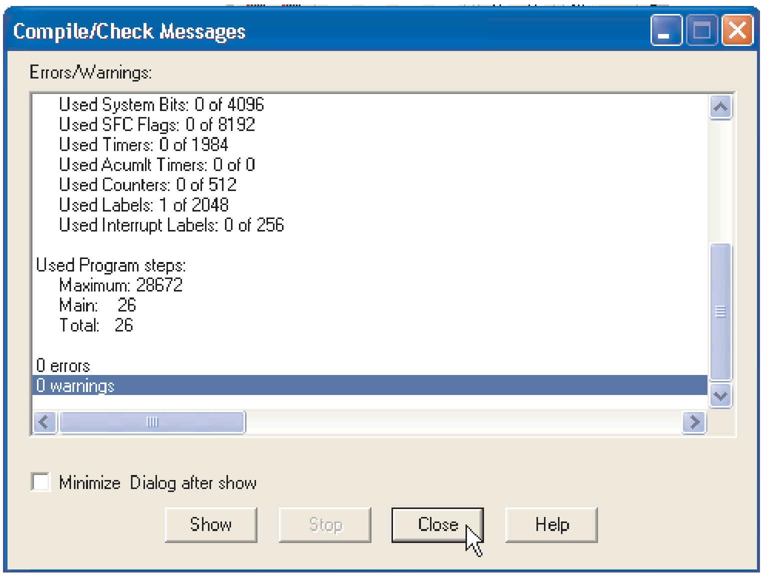
	POU name		Comment
0	Process	...	

Compiling the Program

Compile the project using the ***Rebuild All*** operation from the tool bar:



Following compilation the following should be displayed:



Compile/Check Messages

Errors/Warnings:

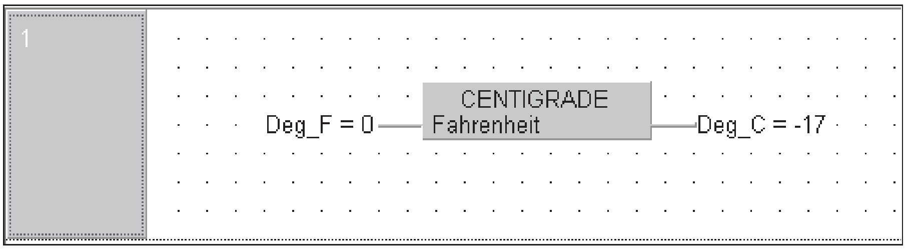
```
    Used System Bits: 0 of 4096
    Used SFC Flags: 0 of 8192
    Used Timers: 0 of 1984
    Used Acumlt Timers: 0 of 0
    Used Counters: 0 of 512
    Used Labels: 1 of 2048
    Used Interrupt Labels: 0 of 256

Used Program steps:
    Maximum: 28672
    Main:    26
    Total:   26

0 errors
0 warnings
```

□ Minimize Dialog after show

If there are errors, click on the error detail and resolve the problem(s).

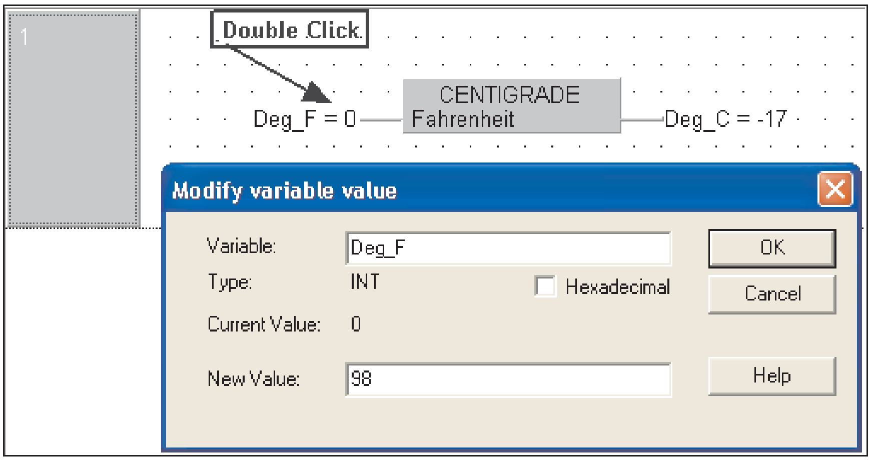
Monitoring the program

① Transfer the project to the PLC and monitor this network using the Monitor button on the toolbar:



② Using the on screen variable forcing feature, input numbers into the ‘Deg_F’ variable as follows:

‘Double Click’ on the input variable and enter a value into the ***Modify variable value*** dialogue as shown:



Modify variable value

Variable: Deg_F

Type: INT □ Hexadecimal

Current Value: 0

New Value: 98

For reference, 100 deg F = 37 deg C (actual 37.7 deg C)

6.1.2 Processing Real (Floating Point) Numbers

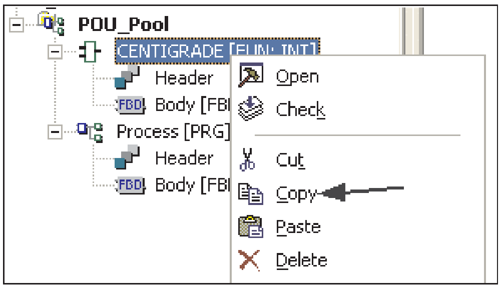
The existing CENTIGRADE function currently can only process 16 Bit Integer Whole Number (-32768 to +32767) values which is the numeric system default when creating Functions. The following example will utilise the Function ‘CENTIGRADE’, modifying it to process “REAL” floating point values*.

* Only valid on base units supporting this feature.

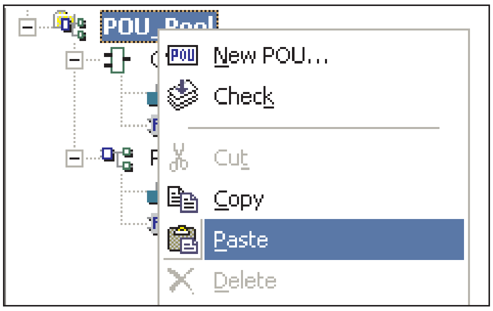
Duplicating a Function

Make a duplicate copy of the function ‘CENTIGRADE’ and rename it ‘CENTIGRADE1’ as follows:

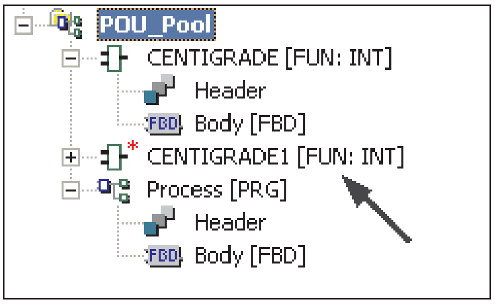
① Right Click on the CENTIGRADE Icon in the POU Pool of the project and select ***Copy***.



② Right Click on the POU pool icon of the project and select ***Paste***.

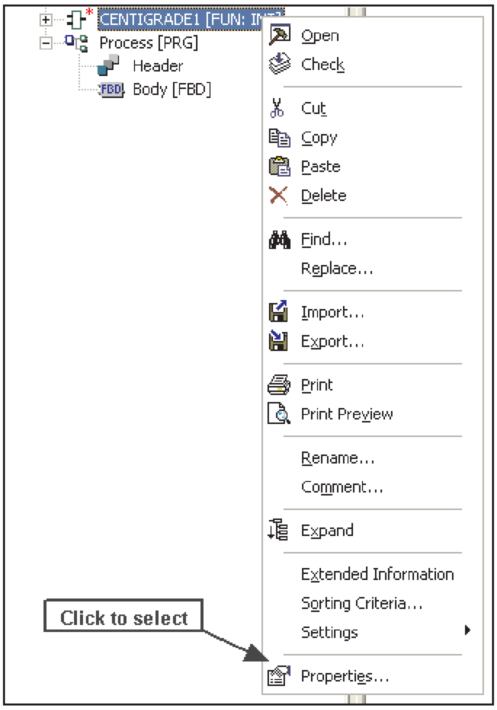


The system will automatically paste a duplicate copy of ‘CENTIGRADE’ and rename it to ‘CENTIGRADE1’:

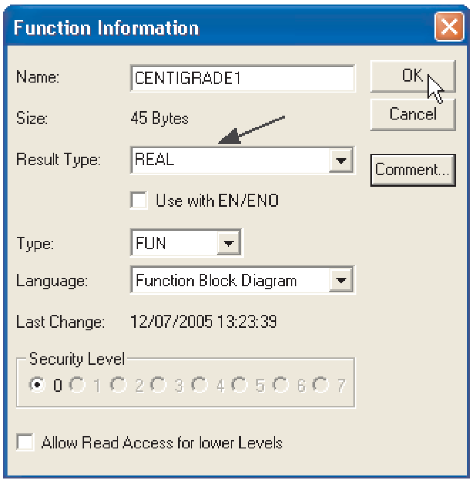


Changing the Result type of a Function

① Right click on the newly created Function ‘CENTIGRADE1’ and click on ***Properties***.



② On displaying the ***Function Information*** window, set the result type to REAL.



Name: CENTIGRADE1

Size: 45 Bytes

Result Type: REAL

□ Use with EN/ENO

Type: FUN

Language: Function Block Diagram

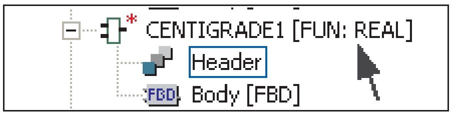
Last Change: 12/07/2005 13:23:39

Security Level

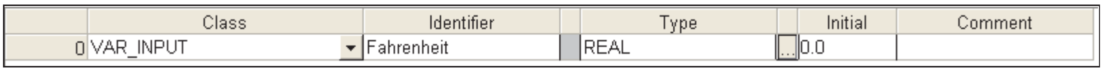
● 0 ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7

□ Allow Read Access for lower Levels

The type should now displayed as ***Real*** in the Project Navigation Window:



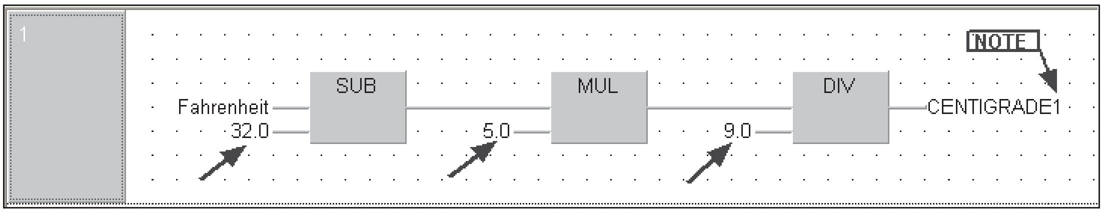
③ Modify the Header of CENTIGRADE1 so that the Fahrenheit variable is of type ‘REAL’:



	Class	Identifier	Type	Initial	Comment
0	VAR_INPUT	Fahrenheit	REAL	0.0	

Modifying Constants to type 'REAL'

① Open the Body of CENTIGRADE1 and modify the constants to 'Floating Point' types (i.e. 32.0) and the output variable name to read as follows:

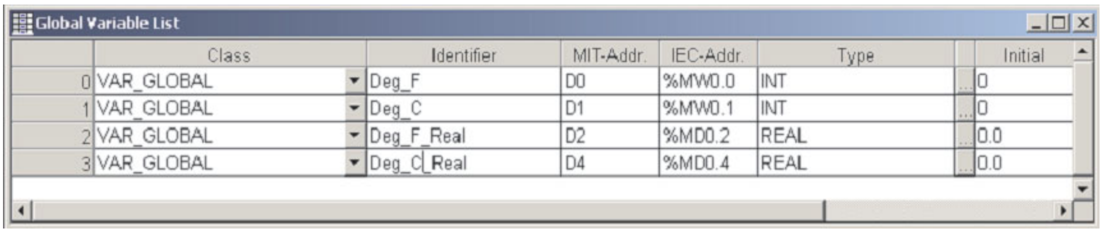


NB: Remember to alter CENTIGRADE to CENTIGRADE1.

② Close editors and save all changes.

Placing the "REAL" number Function 'CENTIGRADE1' onto the working POU "Process"

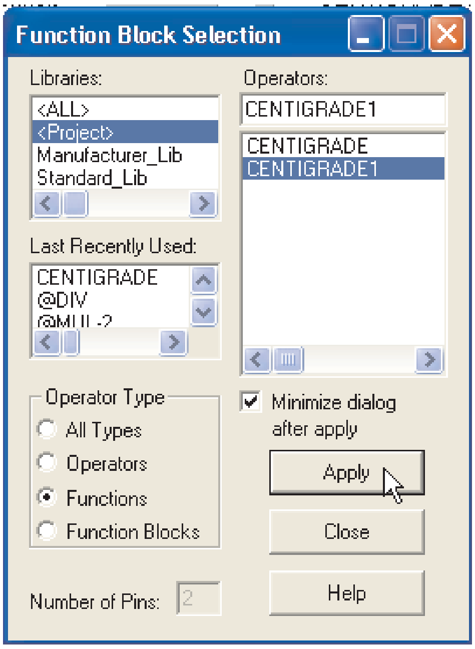
① In the GVL editor, create two new variables thus:



Global Variable List

	Class	Identifier	MIT-Addr.	IEC-Addr.	Type	Initial
0	VAR_GLOBAL	Deg_F	D0	%MW0.0	INT	0
1	VAR_GLOBAL	Deg_C	D1	%MW0.1	INT	0
2	VAR_GLOBAL	Deg_F_Real	D2	%MD0.2	REAL	0.0
3	VAR_GLOBAL	Deg_C_Real	D4	%MD0.4	REAL	0.0

② Open the Body of POU "Process" and place the Function CENTIGRADE1 into it as shown below:



Function Block Selection

Libraries:
<ALL>
<Project>
Manufacturer_Lib
Standard_Lib

Operators: CENTIGRADE1

CENTIGRADE
CENTIGRADE1

Last Recently Used: CENTIGRADE, @DIV, @MUL-2

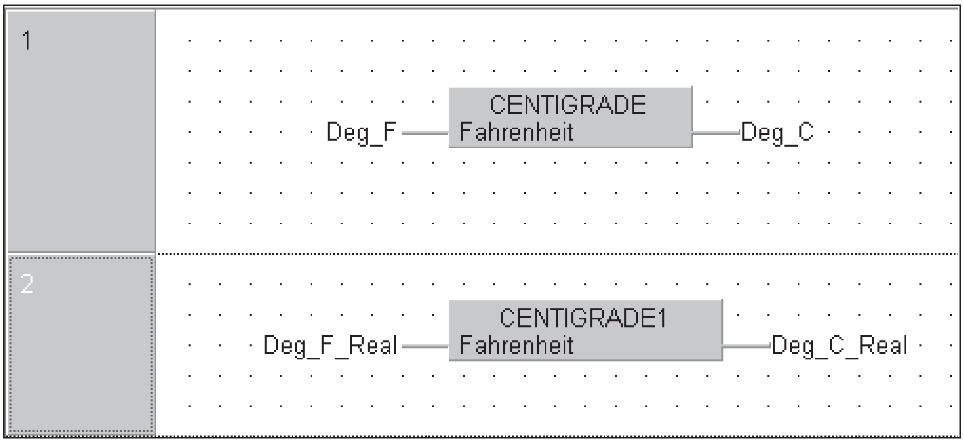
Operator Type
○ All Types
○ Operators
● Functions
○ Function Blocks

☑ Minimize dialog after apply

Number of Pins: 2

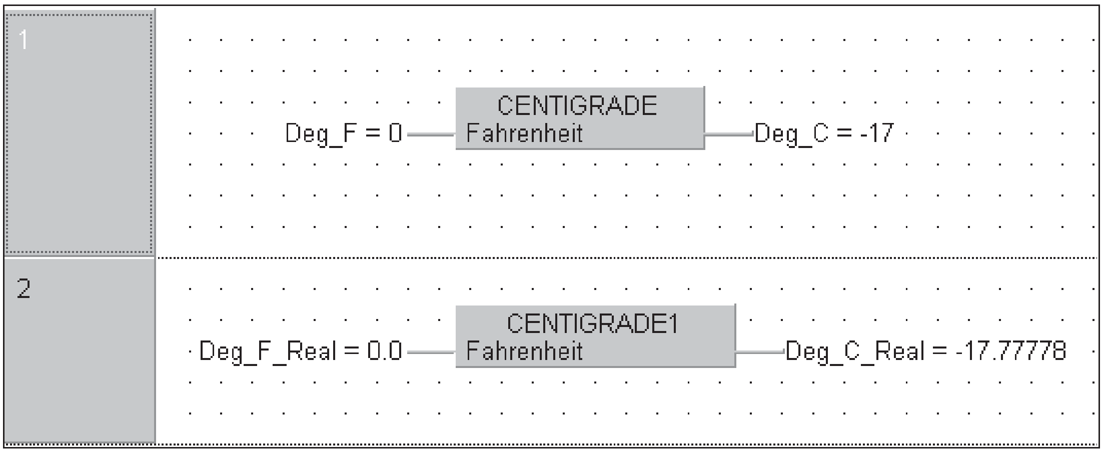
NOTE REAL numbers use 2 consecutive Registers (32 Bits) and are stored in a special portable IEE format, hence the allocation in the above GVL example.

③ Complete the POU “Process” to read as follows:



Save the Project, Close all open dialogues and rebuild the project.

Transfer the project to the PLC and monitor this network using the Monitor button on the toolbar:



Modify the value of the input variable “Deg_F_Real” and observe the output result on the display. Note the 7 Digit floating point accuracy.

6.2 Creating a Function Block

Objective:

Build a Function Block to act as a Star/Delta Starter. Declare the following variables:

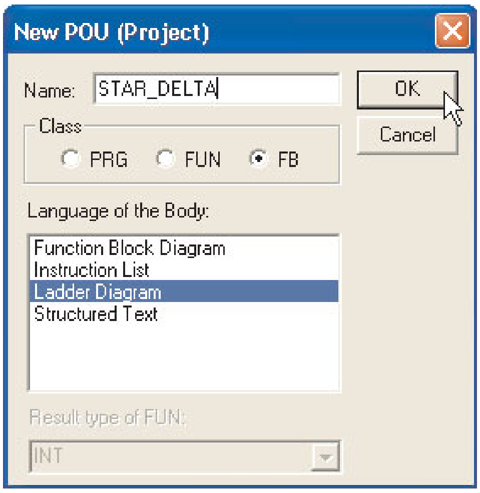
– Start Pushbutton:	**START**
– Stop Pushbutton:	**STOP**
– Overload Contact:	**OVERLOAD**
– Switchover Time:	**TIMEBASE**
– Time Register:	**TIME_COIL**
– Star Contactor Output:	**STAR_COIL**
– Delta Contactor Output:	**DELTA_COIL**

Name the Function Block **STAR_DELTA.**

Procedure:

① Start a new "Empty" project in GX IEC Developer called "**Motor Control**" with no POU's.

② Create a new POU named "STAR_DELTA" of Class "Function Block" (***FB***) with a language Body type ***Ladder Diagram***.


New POU (Project)

Name: STAR_DELTA

Class ○ PRG ○ FUN ● FB

Language of the Body:
Function Block Diagram
Instruction List
Ladder Diagram
Structured Text

Result type of FUN: INT

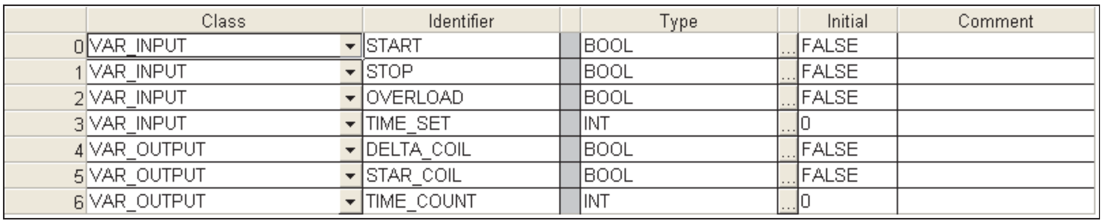
STAR_DELTA will now have appeared on the POU tree.

③ Click once to open the Header and Body branches.

④ Double click, to open the Header.

Declaring Local Variables

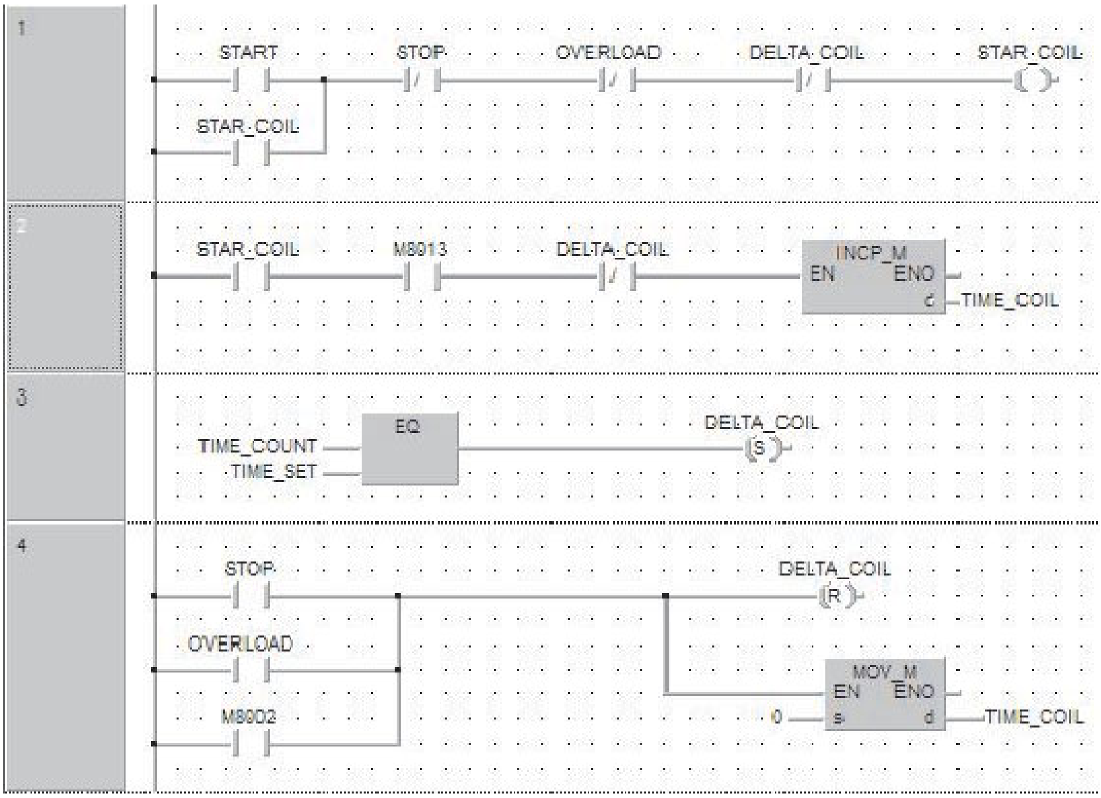
① Declare variables as shown below.



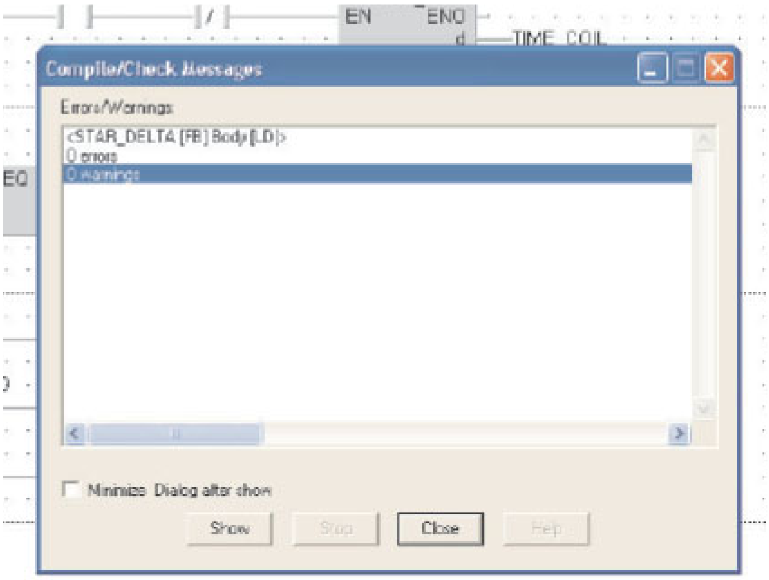
	Class	Identifier	Type	Initial	Comment
0	VAR_INPUT	START	BOOL	FALSE	
1	VAR_INPUT	STOP	BOOL	FALSE	
2	VAR_INPUT	OVERLOAD	BOOL	FALSE	
3	VAR_INPUT	TIME_SET	INT	0	
4	VAR_OUTPUT	DELTA_COIL	BOOL	FALSE	
5	VAR_OUTPUT	STAR_COIL	BOOL	FALSE	
6	VAR_OUTPUT	TIME_COUNT	INT	0	

② Check, save and then close the Header window.

③ Open the body and build the ladder networks as shown below:



④ Check the Body, there should be no errors and no warnings!


Compile/Check Messages

Errors/Warnings

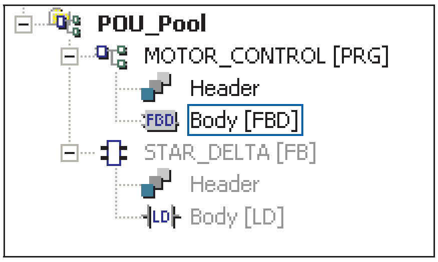
<STAR_DELTA [FB] Body [LD]>
0 errors
0 warnings

□ Minimize Dialog after show

Creating New Program POU “Motor Control”

① Close down all work windows and any dialogues that may be open.

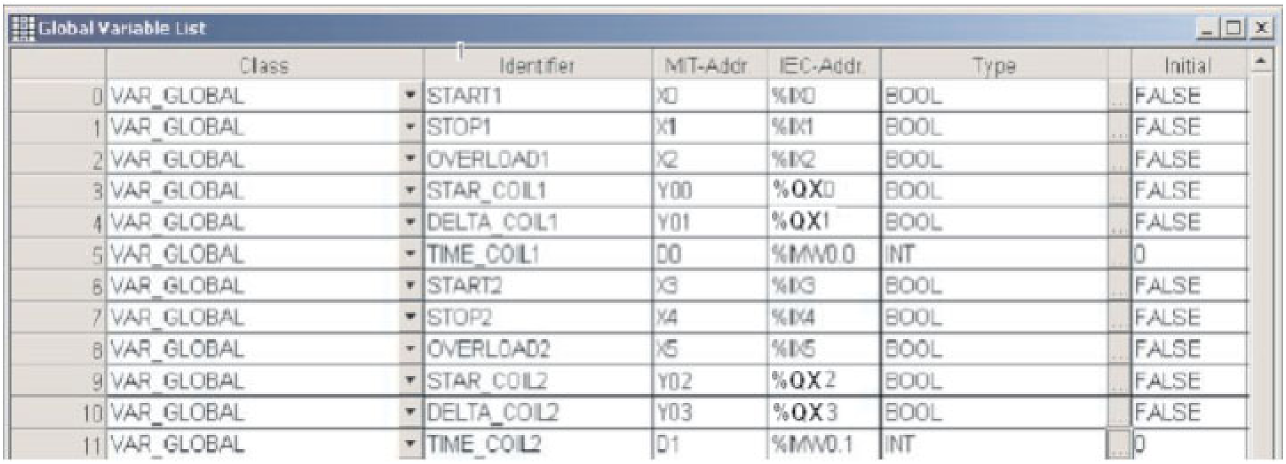
② Create a new POU “MOTOR_CONTROL” of Class ***PRG*** and FBD (***Function Block Diagram***) as the language of the body.



- POU_Pool
  - MOTOR_CONTROL [PRG]
    - Header
    - Body [FBD]
  - STAR_DELTA [FB]
    - Header
    - Body [LD]

Creating new Global Variables List

Open the GVL and enter the following I/O details:

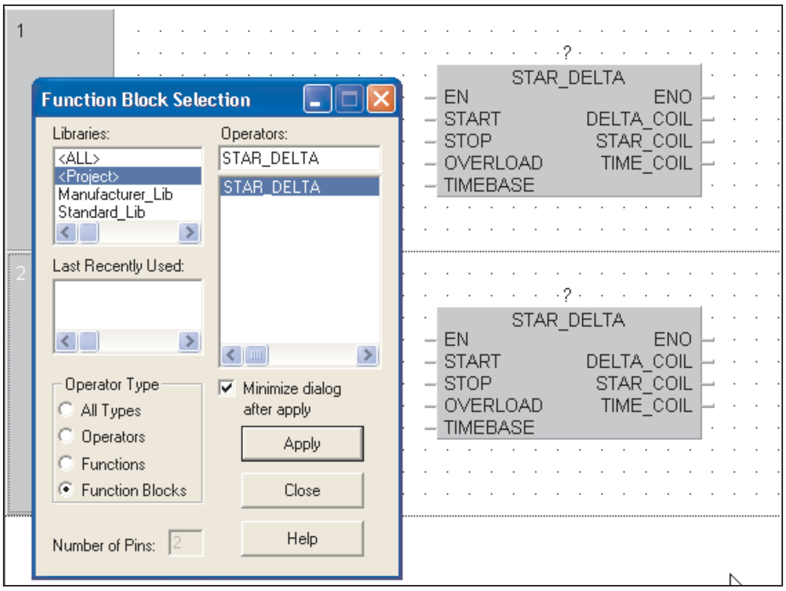


Global Variable List

	Class	Identifier	MIT-Addr	IEC-Addr	Type	Initial
0	VAR_GLOBAL	START1	X0	%IX0	BOOL	FALSE
1	VAR_GLOBAL	STOP1	X1	%IX1	BOOL	FALSE
2	VAR_GLOBAL	OVERLOAD1	X2	%IX2	BOOL	FALSE
3	VAR_GLOBAL	STAR_COIL1	Y00	%QX0	BOOL	FALSE
4	VAR_GLOBAL	DELTA_COIL1	Y01	%QX1	BOOL	FALSE
5	VAR_GLOBAL	TIME_COIL1	D0	%MW0.0	INT	0
6	VAR_GLOBAL	START2	X3	%IX3	BOOL	FALSE
7	VAR_GLOBAL	STOP2	X4	%IX4	BOOL	FALSE
8	VAR_GLOBAL	OVERLOAD2	X5	%IX5	BOOL	FALSE
9	VAR_GLOBAL	STAR_COIL2	Y02	%QX2	BOOL	FALSE
10	VAR_GLOBAL	DELTA_COIL2	Y03	%QX3	BOOL	FALSE
11	VAR_GLOBAL	TIME_COIL2	D1	%MW0.1	INT	0

Assigning Instance Names

① Open the Body of MOTOR_CONTROL and enter create two networks. Place a Instance of the Function Block STAR_DELTA into each network as shown in the following figure:



Function Block Selection

Libraries: <ALL>, <Project>, Manufacturer_Lib, Standard_Lib

Last Recently Used:

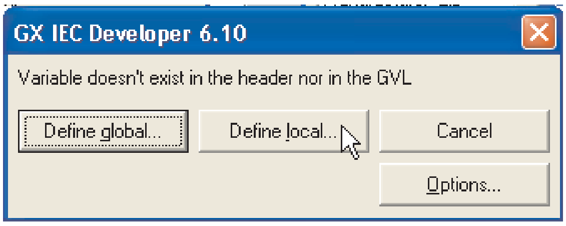
Operators: STAR_DELTA

Operator Type: ○ All Types ○ Operators ○ Functions ● Function Blocks

☑ Minimize dialog after apply

Number of Pins: 2

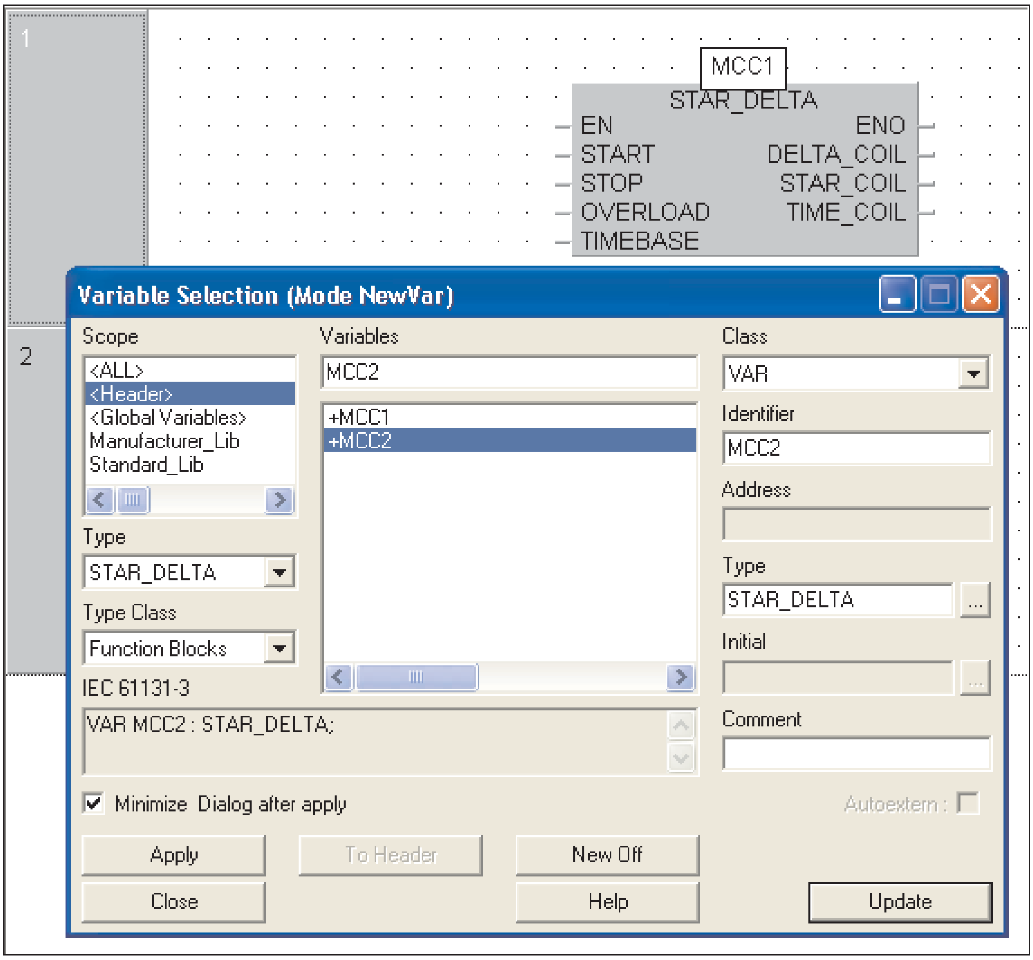
② Assign 'instance names' to both instances of the Function Block, STAR_DELTA by typing MCC1 and MCC2 into the Instance names above each Instance of the FB. At the system prompt, click ***Define Local***.



GX IEC Developer 6.10

Variable doesn't exist in the header nor in the GVL

③ Create entries for the instance names in the header for MCC1 and MCC2 as follows:



Variable Selection (Mode NewVar)

Scope

<ALL>
<Header>
<Global Variables>
Manufacturer_Lib
Standard_Lib

Variables: MCC2

+MCC1
+MCC2

Class: VAR

Identifier: MCC2

Address:

Type: STAR_DELTA

Type Class: Function Blocks

Type: STAR_DELTA

Initial:

IEC 61131-3

VAR MCC2 : STAR_DELTA;

Comment:

☑ Minimize Dialog after apply

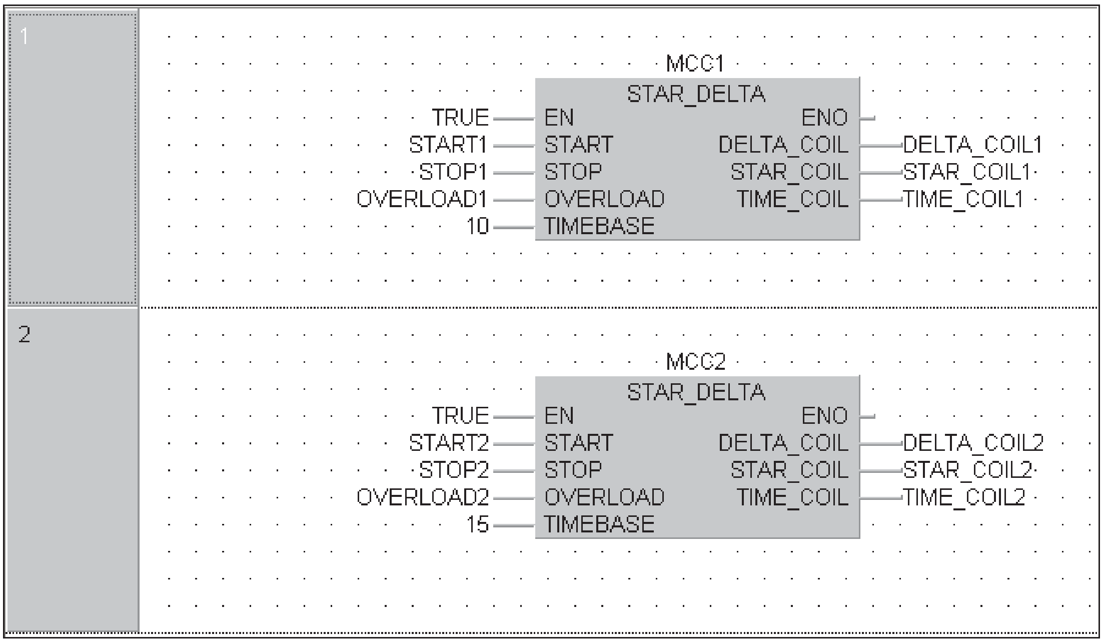
Autoextern : □

An Instance is the copy of the function block for this POU. For this example simply type MCC1 and MCC2. Notice that once entered, the instances are listed in the variable selection window as +MCC1 and +MCC2 as Type: STAR_DELTA.

The Instances must be declared in the POU Header. As can be seen from the previous figures, Instance names are added in the same way as adding any other new variable from the POU body.

Assigning Variables to a Function Block

Now complete the POU by assigning variables to your Function Blocks as shown below:



NOTES

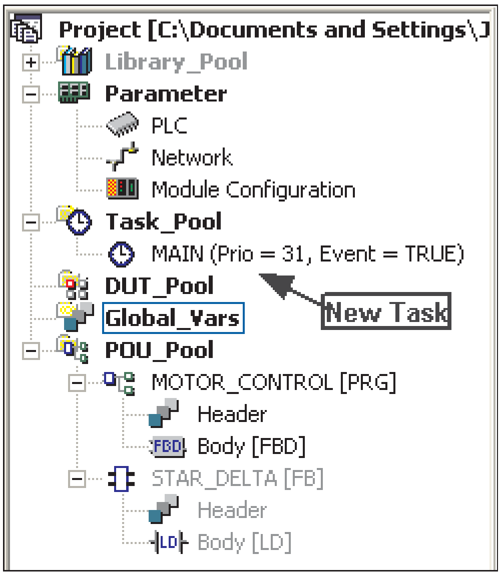
Mitsubishi addresses or symbolic declarations may be used. However, if Mitsubishi ‘MELSEC’ direct addresses are used then the program will no longer adhere to the IEC conventions.

Designating the variable “TRUE” as above, automatically assigns a ‘normally on’ contact (Special relay M8000 in MELSEC FX series) which is neater and conforms to IEC conventions.

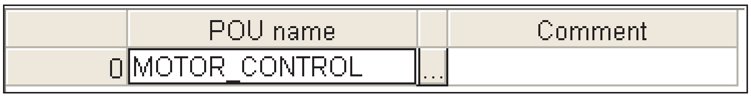
The STAR_DELTA FB can be used many times in the project and must use different Instance names.

Creating a New Task:

① Create a new Task “MAIN” in the task pool:



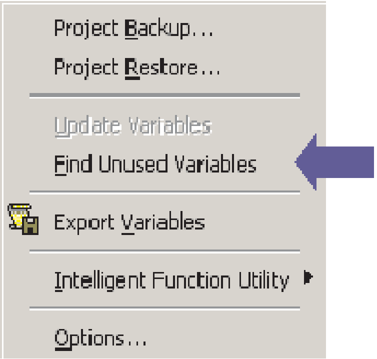
② Double click on the task and bind the POU “MOTOR_CONTROL” to the task “MAIN”:



	POU name		Comment
0	MOTOR_CONTROL	...	

③ Save the Program, close all windows and dialogues.

Find unused Variables



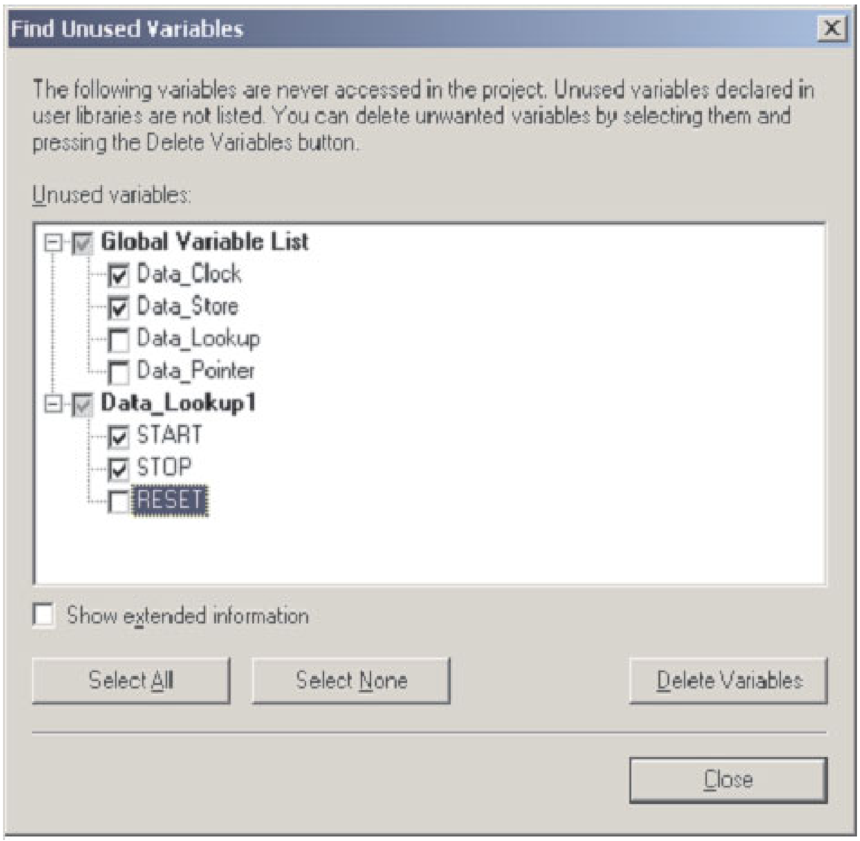
By using the function ***Extras*** → ***Find unused Variables*** you can find and delete all unused global and local variables that are declared but not used in a project.

Unused global and local variables will be detected in the whole project, excluding the user libraries.

NOTE

Finding unused variables can only be performed if the project has been build and was not changed since them. Otherwise a warning message will be displayed.

Each unused variable is listed under the container of its declaration: the Global Variable List for global variables, or the corresponding POU for local variables. Only those containers are listed where unused variables exist. For example, if there is no global variable, the Global Variable List location will not be enlisted. Containers are written in bold text and appear at a higher level than their contained items.



Find Unused Variables

The following variables are never accessed in the project. Unused variables declared in user libraries are not listed. You can delete unwanted variables by selecting them and pressing the Delete Variables button.

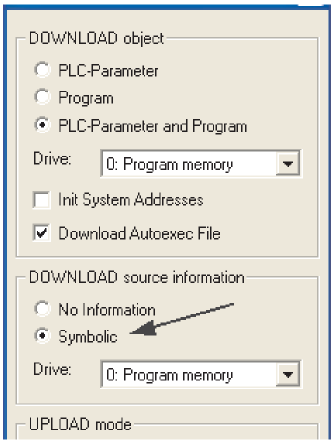
Unused variables:

- ☑ **Global Variable List**
  - ☑ Data_Clock
  - ☑ Data_Store
  - □ Data_Lookup
  - □ Data_Pointer
- ☑ **Data_Lookup1**
  - ☑ START
  - ☑ STOP
  - □ RESET

□ Show extended information

NOTE

This can produce large reductions in the size of the source code. This is important particularly if the option to send all ***Symbolic*** (Source) Code to the PLC has been selected for download:

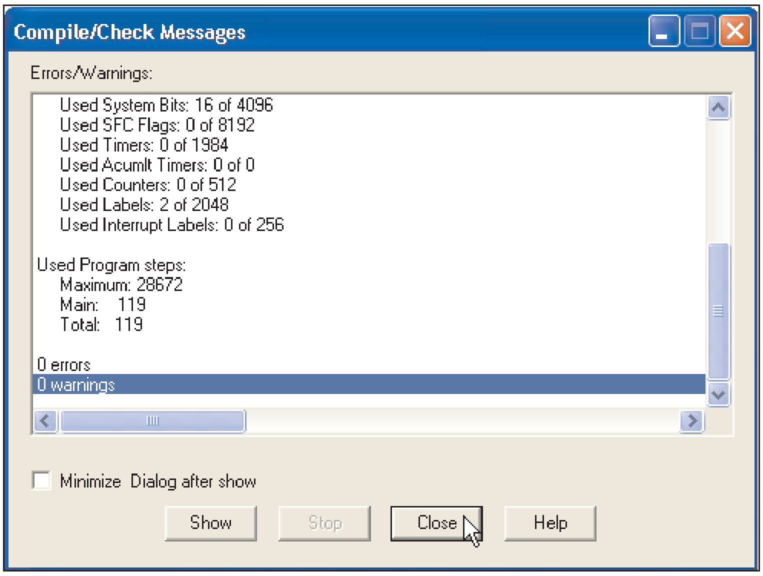


DOWNLOAD object
- ○ PLC-Parameter
- ○ Program
- ● PLC-Parameter and Program
- Drive: 0: Program memory
- □ Init System Addresses
- ☑ Download Autoexec File

DOWNLOAD source information
- ○ No Information
- ● Symbolic
- Drive: 0: Program memory

UPLOAD mode

Compile the program in the normal manner, using the “Rebuild All” button on the toolbar:



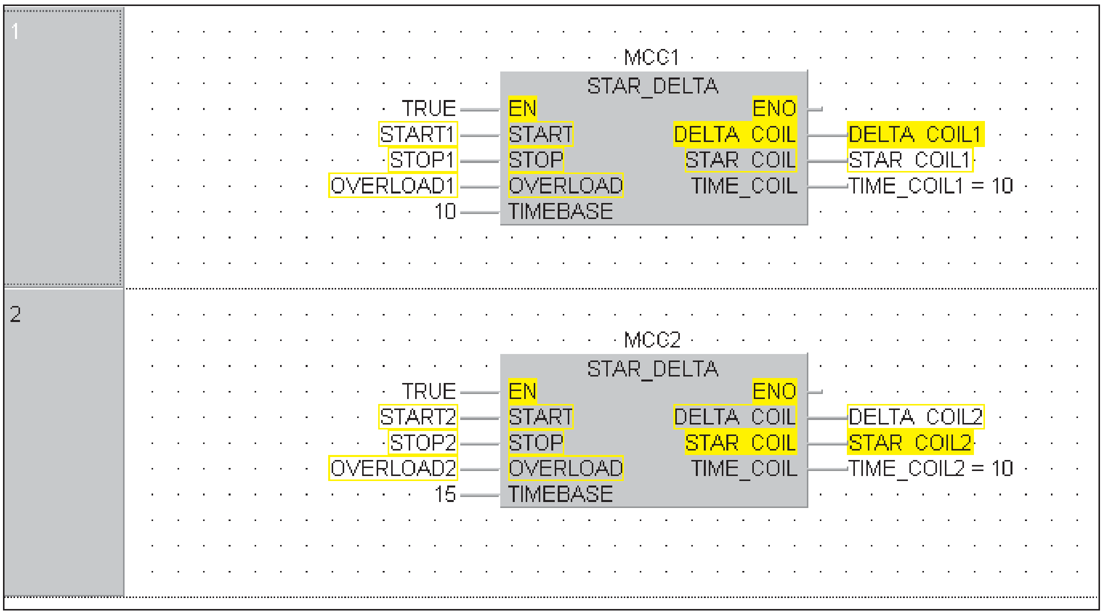
Compile/Check Messages

Errors/Warnings:

```
    Used System Bits: 16 of 4096
    Used SFC Flags: 0 of 8192
    Used Timers: 0 of 1984
    Used Acumlt Timers: 0 of 0
    Used Counters: 0 of 512
    Used Labels: 2 of 2048
    Used Interrupt Labels: 0 of 256

Used Program steps:
    Maximum: 28672
    Main:    119
    Total:   119

0 errors
0 warnings
```

□ Minimize Dialog after show

Open the MOTOR_CONTROL POU and monitor the program for correct operation.

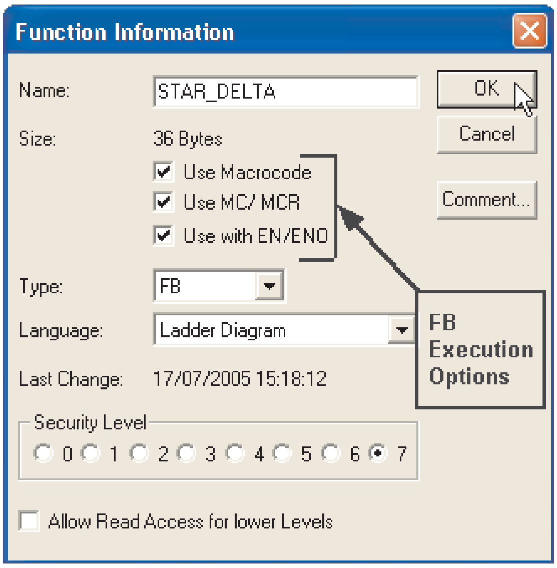


6.3 Execution Options of Function Blocks

Function blocks can be executed in different ways:

- Macrocode execution
- MC – MCR execution
- Use with EN/ENO

The execution mode is selected in the ***Function Information*** dialogue box:



Function Information

Name: STAR_DELTA

Size: 36 Bytes

☑ Use Macrocode

☑ Use MC/ MCR

☑ Use with EN/ENO

Type: FB

Language: Ladder Diagram

Last Change: 17/07/2005 15:18:12

Security Level: ○ 0 ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ● 7

□ Allow Read Access for lower Levels

How to set the execution option:

① Select the function block in the Project Navigator window.

② Display the Function Information dialogue box by right clicking and select ***Properties***.

③ Activate the check box. The use of MC-MCR option can only be activated when the other two options have already been activated.

This does not make any changes to instantiation and the programming of instances in the various programming languages.

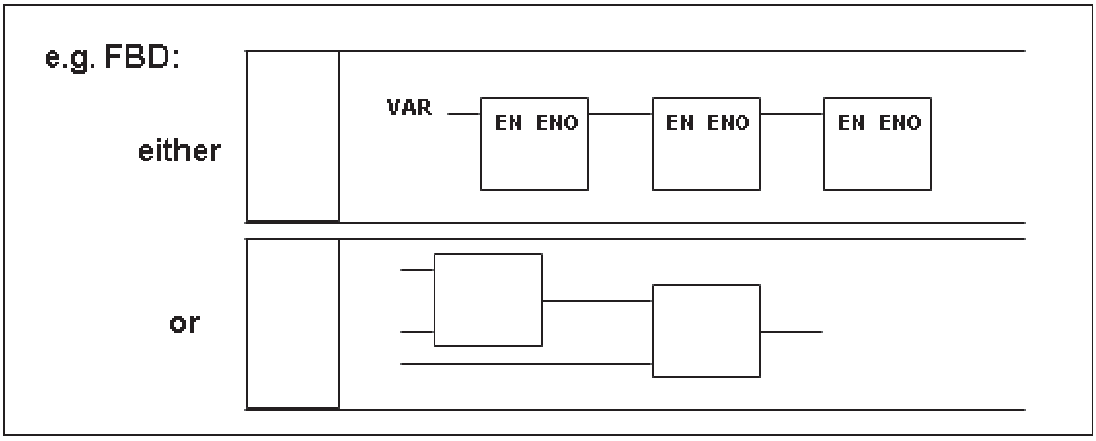
6.3.1 Macrocode execution

- Standard execution: The function block is called via a system label.
- Macrocode execution: The function block is expanded internally.

With Macro Code	Without Macro Code (standard execution)
No internal system labels are needed to execute a function block instance.	Each instance uses internal system labels (pointers).
Consequence: The number of function blocks you can use is only limited by the size of the PLC memory as function blocks are independent of system labels.	Consequence: Since the number of available system labels is limited (FX: 128, A: 256, Q: 1024) you cannot use more than a theoretical limited number of function blocks. In practice this number is even smaller as system labels are also required for other internal processes.
User-oriented execution of the function block	Implementation of the function block construct in conformity with the IEC 61131-3 standard
No restrictions on the handling of timers and coils within the function block.	Restrictions on the handling of timers and coils within the function block (subroutines).

6.3.2 Enable / EnableOutput (EN/ENO)

- The EN input makes the function (or FB, see later), conditional (Switch On/Off)
- The ENO reflects the status of the EN line.
- Only instructions with or without EN should be used in a network, do not mix both types.
- The EN/ENO chain should have all its pre-conditions at the beginning:

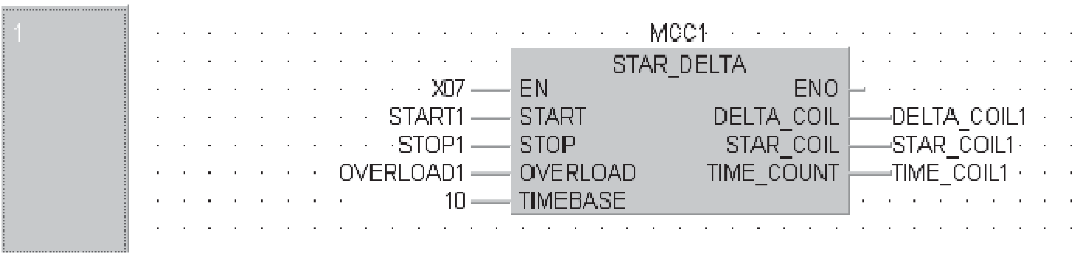


Function Definitions

- All devices suffixed "_E" have EN / ENO lines, otherwise they do not.
- All devices suffixed "_M" are manufacturers instructions, i.e. in this case from the relevant Mitsubishi instruction set.

Exercise (Gated Operation)

Edit the Function Block STAR_DELTA to have an EN/ENO input/output feature. Drive the EN (enable) input with external MELSEC X07 contact:

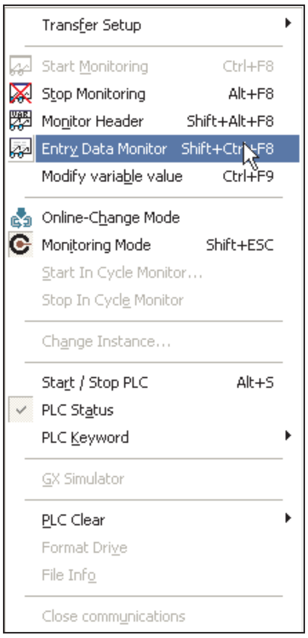


7 Advanced Monitoring Functions

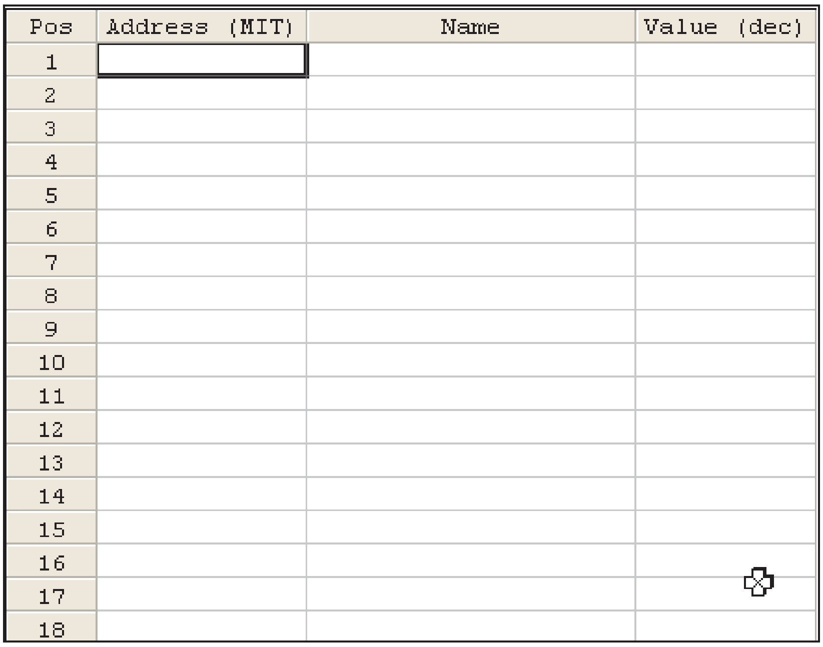
The following diagrams are used for illustration purposes only; use the STAR_DELTA project and its relevant devices with the following procedures.

7.1 Entry Data Monitoring

① Whilst in Monitor Mode, select ***Entry Data Monitor*** from the ***Online*** Menu:

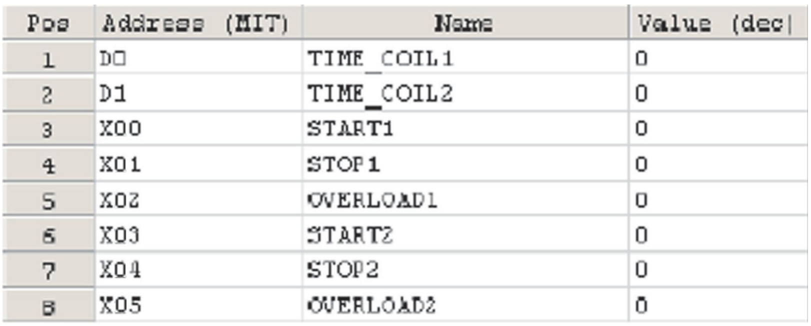


The following table will be displayed:



Pos	Address (MIT)	Name	Value (dec)
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			

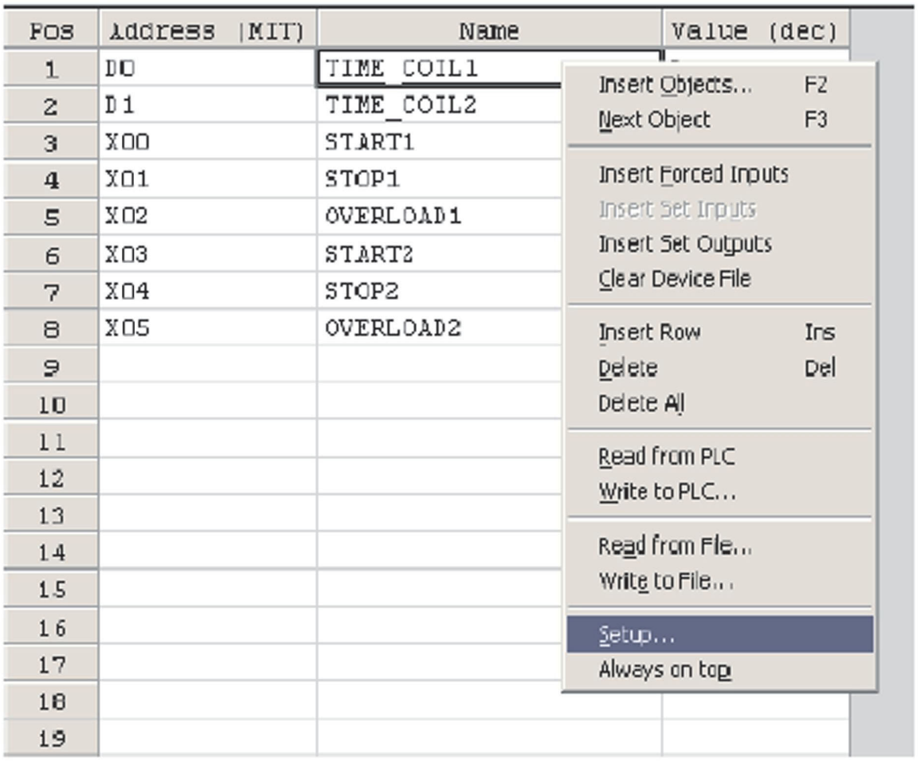
② Click in the Mitsubishi Address left hand column and type in the required device, any identifier name will be automatically shown together with the current value. Column widths can be altered. In the head of the table, move the cursor over the left border of the column you want to alter. Then press the left mouse button and move the border to the left or right. Release the left mouse button at the desired position.



Pos	Address (MIT)	Name	Value (dec)
1	D0	TIME_COIL1	0
2	D1	TIME_COIL2	0
3	X00	START1	0
4	X01	STOP1	0
5	X02	OVERLOAD1	0
6	X03	START2	0
7	X04	STOP2	0
8	X05	OVERLOAD2	0

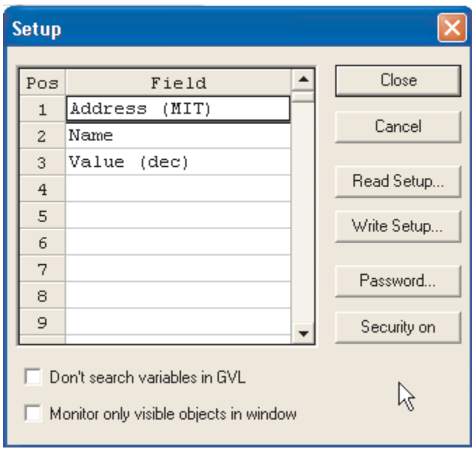
7.1.1 Customising the EDM

① Right Clicking the mouse button, displays the following window. Select ***Setup***.



Pos	Address (MIT)	Name	Value (dec)
1	D0	TIME_COIL1	
2	D1	TIME_COIL2	
3	X00	START1	
4	X01	STOP1	
5	X02	OVERLOAD1	
6	X03	START2	
7	X04	STOP2	
8	X05	OVERLOAD2	
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			

The ***Setup*** window allows the EDM to be user configurable; clicking the right mouse button, displays the configurator window. In this procedure Columns will be added to the EDM table for IEC Address and Hex Value Monitor.



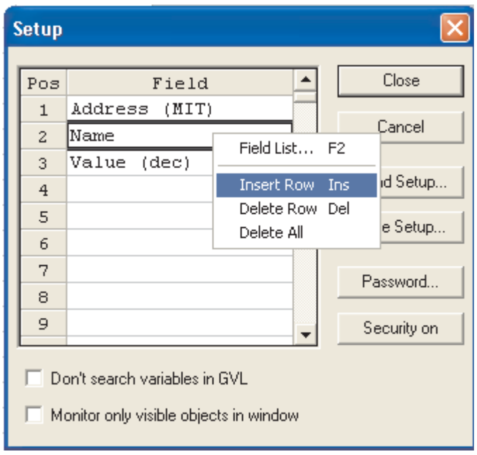
Setup

Pos	Field
1	Address (MIT)
2	Name
3	Value (dec)
4	
5	
6	
7	
8	
9	

□ Don't search variables in GVL

□ Monitor only visible objects in window

② Highlight or right click on the ***Name*** field and select ***Insert Row*** as shown.



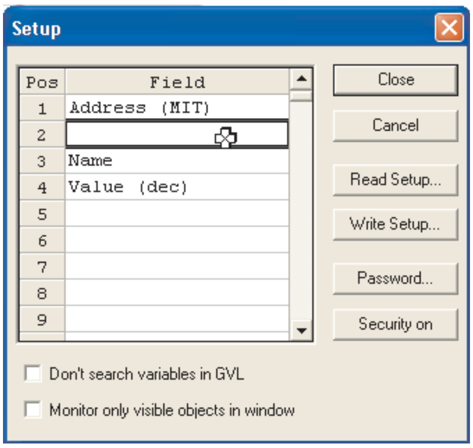
Setup

Pos	Field
1	Address (MIT)
2	Name
3	Value (dec)
4	
5	
6	
7	
8	
9	

□ Don't search variables in GVL

□ Monitor only visible objects in window

A second window appears, showing options for this row, select ***Value (hex)***, ***Value (bin)***. Repeat for ***Address (IEC)*** and ***Type***.



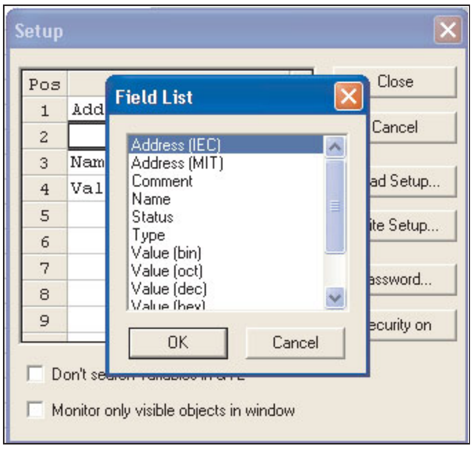
Setup

Pos	Field
1	Address (MIT)
2	
3	Name
4	Value (dec)
5	
6	
7	
8	
9	

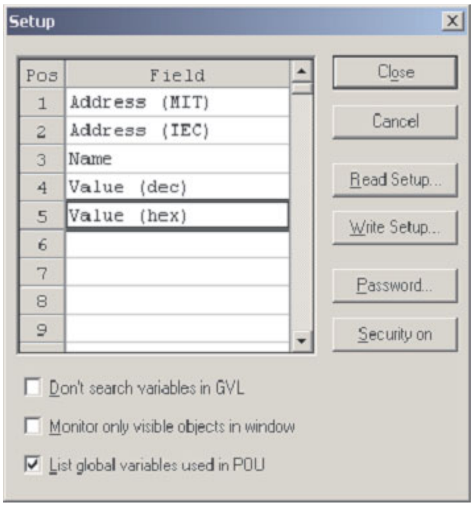
□ Don't search variables in GVL

□ Monitor only visible objects in window

③ Double click on the empty field or press F2 and select ***Address (IEC)*** from the list as shown.



④ Click ***OK*** and the item will be added to the EDM layout. Add ***Value (hex)*** to the Pos 5 field in the table.



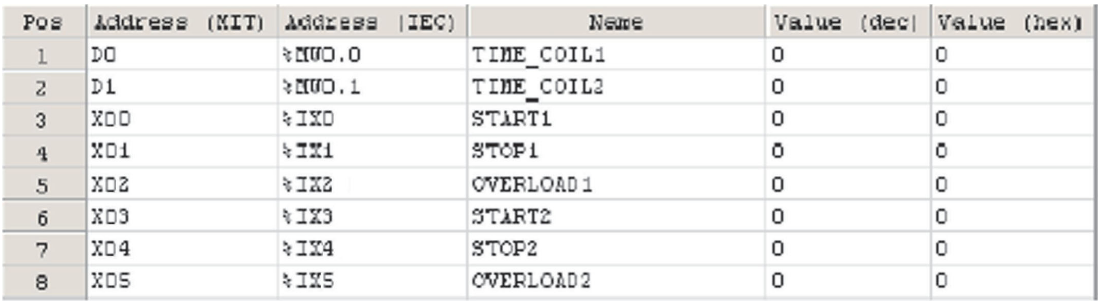
Pos	Field
1	Address (MIT)
2	Address (IEC)
3	Name
4	Value (dec)
5	Value (hex)
6	
7	
8	
9	

☐ Don't search variables in GVL

☐ Monitor only visible objects in window

☑ List global variables used in POU

⑤ Click to close the setup box and observe altered EDM layout:



Pos	Address (MIT)	Address (IEC)	Name	Value (dec)	Value (hex)
1	D0	%MW0.0	TIME_COIL1	0	0
2	D1	%MW0.1	TIME_COIL2	0	0
3	X00	%IX0	START1	0	0
4	X01	%IX1	STOP1	0	0
5	X02	%IX2	OVERLOAD1	0	0
6	X03	%IX3	START2	0	0
7	X04	%IX4	STOP2	0	0
8	X05	%IX5	OVERLOAD2	0	0

In this way, the EDM table can be used to display multiple data on one table.

Try adjusting the column widths and the zoom facility from the ***View*** menu, to display complete picture. The display size is much dependent on the screen resolution set on the computer being used.

From here values can be entered to any object displayed, i.e. the value of D100 may be altered by entering a number into the respective field.

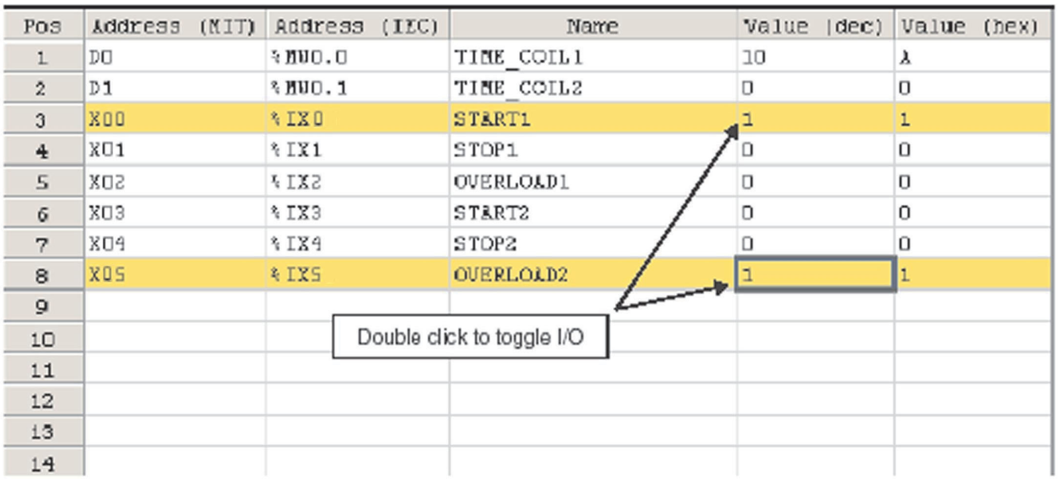
7.1.2 Monitor Limitations

NOTE

Remember, the behaviour of the monitor facility is dependant on the code being run in the PLC; if the PLC code is writing a constant to this address, the value entered will be overwritten by the program. This situation is prevalent here as the values of D0 and D1 are being continuously written to by the PLC code.

7.1.3 Toggling Boolean Variables

Providing the physical input to the PLC is not active, it is possible to toggle the input image in the CPU on and off by double clicking on the value field for that Boolean addresses as shown:

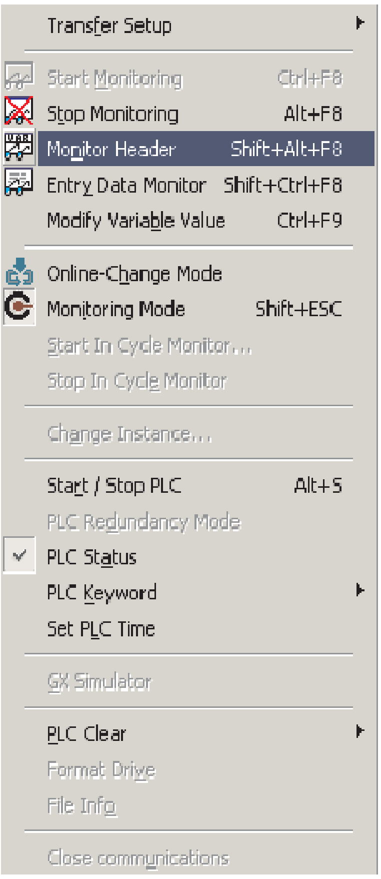


Pos	Address (MIT)	Address (IEC)	Name	Value (dec)	Value (hex)
1	D0	%MW0.0	TIME_COIL1	10	A
2	D1	%MW0.1	TIME_COIL2	0	0
3	X00	%IX0	START1	1	1
4	X01	%IX1	STOP1	0	0
5	X02	%IX2	OVERLOAD1	0	0
6	X03	%IX3	START2	0	0
7	X04	%IX4	STOP2	0	0
8	X05	%IX5	OVERLOAD2	1	1
9					
10					
11					
12					
13					
14					

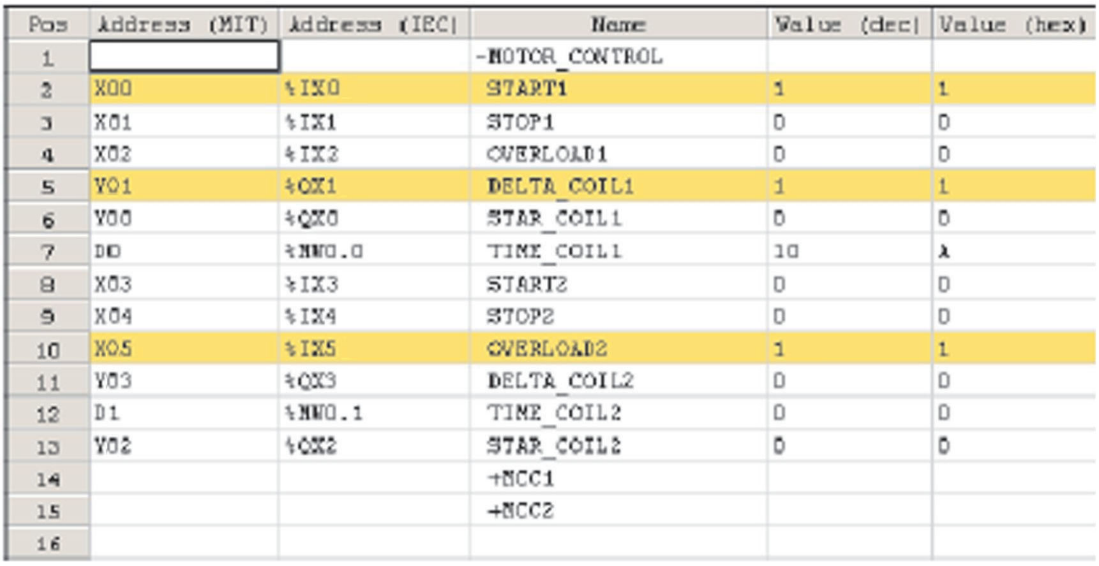
Double click to toggle I/O

7.2 Monitoring Headers

Another facility available, whilst in ***Monitor Mode*** and with the POU body highlighted, is the ***Monitor Header*** function in the ***Online*** menu. It is also available from the Online Toolbar



All elements of the Header identifiers of the highlighted POU are now displayed and monitored:



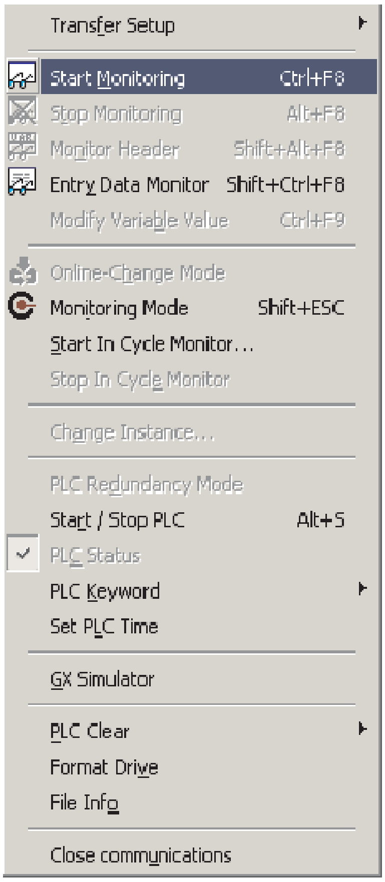
Pos	Address (MIT)	Address (IEC)	Name	Value (dec)	Value (hex)
1			-MOTOR_CONTROL		
2	X00	%IX0	START1	1	1
3	X01	%IX1	STOP1	0	0
4	X02	%IX2	OVERLOAD1	0	0
5	Y01	%QX1	DELTA_COIL1	1	1
6	Y00	%QX0	STAR_COIL1	0	0
7	D0	%MW0.0	TIME_COIL1	10	A
8	X03	%IX3	START2	0	0
9	X04	%IX4	STOP2	0	0
10	X05	%IX5	OVERLOAD2	1	1
11	Y03	%QX3	DELTA_COIL2	0	0
12	D1	%MW0.1	TIME_COIL2	0	0
13	Y02	%QX2	STAR_COIL2	0	0
14			+MCC1		
15			+MCC2		
16					

Note that the Boolean variables in the EDM are shown highlighted, when monitoring.

7.3 Monitor Mode Essentials

Multiple Windows may be monitored simultaneously by first opening them separately and using 'Tile Windows' feature in the Window Menu. It is important to realise when first entering Monitor mode, only the target window in view will be monitored.

Further windows may be monitored by first bringing them into the target view and clicking individually on the ***Start Monitoring*** (Ctrl+F8) selection from the ***Online*** menu:

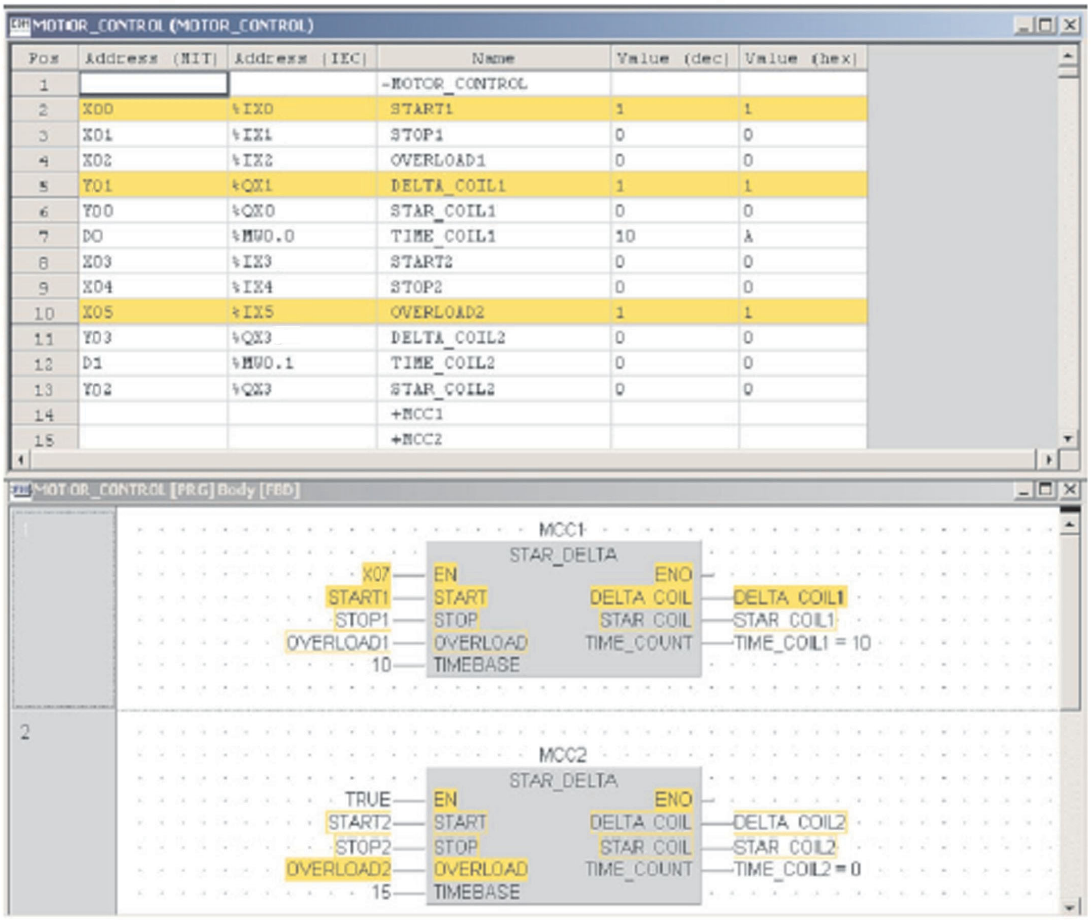


NOTE

This monitor initialisation method is to prevent all open windows from being monitored simultaneously even if they are open but not in view. This would have the effect of potentially significantly increasing the communications traffic between the PLC and the Computer. This would ultimately result in very slow monitor response times on the GX IEC Developer displays, particularly on FX PLC's.

Simultaneous Monitoring of Header and Body

Here is an example of Monitoring a POU and its header simultaneously:



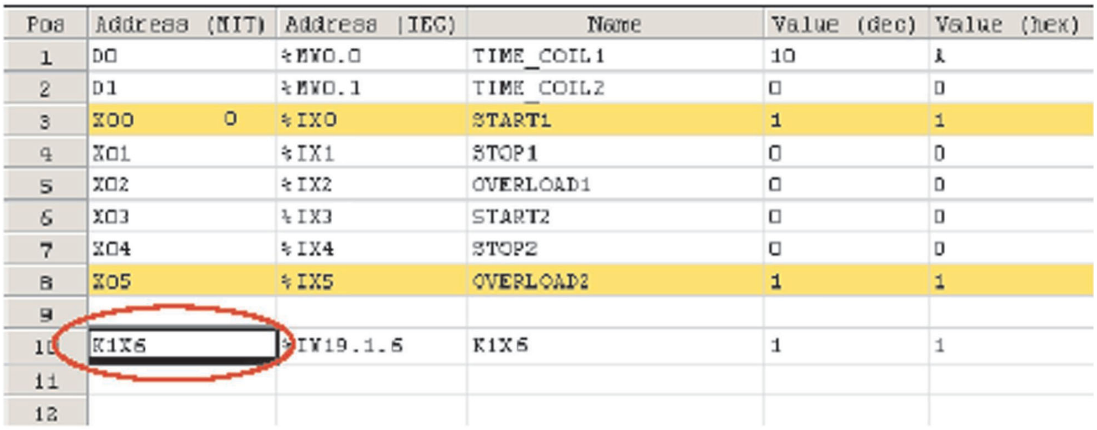
MOTOR_CONTROL (MOTOR_CONTROL)

Pos	Address (MIT)	Address (IEC)	Name	Value (dec)	Value (hex)
1			-MOTOR_CONTROL		
2	X00	%IX0	START1	1	1
3	X01	%IX1	STOP1	0	0
4	X02	%IX2	OVERLOAD1	0	0
5	Y01	%QX1	DELTA_COIL1	1	1
6	Y00	%QX0	STAR_COIL1	0	0
7	D0	%MW0.0	TIME_COIL1	10	A
8	X03	%IX3	START2	0	0
9	X04	%IX4	STOP2	0	0
10	X05	%IX5	OVERLOAD2	1	1
11	Y03	%QX3	DELTA_COIL2	0	0
12	D1	%MW0.1	TIME_COIL2	0	0
13	Y02	%QX3	STAR_COIL2	0	0
14			+MCC1		
15			+MCC2		

MOTOR_CONTROL [PRG] Body [FBD]

7.4 Monitoring Mitsubishi “Transfer Form” Objects

It is also possible to monitor using the Mitsubishi Kn (Official – ‘Transfer Form’) notation for Boolean objects. For example K1X0 monitors X0 - X3 as shown in the following example:

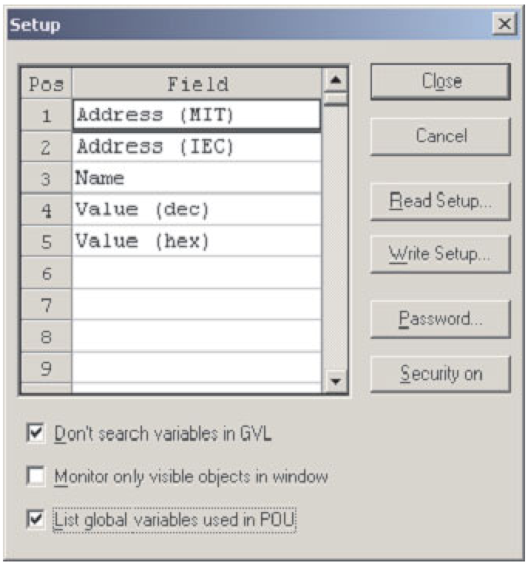


Pos	Address (MIT)	Address (IEC)	Name	Value (dec)	Value (hex)
1	D0	%MW0.0	TIME_COIL1	10	A
2	D1	%MW0.1	TIME_COIL2	0	0
3	X00 0	%IX0	START1	1	1
4	X01	%IX1	STOP1	0	0
5	X02	%IX2	OVERLOAD1	0	0
6	X03	%IX3	START2	0	0
7	X04	%IX4	STOP2	0	0
8	X05	%IX5	OVERLOAD2	1	1
9					
10	K1X6	%IW19.1.6	K1X6	1	1
11					
12					

Setup Options

Don’t Search Variables in GVL - if a direct Mitsubishi address is entered into the ***Entry Data Monitor*** (EDM), for example M0 the system automatically searches the GVL for the identifier. This can take a long time in large projects. By checking the box as shown, this automatic search is disabled.

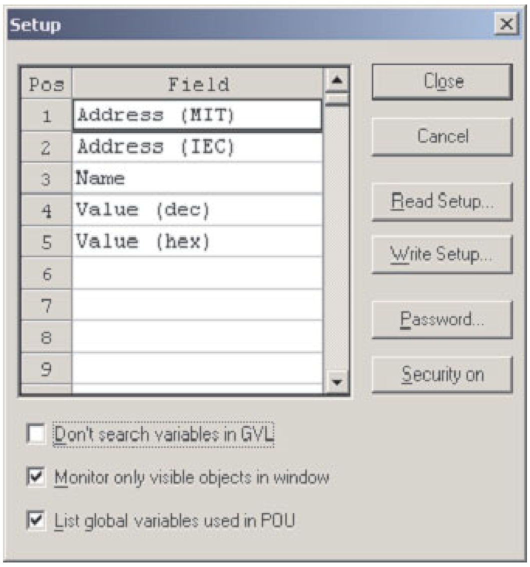
Monitor only Visible Objects in Window - generally all elements in the EDM are monitored, even if they are not visible. By checking the box as shown, only objects in the active window are monitored. This speeds up response for large headers.



Setup

Pos	Field
1	Address (MIT)
2	Address (IEC)
3	Name
4	Value (dec)
5	Value (hex)
6	
7	
8	
9	

☑ Don't search variables in GVL
□ Monitor only visible objects in window
☑ List global variables used in POU



Setup

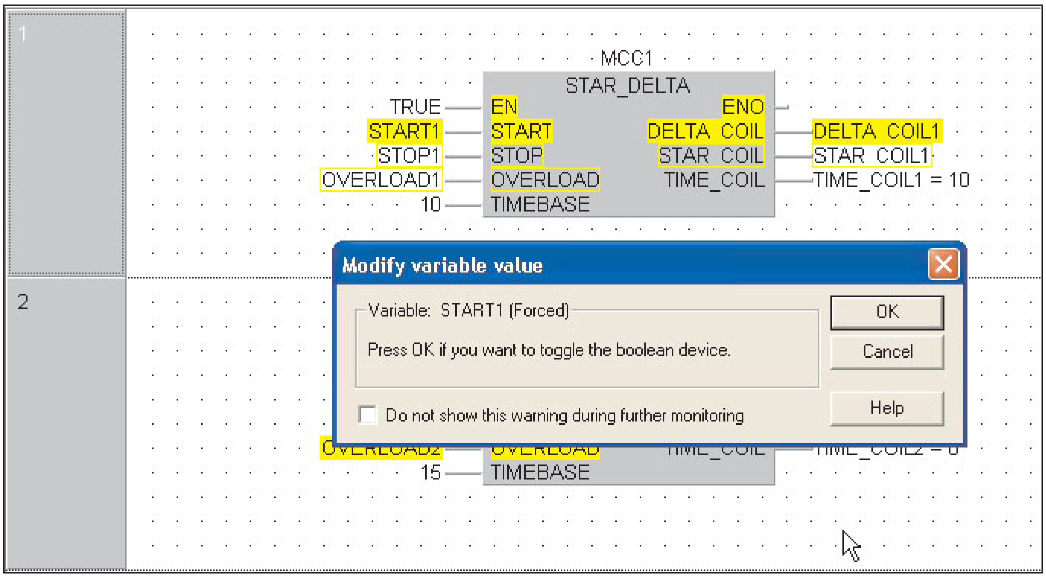
Pos	Field
1	Address (MIT)
2	Address (IEC)
3	Name
4	Value (dec)
5	Value (hex)
6	
7	
8	
9	

□ Don't search variables in GVL
☑ Monitor only visible objects in window
☑ List global variables used in POU

7.5 Modifying Variable Values from the POU Body

It is possible to change the value of a variable from the POU body, in Monitor Mode. This can be a toggle of a Boolean or writing a value to an Integer/Real value etc. To invoke this, double click on the variable label, i.e. ENABLE. This dialogue will appear, click OK to toggle on, click OK again to toggle off. If there is PLC code writing to this variable, then this will overwrite this action.

The dialogue box can be disabled, so that operation is simply by the mouse.



Variable: START1 (Forced)

Press OK if you want to toggle the boolean device.

□ Do not show this warning during further monitoring

For Integer/Real variables, use the same procedure, i.e. double click on the variable name, whilst in monitor mode. The new value can be entered either as decimal or as a hexadecimal value.

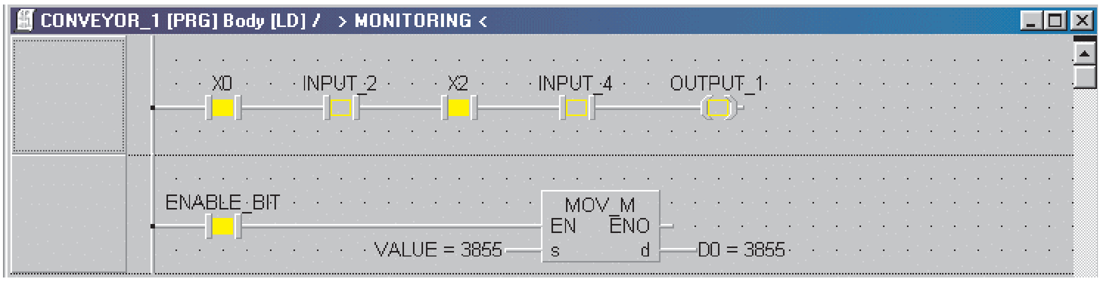
Again, if there is PLC code writing to this variable, then this will overwrite this action.

NOTE Both operations also operate on direct MELSEC addresses (For further illustrations, see previous section: “Functions”).

IMPORTANT TIP

When using the Ladder editor, hold down the CTRL key and double click on the variable name. The actual address of the selected GV will then be displayed, as shown below. Repeating the operation will toggle back to the identifier.

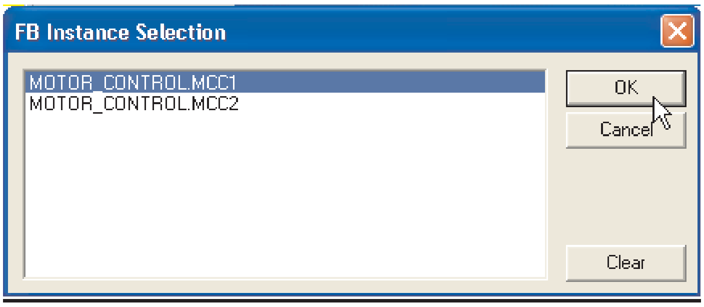
If Monitor Mode is stopped, then started again, identifiers are displayed.



7.6 Monitoring “Instances” of Function Blocks

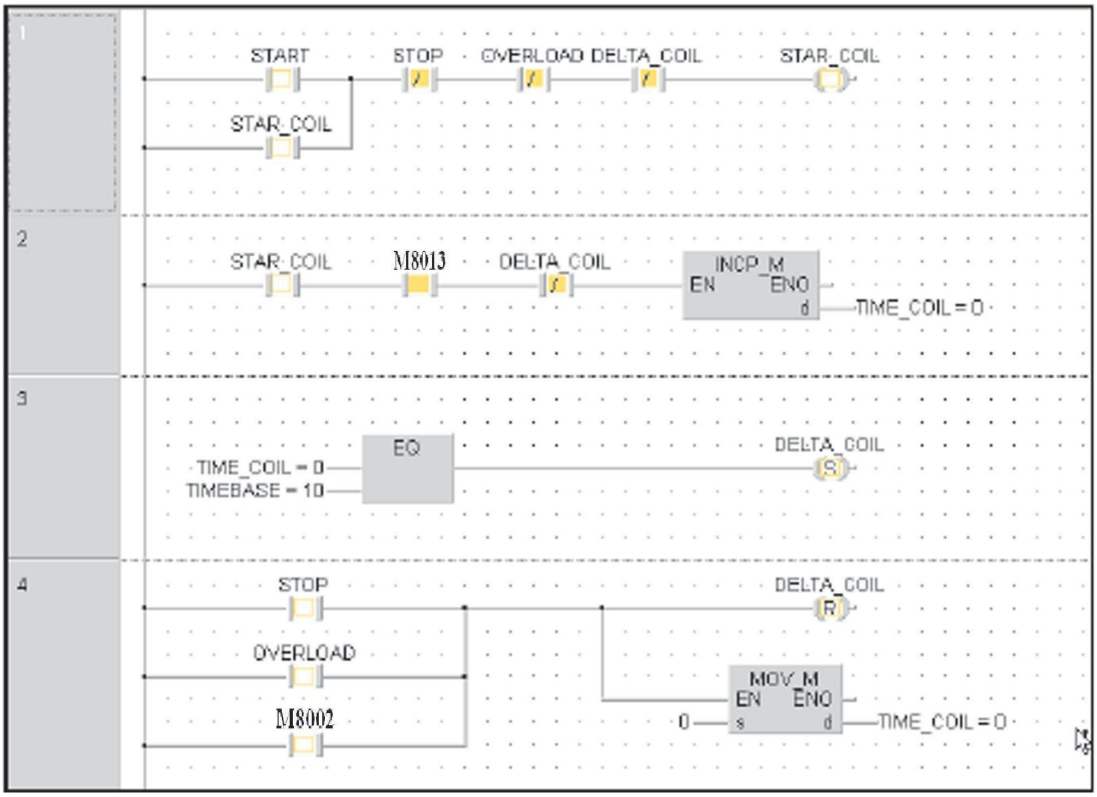
Individual “Instances” of Function Blocks may be monitored independently.

① To monitor an instance of the POU FB STAR_DELTA in the current project, open the POU Body and click on the Monitor mode button. The following dialogue choice window will be displayed:



MOTOR_CONTROL.MCC1
MOTOR_CONTROL.MCC2

② Select the instance of the Function Block MOTOR_CONTROL.MCC1 and observe the monitored page:

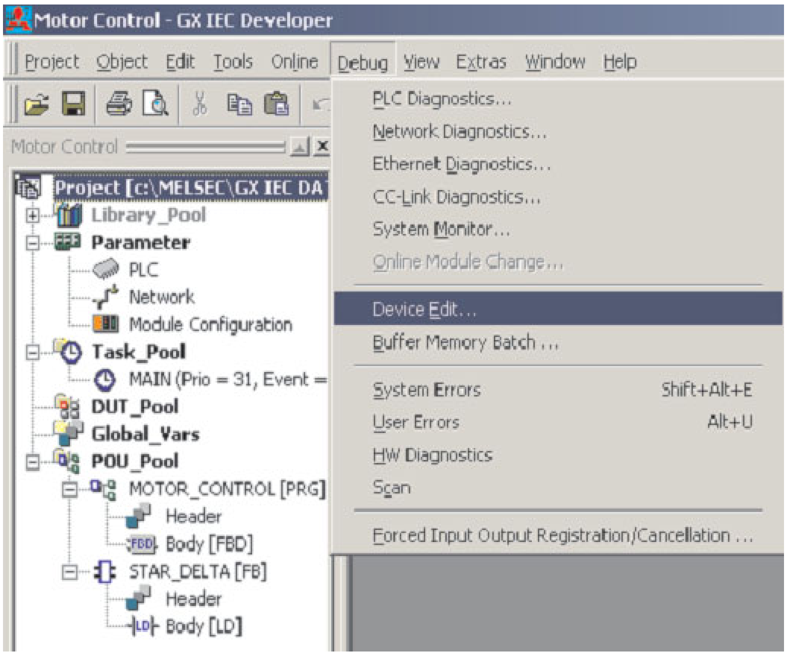


In this manner every instance of any Function Block may be monitored autonomously.

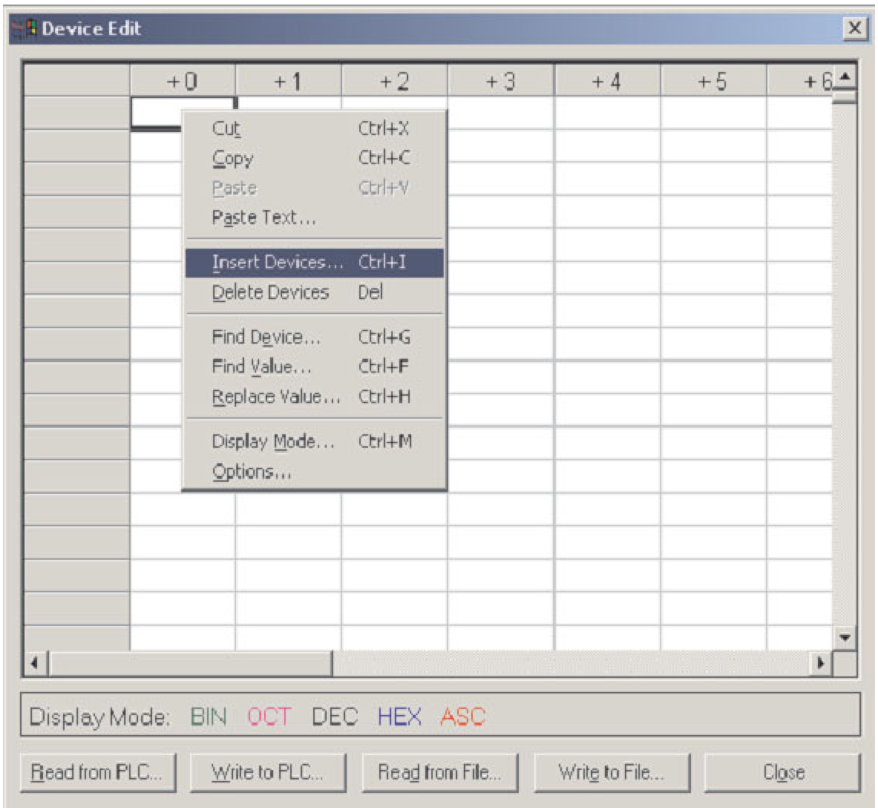
8 Device Edit

The ***Device Edit*** function is used to edit batch devices.

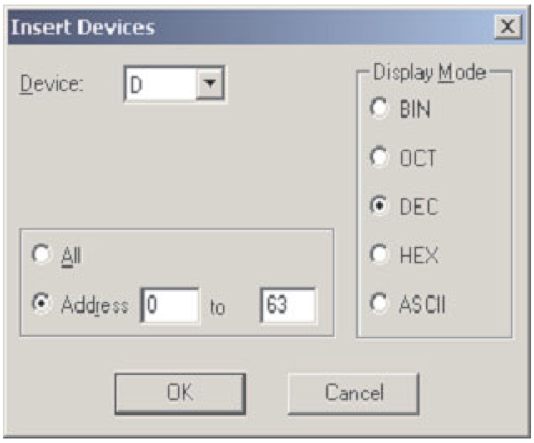
① Select ***Device Edit*** from the ***Debug*** menu.



② Highlight the cell in the top left hand corner. Click the right mouse button and then select ***Insert Devices***:



③ Select a device type, from the ***Device*** selection box. If you want all devices of this type, then just click ***OK***. It's more likely though; you will want to enter a range by clicking on the address field and entering your range, then click ***OK***.



Insert Devices

Device: D

○ All

● Address 0 to 63

Display Mode

○ BIN

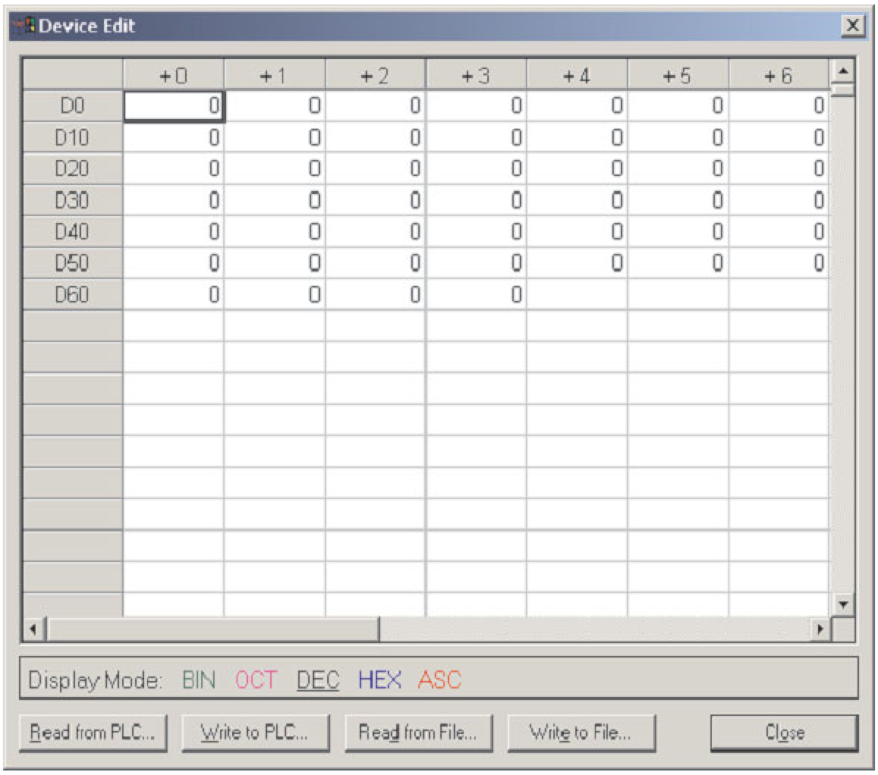
○ OCT

● DEC

○ HEX

○ ASCII

The device table can be configured as you wish and can be stored, as a file or written to the PLC. Information can also be uploaded from the PLC and displayed as below.



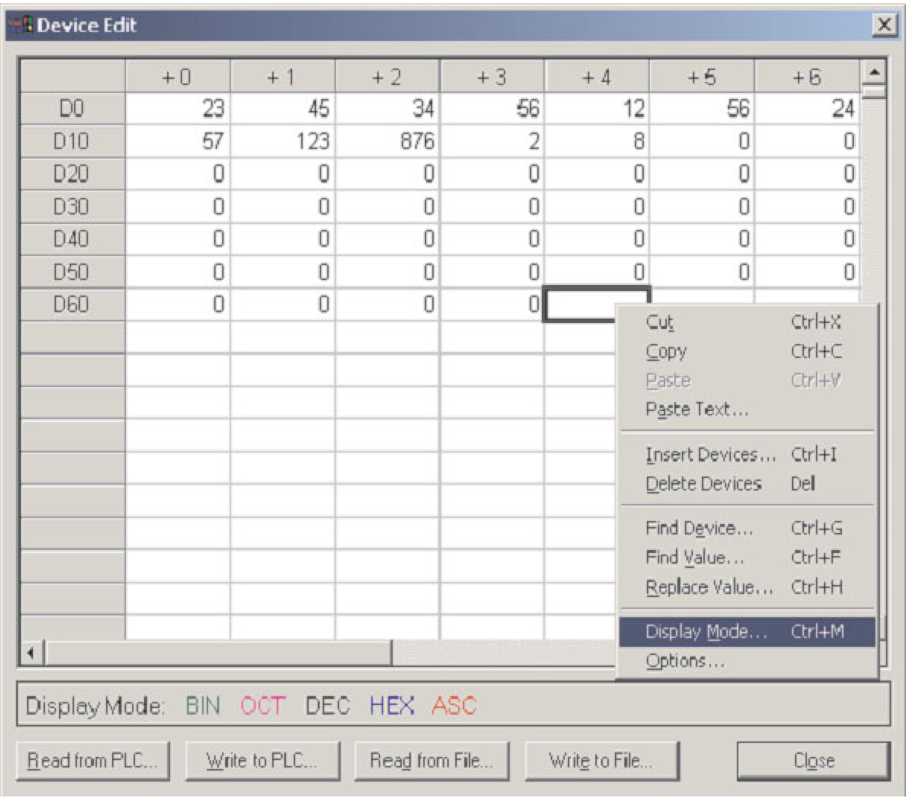
Device Edit

	+0	+1	+2	+3	+4	+5	+6
D0	0	0	0	0	0	0	0
D10	0	0	0	0	0	0	0
D20	0	0	0	0	0	0	0
D30	0	0	0	0	0	0	0
D40	0	0	0	0	0	0	0
D50	0	0	0	0	0	0	0
D60	0	0	0	0			

Display Mode: BIN OCT DEC HEX ASC

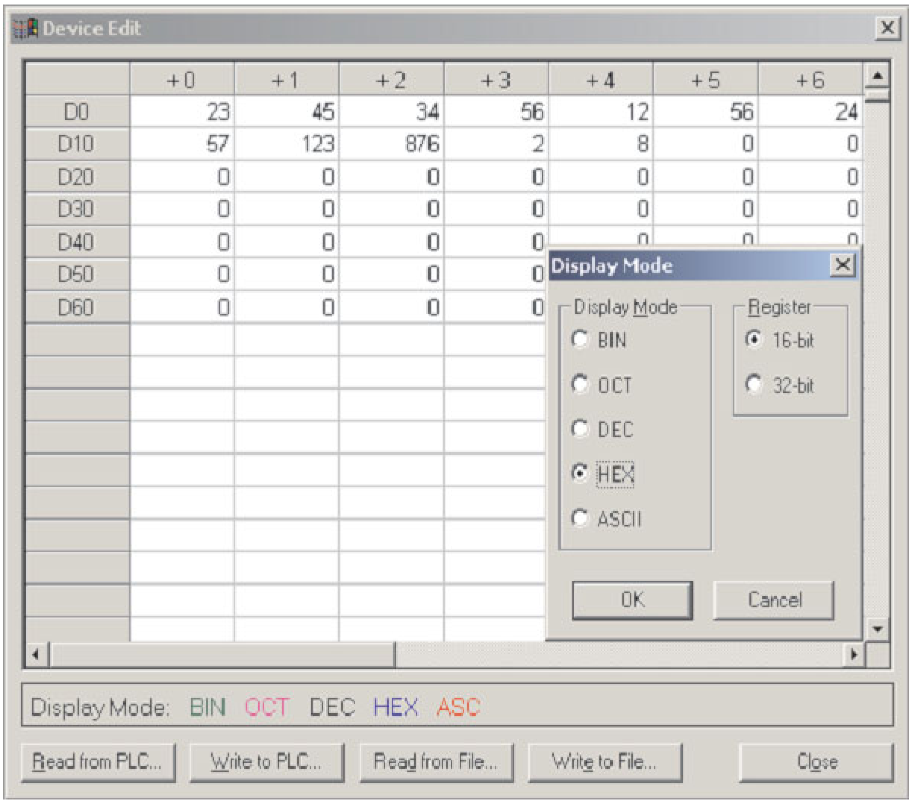
The right mouse button supports many editing functions, find and replace, copy / paste, etc.

④ Highlight a row by clicking on the left hand box, i.e. “D0” Select ***Display Mode***:



	+0	+1	+2	+3	+4	+5	+6
D0	23	45	34	56	12	56	24
D10	57	123	876	2	8	0	0
D20	0	0	0	0	0	0	0
D30	0	0	0	0	0	0	0
D40	0	0	0	0	0	0	0
D50	0	0	0	0	0	0	0
D60	0	0	0	0			

This window allows the display format to be changed - try ***HEX***.



	+0	+1	+2	+3	+4	+5	+6
D0	23	45	34	56	12	56	24
D10	57	123	876	2	8	0	0
D20	0	0	0	0	0	0	0
D30	0	0	0	0	0	0	0
D40	0	0	0	0	0	0	0
D50	0	0	0	0			
D60	0	0	0	0			

Display Mode

Display Mode: ○ BIN ○ OCT ○ DEC ● HEX ○ ASCII

Register: ● 16-bit ○ 32-bit

It should be noticed that the selected row now displays values in hexadecimal, the other values remain unchanged. In fact, individual cells can have different display formats, making this feature extremely flexible.

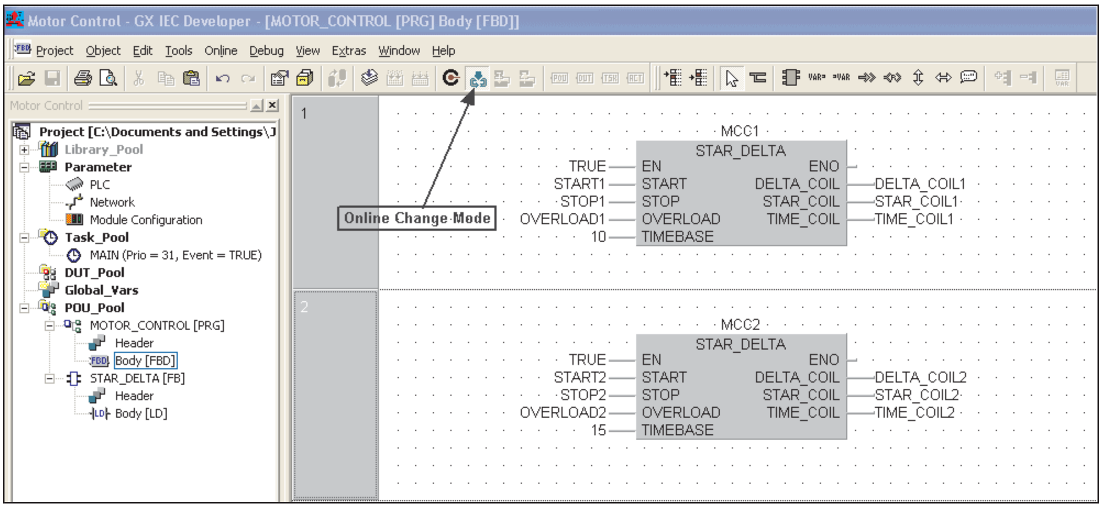
9 Online Mode

There are two methods for evoking online editing; via the online menu or the toolbar icon. Use ***Save as*** in the ***Project*** menu to create a copy of the current project. Rename the Copy to "Motor_Control_Mod". The following operations will apply to this modified program.

Rebuild the project and download it to the PLC.

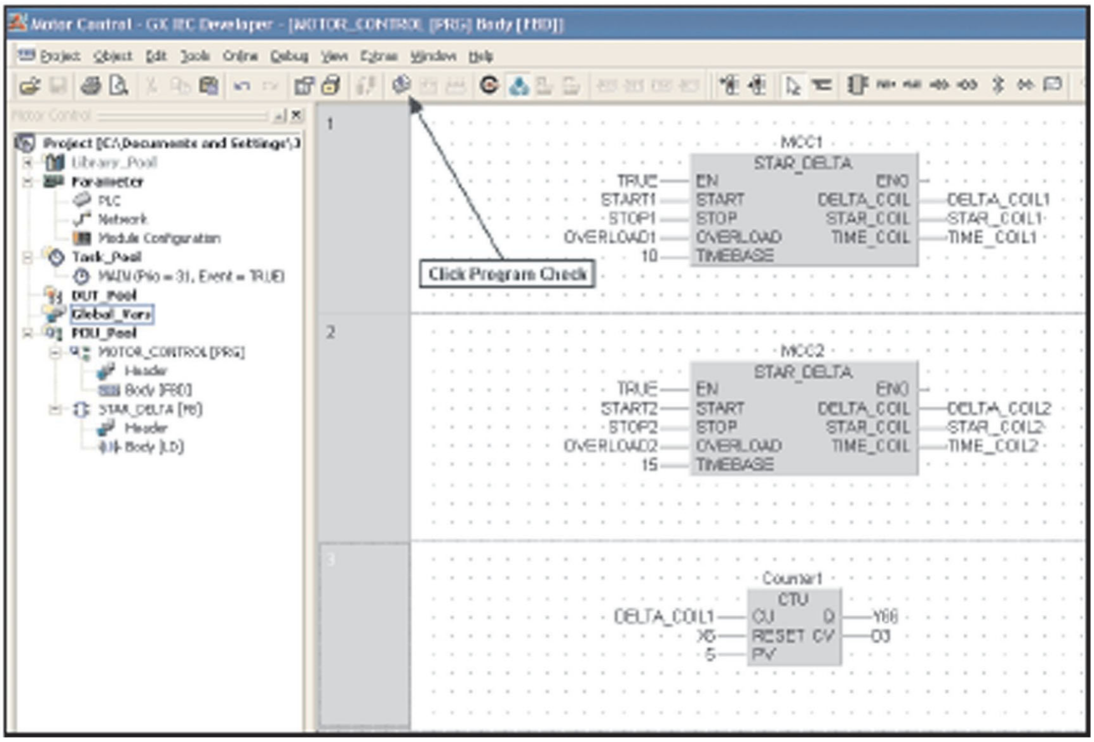
9.1 Online Change Mode

① Open the body of the 'MOTOR_CONTROL' POU and select ***Online change mode***:



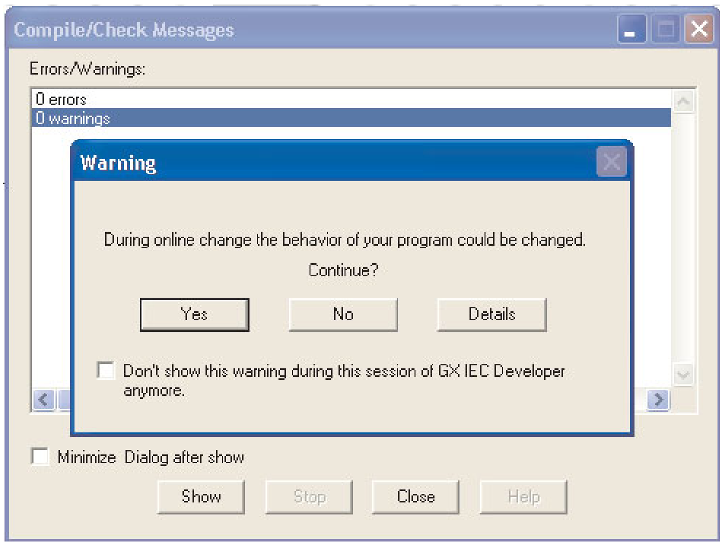
Online Change Mode

② Add an additional network as shown below:



Click Program Check

③ Then with the mouse, click away from this network or click on the check button and the changes are compiled and sent to the PLC automatically following a prompt to carry out or abort the action:



Compile/Check Messages

Errors/Warnings:

0 errors
0 warnings

Warning

During online change the behavior of your program could be changed.

Continue?

Yes No Details

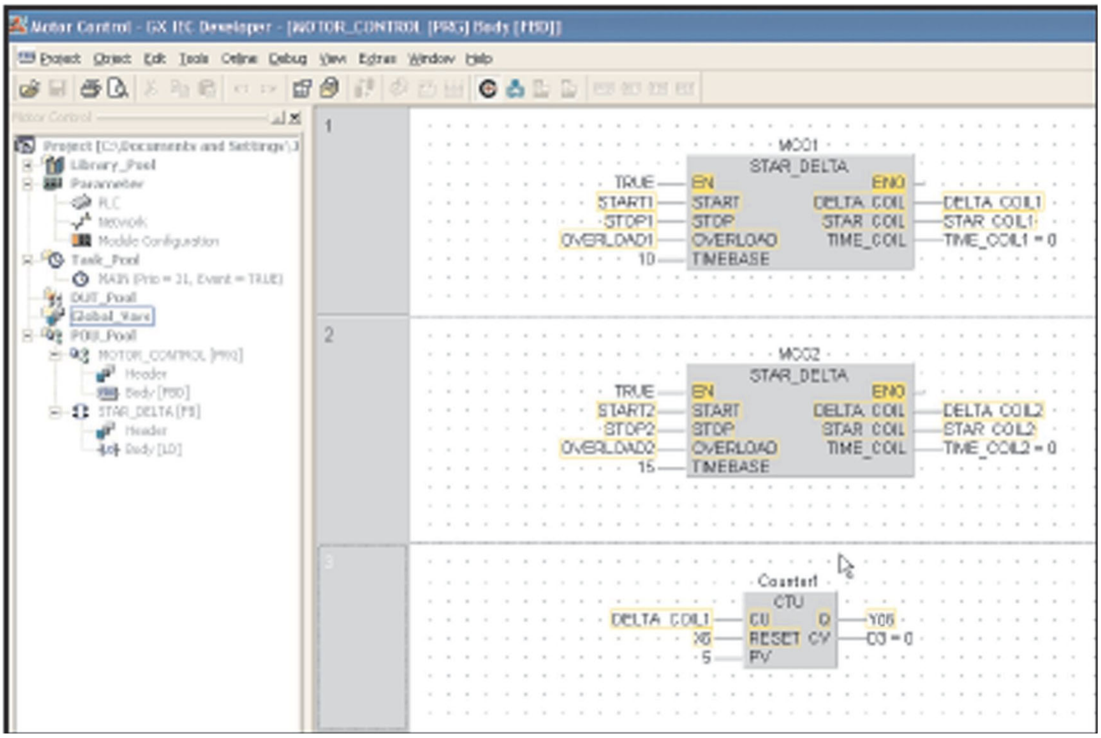
□ Don't show this warning during this session of GX IEC Developer anymore.

□ Minimize Dialog after show

Show Stop Close Help

NOTE Online editing is only allowed if the code is identical in the resident project and PLC.

④ Enter Monitor mode and observe the operation of the modified block:

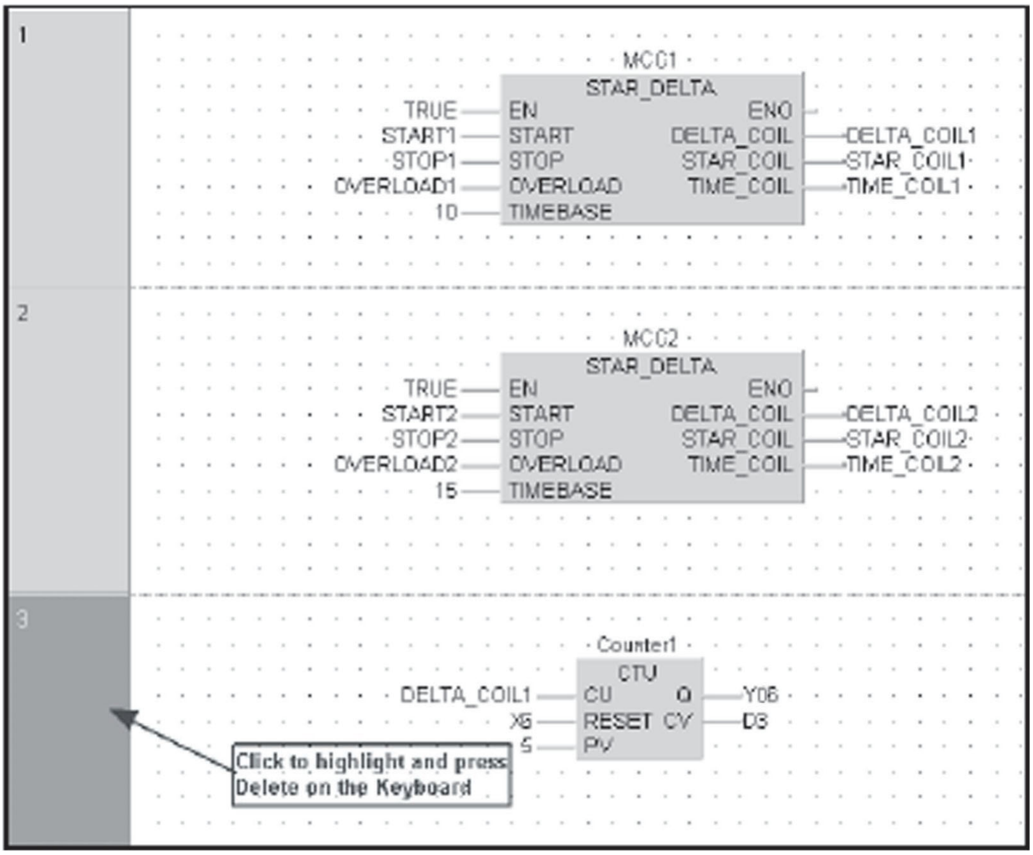


Motor Control - GX IEC Developer - [MOTOR_CONTROL [PRG] Body [FBD]]

9.2 Online Program Change

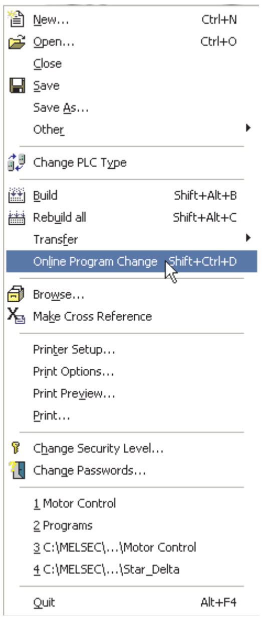
Where complete networks are to be added or removed, the “Online Program Change” operation must be used. This method is the preferred method of making changes to the program whilst on-line. For example: If the recently added counter network is to be removed from the program, carry out the following procedure (Remember the PLC and GX IEC Developer programs must be identical before proceeding).

① Highlight network 3 on the POU body “MOTOR_CONTROL” and press "Delete" on the keyboard.



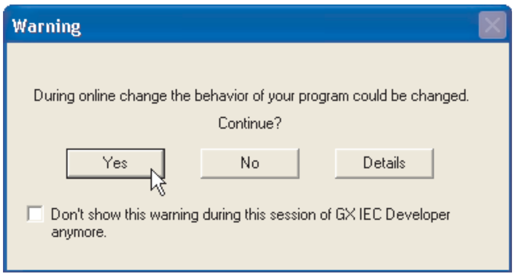
Click to highlight and press Delete on the Keyboard

② Invoke the ***Online Program Change*** feature from the ***Project*** Menu. GX IEC Developer will compile and write the online change automatically.



The system will prompt to continue or abort the process at this point.

③ Click ***Yes*** and wait for the download synchronisation process to complete:



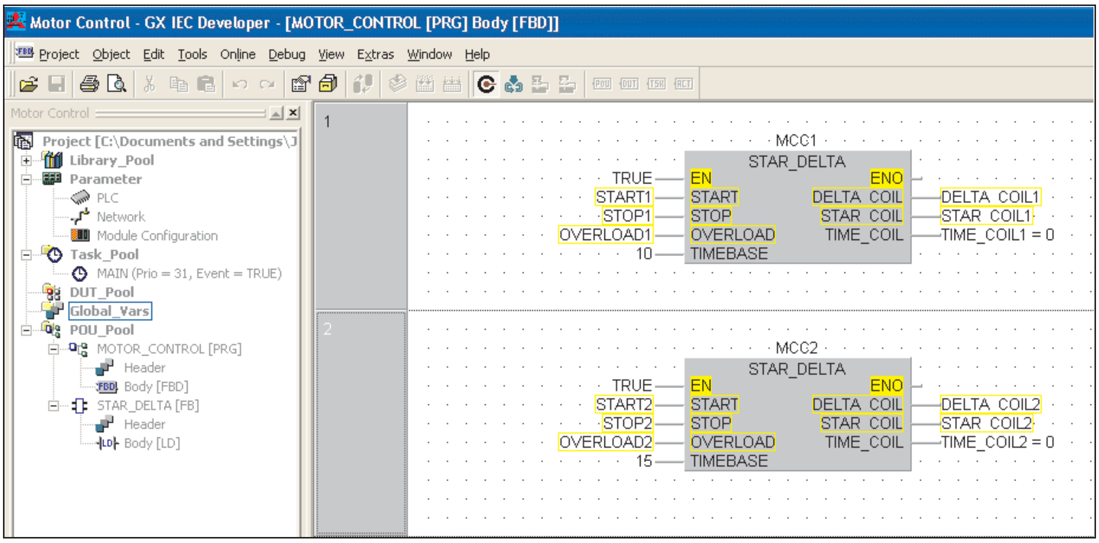
Warning

During online change the behavior of your program could be changed.

Continue?

☐ Don't show this warning during this session of GX IEC Developer anymore.

④ Confirm correct operation by entering ***Monitor mode*** in the active POU.



10 Data Unit Types (DUT)

The following example illustrates the operation of DUT (**D**ata **U**nit **T**ypes).

The previous “Motor Control” example will be used to illustrate the procedures for creating and using DUT’s.

User defined Data Unit Types (DUT), can be created. This can be useful for programs which contain common parts, for example; the control of a number of identical ‘Star Delta’ motor starters. Therefore a Data Unit Type, called ‘SD’ can be created, composing patterns of different elements, i.e. INT, BOOL etc.

When completing a global variable list, identifiers of type SD can be used. This means that the predefined group called ‘SD’ can be used with the elements defined as required for each Motor Control, thus reducing design time and allowing re-use of the DUT together with Function Blocks.

If an element called START exists in type “SD,” then it can be reused for each ‘Star Delta’ Motor Control instance when declared in the GVL; STAR_DELTA1.START, STAR_DELTA2.START etc.

This means for one declaration, many derivatives can be used. One particular use for this procedure is in the interface to Tag Groups in SCADA systems. This can keep communication cycles fast and efficient by utilising shorter and sequential data transactions, instead of multiple fragmented data requests to and from the PLC.

10.1 Example use of a DUT

The following example illustrates the use of a DUT.

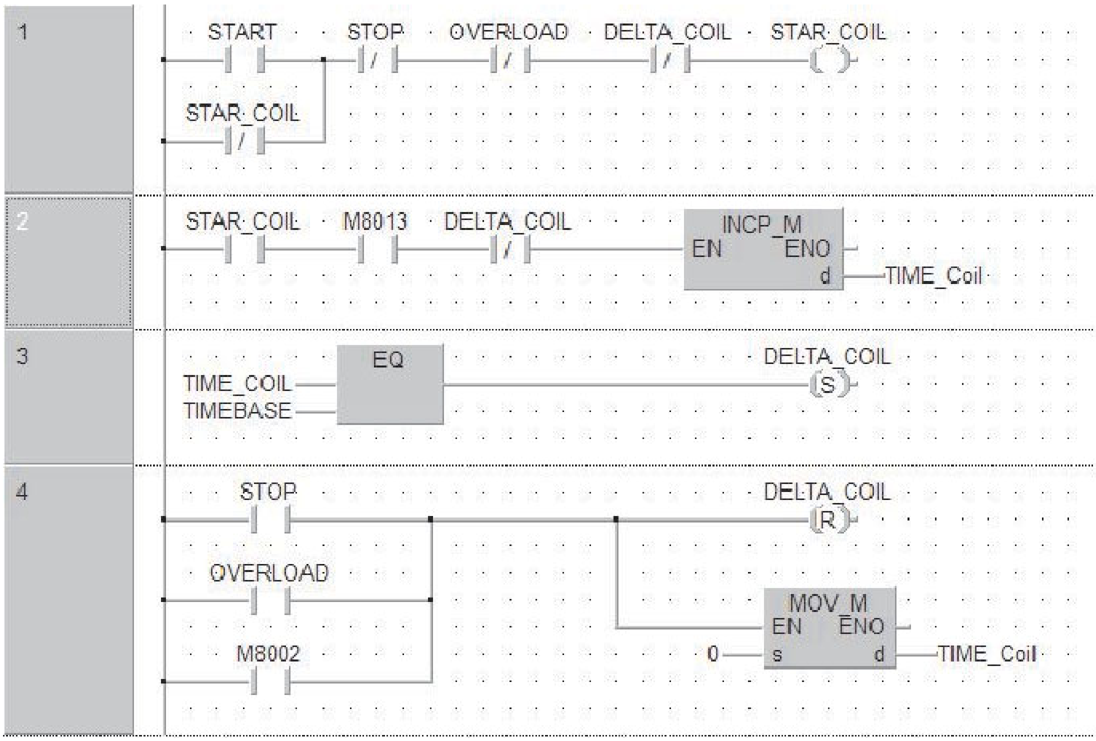
① Create a new project called “Motor Control DUT”:

② Ceate a new Program POU called MOTOR_CONTROL

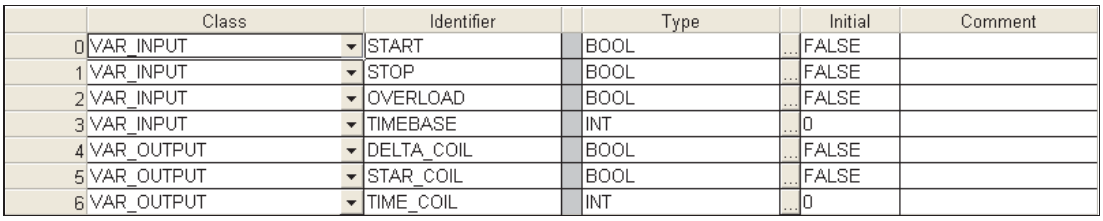
③ Create a new Task in the task pool called MAIN and bind the Program MOTOR_CONTROL to it.

④ Create a new Function Block “STAR_DELTA” and re-enter the following program code. Alternatively, ‘Copy-Paste’ the original function block, ‘Body and Header’, from the project “Motor Control” as follows:

Body: STAR_DELTA

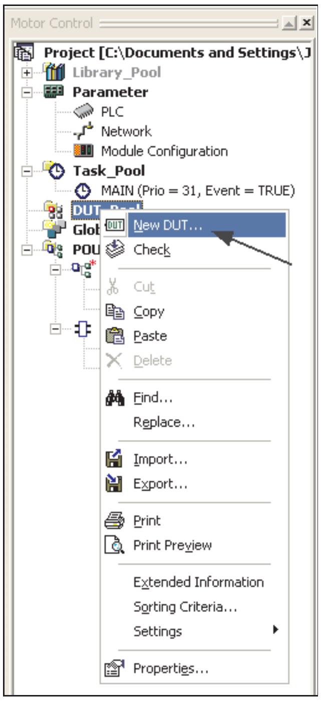


Header: STAR_DELTA



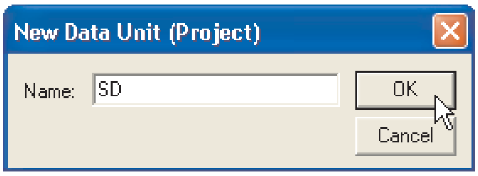
	Class	Identifier	Type	Initial	Comment
0	VAR_INPUT	START	BOOL	FALSE	
1	VAR_INPUT	STOP	BOOL	FALSE	
2	VAR_INPUT	OVERLOAD	BOOL	FALSE	
3	VAR_INPUT	TIMEBASE	INT	0	
4	VAR_OUTPUT	DELTA_COIL	BOOL	FALSE	
5	VAR_OUTPUT	STAR_COIL	BOOL	FALSE	
6	VAR_OUTPUT	TIME_COIL	INT	0	

The Header contains the definitions (Mask) of the data types that will be used when creating the DUT “SD”.



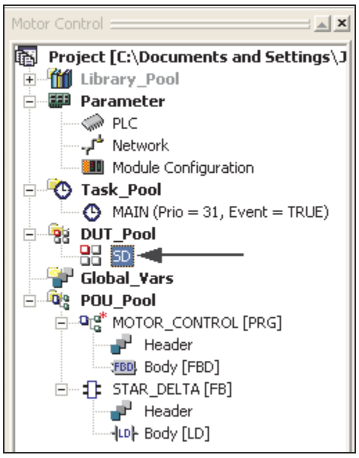
⑤ Create a new DUT by right clicking on the ***DUT Pool*** icon in the Program navigation window

or from the DUT icon on the toolbar.



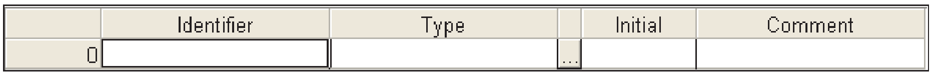
Name: SD

⑥ Enter the new DUT name as SD at the prompt.



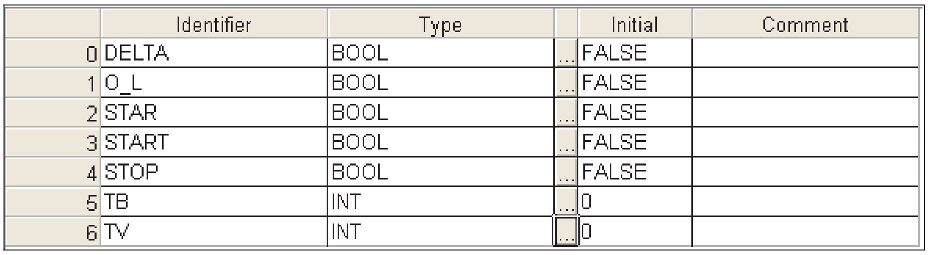
The new DUT will now be displayed under the ***DUT Pool*** in the project.

⑦ Open the DUT by clicking on the Icon and the following will be displayed:



	Identifier	Type		Initial	Comment
0			...		

⑧ Enter the following data into the DUT “SD”.

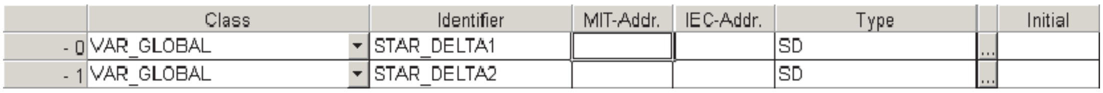


	Identifier	Type		Initial	Comment
0	DELTA	BOOL	...	FALSE	
1	O_L	BOOL	...	FALSE	
2	STAR	BOOL	...	FALSE	
3	START	BOOL	...	FALSE	
4	STOP	BOOL	...	FALSE	
5	TB	INT	...	0	
6	TV	INT	...	0	

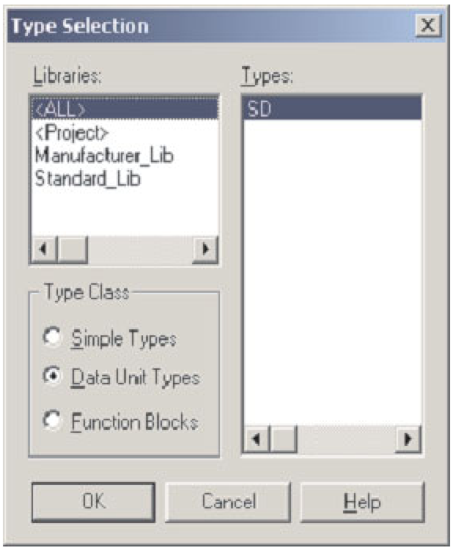
⑨ Close the DUT and save the program.

⑩ Open the GVL and create 2 new entries STAR_DELTA1 and STAR_DELTA2.

⑪ Click the ‘ellipsis’ ... to specify the ***Type*** as “Data Unit Types” SD for both entries:



	Class	Identifier	MIT-Addr.	IEC-Addr.	Type		Initial
- 0	VAR_GLOBAL	STAR_DELTA1			SD	...	
- 1	VAR_GLOBAL	STAR_DELTA2			SD	...	



Type Selection

Libraries:

<ALL>
<Project>
Manufacturer_Lib
Standard_Lib

Types: SD

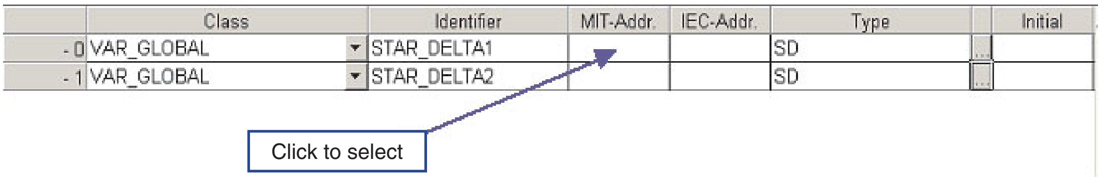
Type Class

○ Simple Types

● Data Unit Types

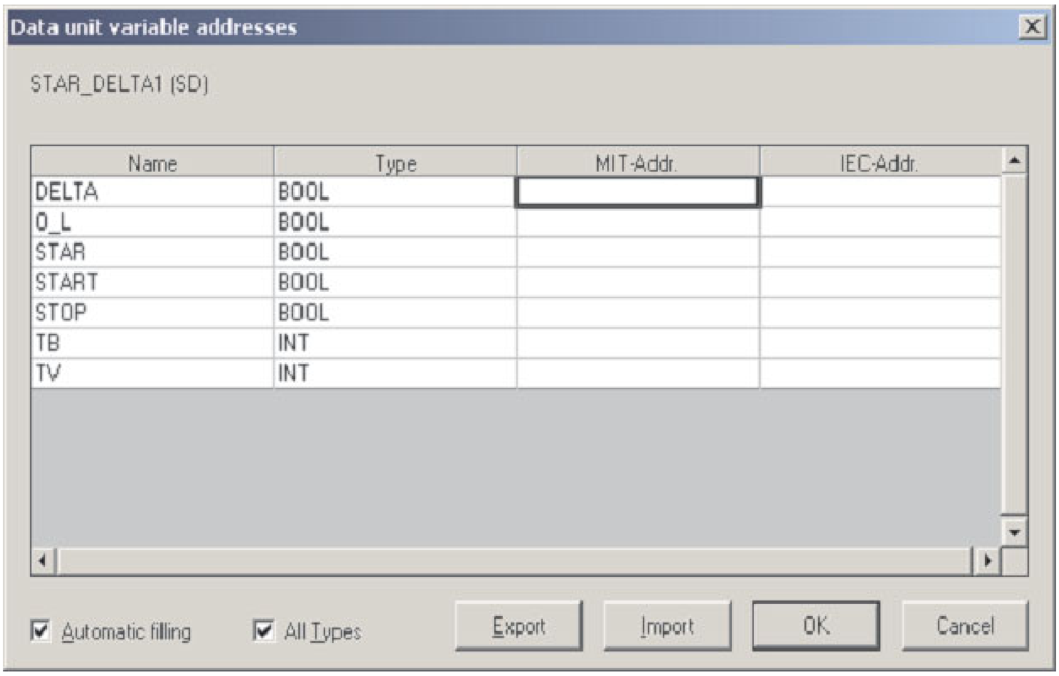
○ Function Blocks

⑫ Next, click on the ***MIT-Addr.*** cell for STAR_DELTA1 to enter the variable data for the selected DUT entry:



	Class	Identifier	MIT-Addr.	IEC-Addr.	Type		Initial
- 0	VAR_GLOBAL	STAR_DELTA1			SD	...	
- 1	VAR_GLOBAL	STAR_DELTA2			SD	...	

Resulting window:



Data unit variable addresses

STAR_DELTA1 (SD)

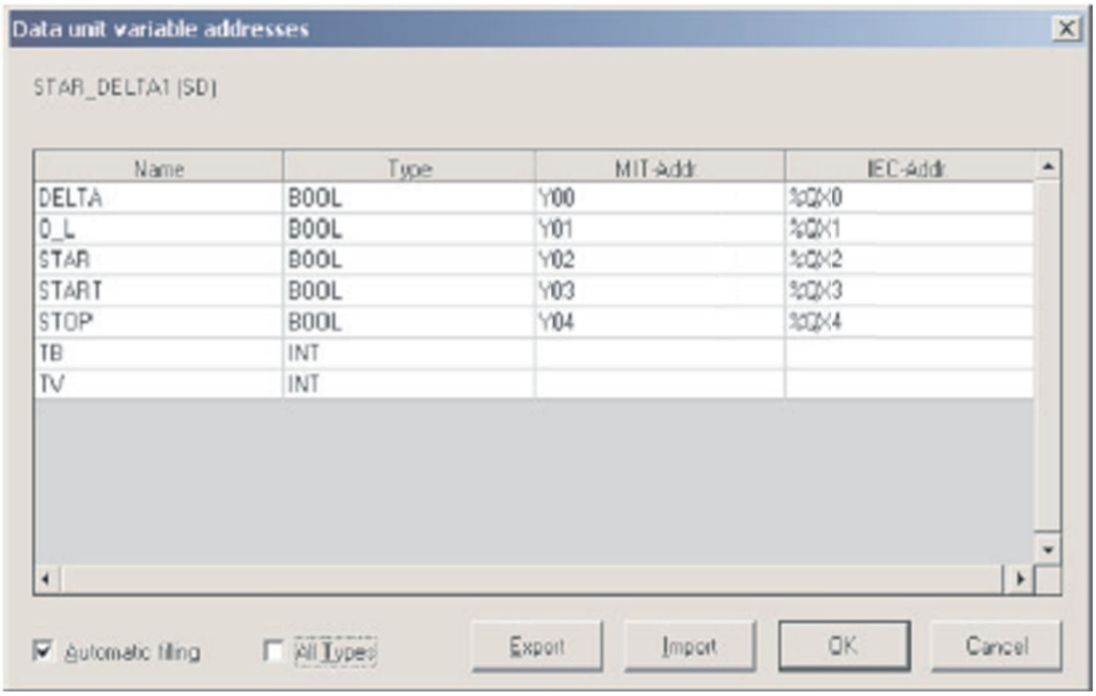
Name	Type	MIT-Addr.	IEC-Addr.
DELTA	BOOL		
O_L	BOOL		
STAR	BOOL		
START	BOOL		
STOP	BOOL		
TB	INT		
TV	INT		

☑ Automatic filling ☑ All Types

10.2 Automatic Filling, Variables

① Deselect ***All types*** as this operation is illegal when using mixed variable types.

② Enter Y00 in the ***MIT-Addr.*** position for the variable: ‘DELTA’:



Data unit variable addresses

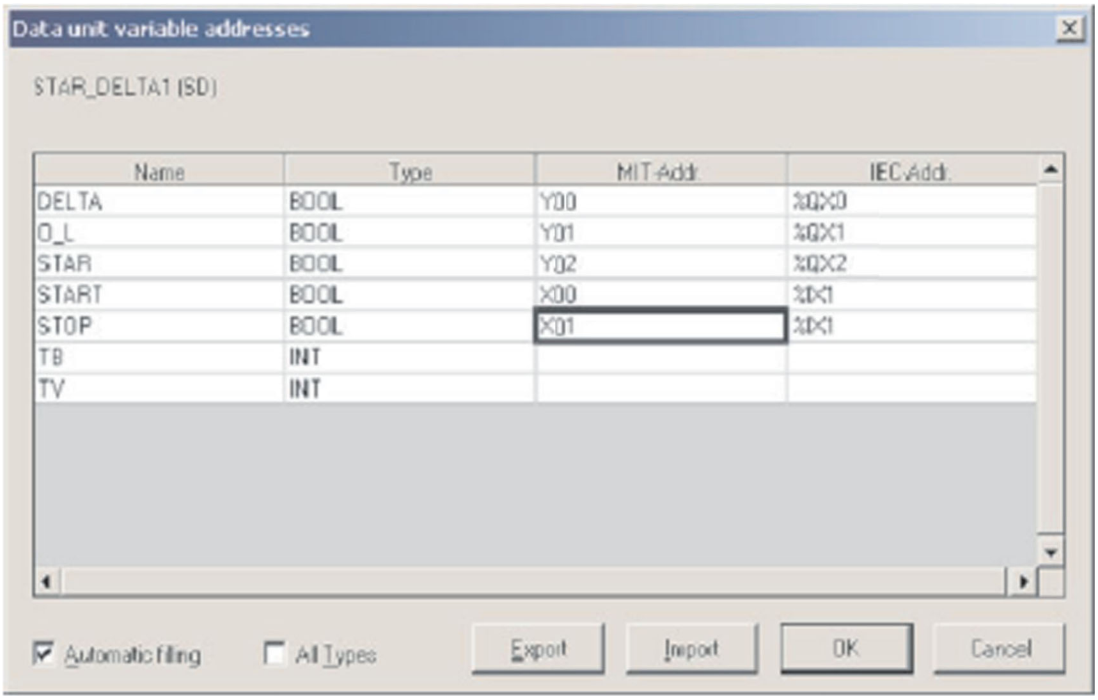
STAR_DELTA1 (SD)

Name	Type	MIT-Addr.	IEC-Addr.
DELTA	BOOL	Y00	%QX0
O_L	BOOL	Y01	%QX1
STAR	BOOL	Y02	%QX2
START	BOOL	Y03	%QX3
STOP	BOOL	Y04	%QX4
TB	INT		
TV	INT		

☑ Automatic filling ☐ All Types

The system will try to sequentially ‘Auto Fill’ the variables of type BOOL. Although in many situations this is recommended, in this case it is only partially successful.

③ Therefore overtype “START and STOP” variables with X00 and X01 thus:



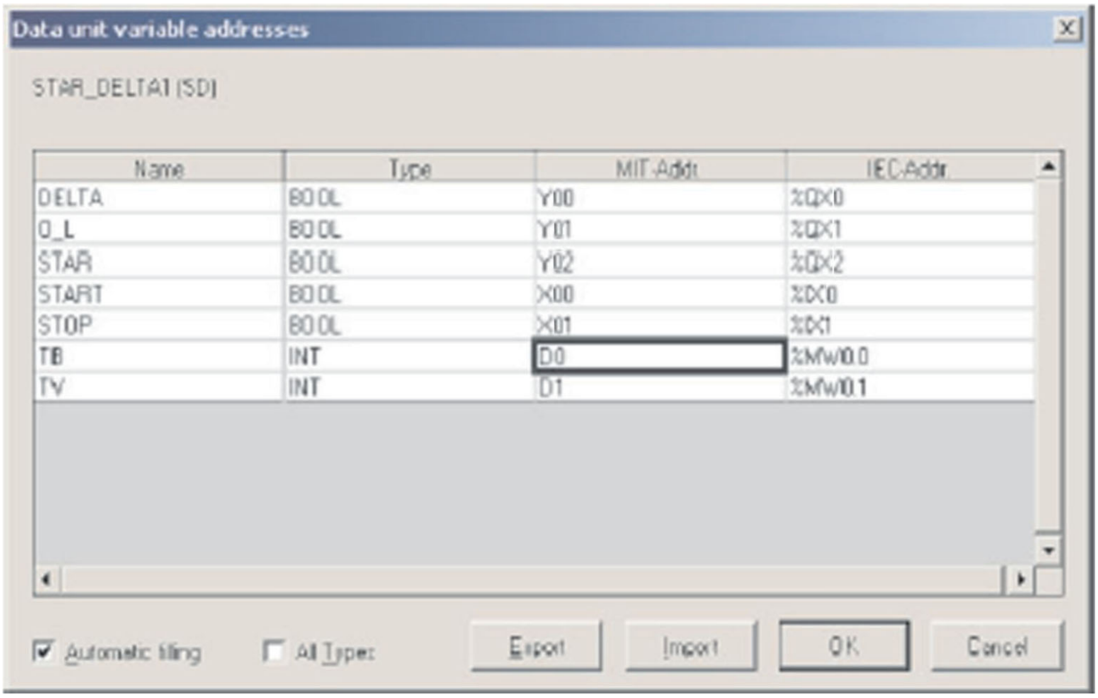
Data unit variable addresses

STAR_DELTA1 (SD)

Name	Type	MIT-Addr.	IEC-Addr.
DELTA	BOOL	Y00	%QX0
O_L	BOOL	Y01	%QX1
STAR	BOOL	Y02	%QX2
START	BOOL	X00	%IX1
STOP	BOOL	X01	%IX1
TB	INT		
TV	INT		

☑ Automatic filling ☐ All Types

④ Finally, enter the two remaining Integer Variables TB and TV using MELSEC addresses D0 and D1 using the “Auto Fill” feature:



Data unit variable addresses

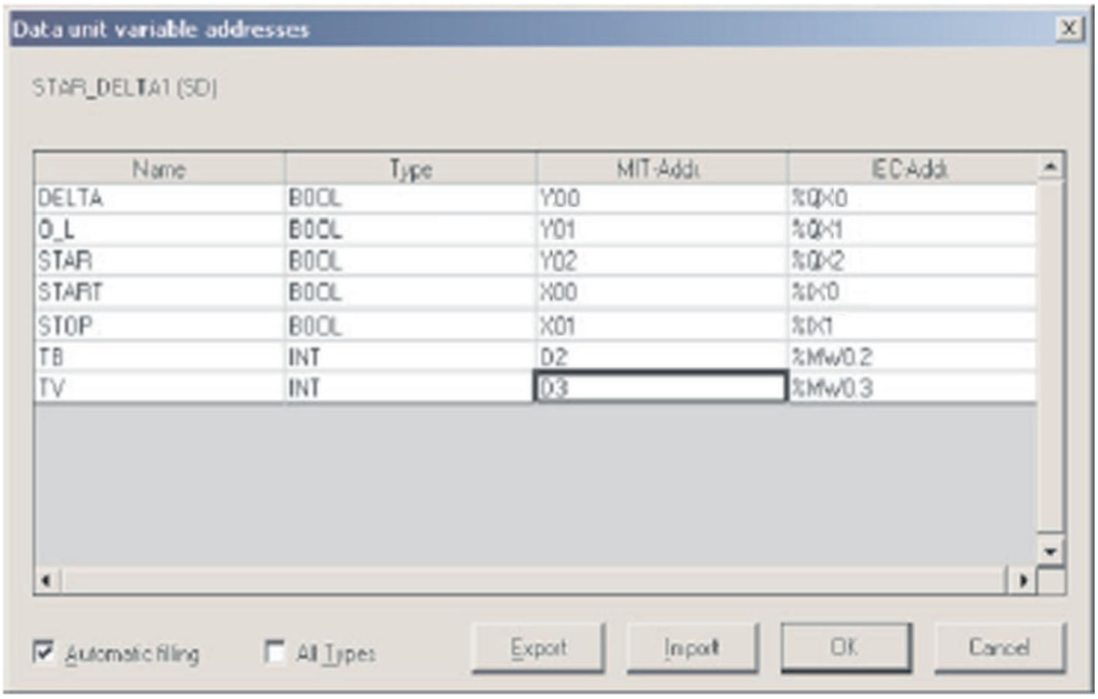
STAR_DELTA1 (SD)

Name	Type	MIT-Addr.	IEC-Addr.
DELTA	BOOL	Y00	%QX0
O_L	BOOL	Y01	%QX1
STAR	BOOL	Y02	%QX2
START	BOOL	X00	%IX0
STOP	BOOL	X01	%IX1
TB	INT	D0	%MW0.0
TV	INT	D1	%MW0.1

☑ Automatic filling □ All Types

⑤ Click OK to save the current configuration.

⑥ Repeat this series of operations for “STAR_DELTA2” entering the next sequential head address for each variable “TYPE”:



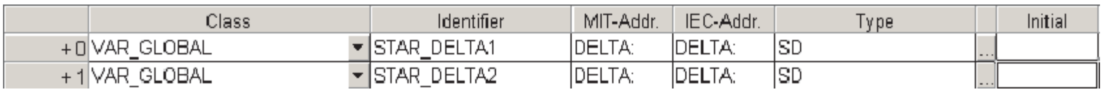
Data unit variable addresses

STAR_DELTA1 (SD)

Name	Type	MIT-Addr.	IEC-Addr.
DELTA	BOOL	Y00	%QX0
O_L	BOOL	Y01	%QX1
STAR	BOOL	Y02	%QX2
START	BOOL	X00	%IX0
STOP	BOOL	X01	%IX1
TB	INT	D2	%MW0.2
TV	INT	D3	%MW0.3

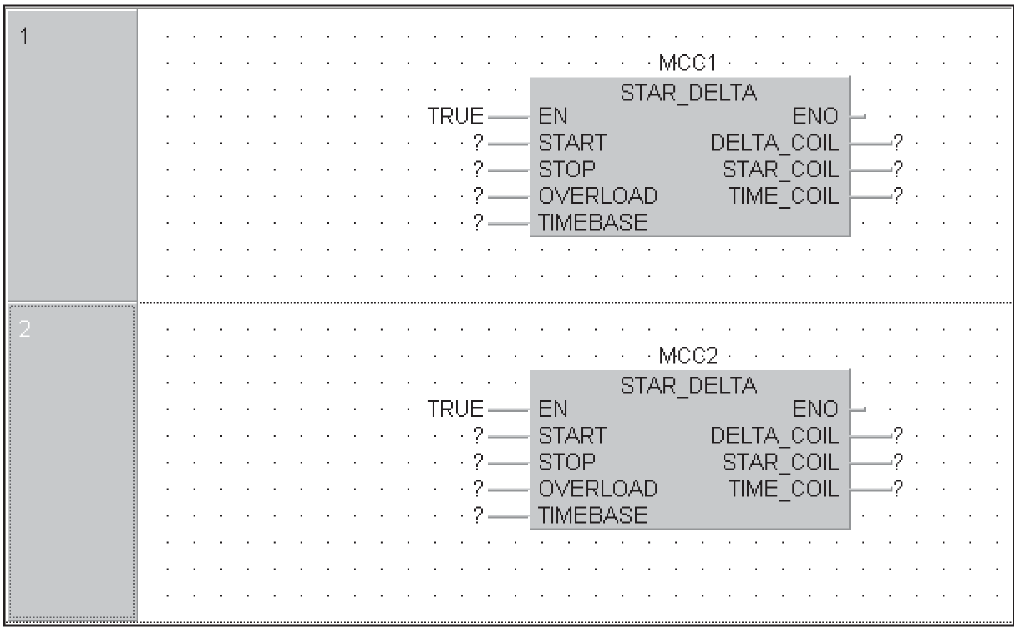
☑ Automatic filling □ All Types

⑦ Examine the GVL, it should read as follows:



	Class	Identifier	MIT-Addr.	IEC-Addr.	Type		Initial
+0	VAR_GLOBAL	STAR_DELTA1	DELTA:	DELTA:	SD	...	
+1	VAR_GLOBAL	STAR_DELTA2	DELTA:	DELTA:	SD	...	

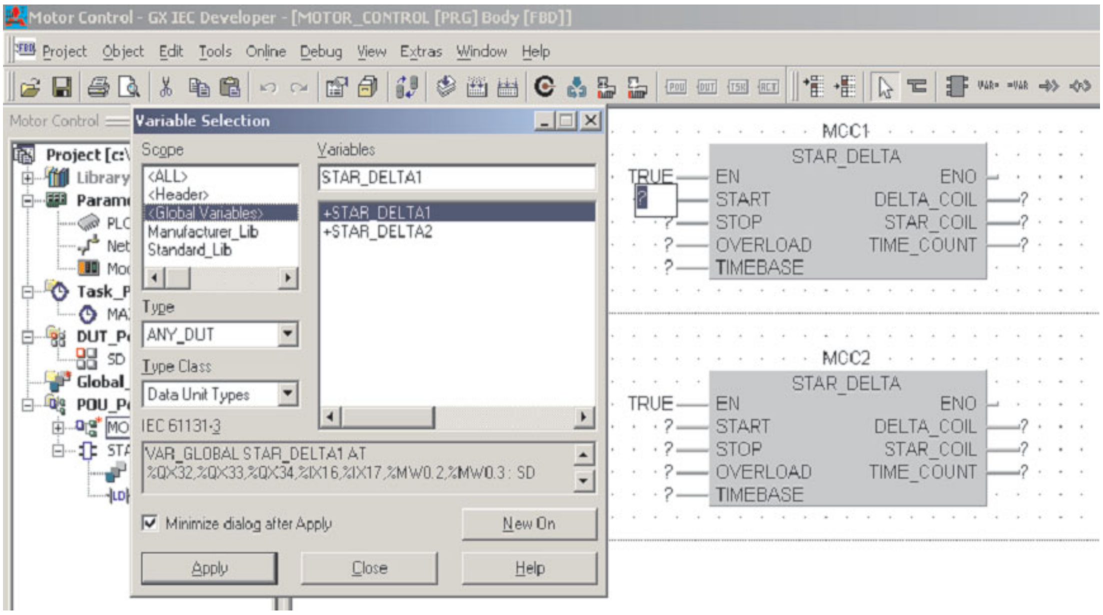
Open the MOTOR_CONTROL program POU and place 2 instances of the user created Function Block STAR_DELTA as shown:



10.3 Assigning DUT Variables to Function Blocks

To assign variables to the Function blocks...

① ...right Click on a variable (or F2). The following variable selection window appears:



Variables: STAR_DELTA1

Type: ANY_DUT

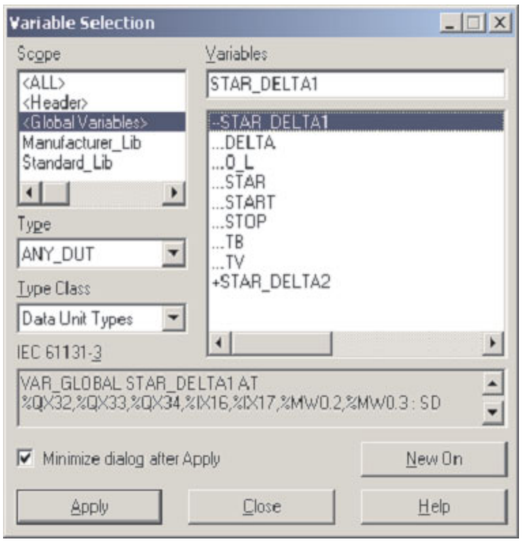
Type Class: Data Unit Types

VAR_GLOBAL STAR_DELTA1 AT %QX32,%QX33,%QX34,%IX16,%IX17,%MW0.2,%MW0.3 : SD

☑ Minimize dialog after Apply

② Set the ***Scope*** to ***Header***, ***Type Class*** to ***Data Unit Types*** and ***Type*** to ***ANY_DUT***.

③ Double Click on +STAR_DELTA1 and the following expanded DUT variable list appears:



Variables: STAR_DELTA1

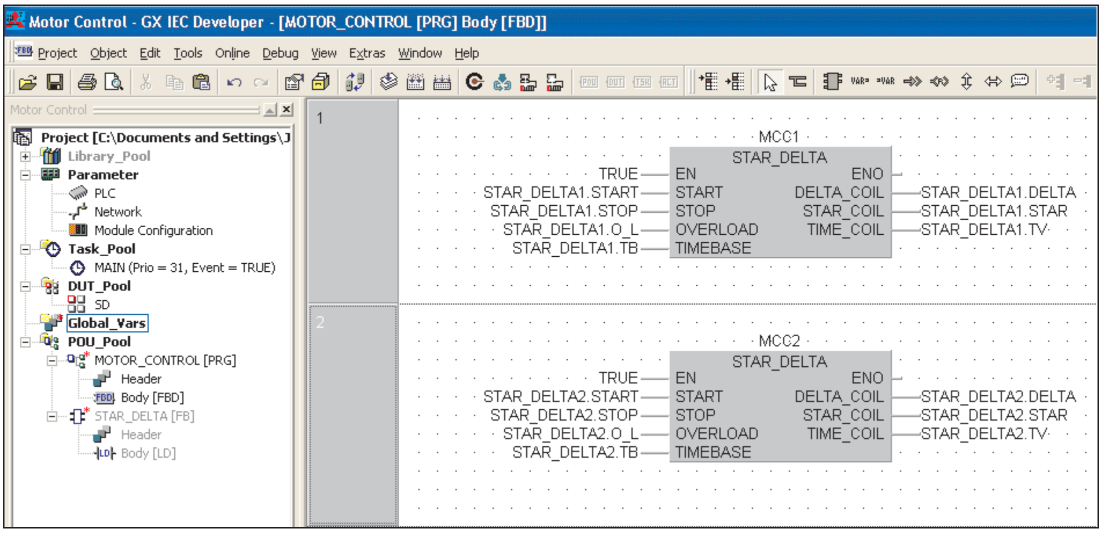
Type: ANY_DUT

Type Class: Data Unit Types

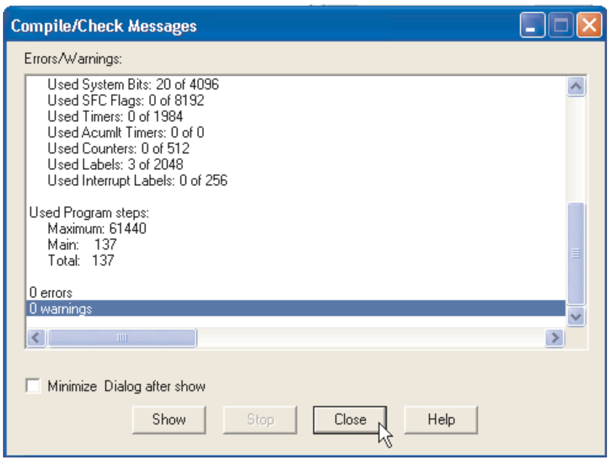
VAR_GLOBAL STAR_DELTA1 AT %QX32,%QX33,%QX34,%IX16,%IX17,%MW0.2,%MW0.3 : SD

☑ Minimize dialog after Apply

④ Pick and assign the variables to the two STAR_DELTA Function Blocks on the MOTOR_CONTROL Program POU as shown:



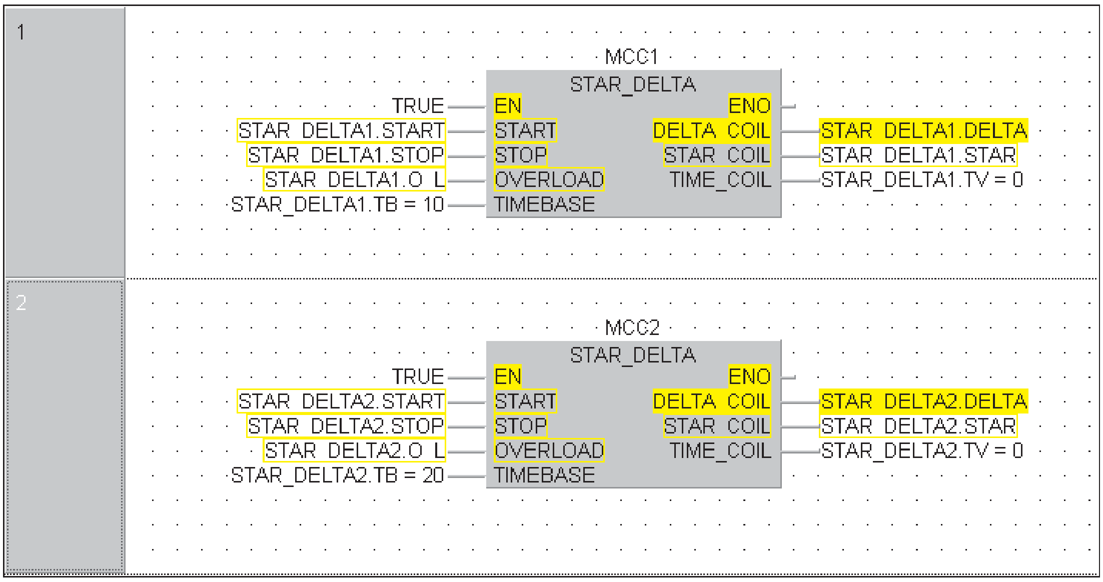
Save the project and ***Rebuild All*** to compile the code:



```
    Used System Bits: 20 of 4096
    Used SFC Flags: 0 of 8192
    Used Timers: 0 of 1984
    Used Acumlt Timers: 0 of 0
    Used Counters: 0 of 512
    Used Labels: 3 of 2048
    Used Interrupt Labels: 0 of 256

Used Program steps:
    Maximum: 61440
    Main:    137
    Total:   137

0 errors
0 warnings
```

Download and monitor the project. Before the Function Blocks can operate, it is necessary to write values into the TIMEBASE inputs: STAR_DELTA1.TB and STAR_DELTA2.TB. This is carried out by using the online variable modification technique described in an earlier section.

Simulate the operation of both Function Blocks as shown on the next page in order to confirm that everything functions as expected:


1
MCC1
STAR_DELTA
TRUE
EN
ENO
STAR DELTA1.START
START
DELTA COIL
STAR DELTA1.DELTA
STAR DELTA1.STOP
STOP
STAR COIL
STAR DELTA1.STAR
STAR DELTA1.O L
OVERLOAD
TIME_COIL
STAR_DELTA1.TV = 0
STAR_DELTA1.TB = 10
TIMEBASE
2
MCC2
STAR_DELTA
TRUE
EN
ENO
STAR DELTA2.START
START
DELTA COIL
STAR DELTA2.DELTA
STAR DELTA2.STOP
STOP
STAR COIL
STAR DELTA2.STAR
STAR DELTA2.O L
OVERLOAD
TIME_COIL
STAR_DELTA2.TV = 0
STAR_DELTA2.TB = 20
TIMEBASE

11 Arrays

11.1 Overview

An array is a field or matrix of variables, of a particular type.

For example, an **ARRAY [0..2] OF INT**, is a one dimensional array of three integer elements (0,1,2). If the start address of the array is D0, then the array consists of D0, D1 and D2.

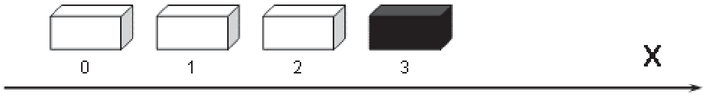
Identifier	Address	Type	Length
Motor_Volts	D0	ARRAY	[0...2] OF INT

In software, program elements can use: Motor_Volts[1] and Motor_Volts[2], as declarations, which in this example mean that D1 and D2 are addressed.

Arrays can have up to three dimensions, for example: ARRAY [0...2, 0...4] has three elements in the first dimension and five in the second. Arrays can provide a convenient way of 'indexing' tag names, i.e. one declaration in the Local or Global Variable Table can access many elements.

The following diagrams illustrate graphical representation of the three array types.

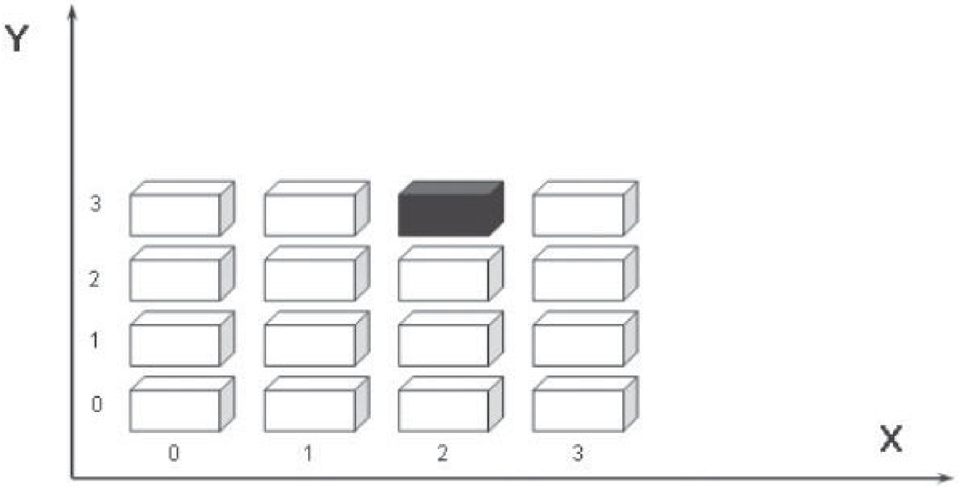
Single Dimensional Array



Identifier	Type
Motor_Speed	ARRAY [0..3] OF INT

■ = Motor_Speed [3]

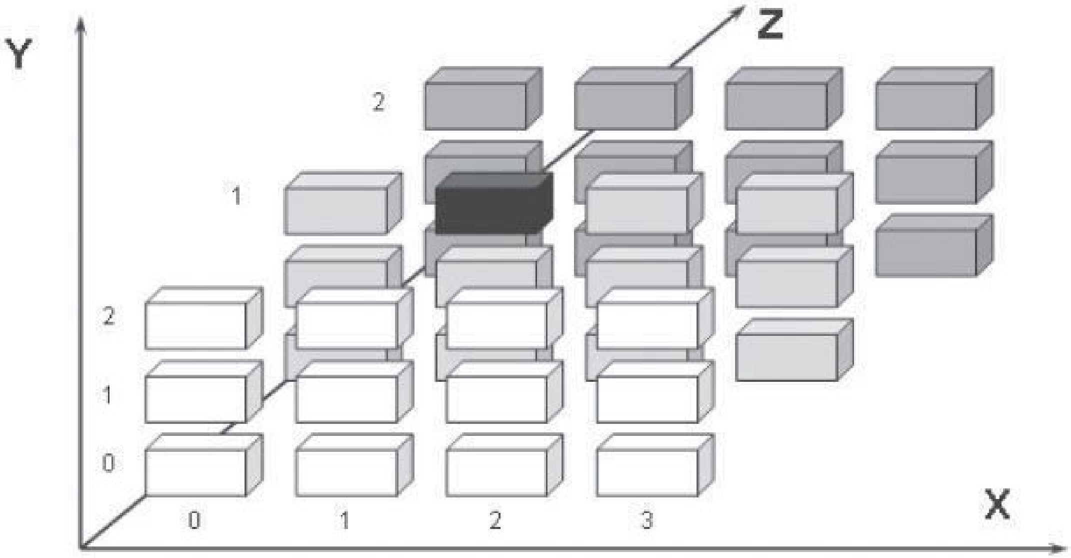
Two Dimensional Array



Identifier	Type
Motor_Volt	**ARRAY [0..3, 0...3] OF INT**

■ **= Motor_Volt [2, 3]**

Three Dimensional Array



Identifier	Type
Motor_Current	**ARRAY [0..3, 0...2, 0..2] OF INT**

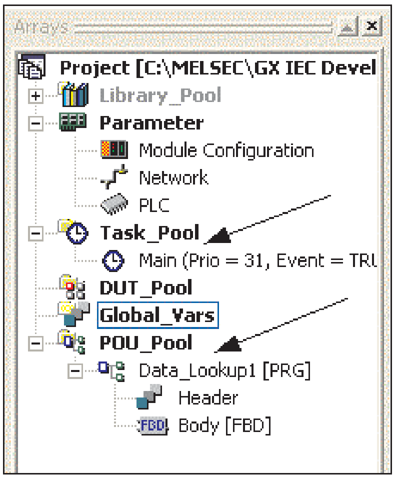
■ **= Motor_Current [1, 2, 1]**

11.2 Array Example: Single Dimension Array

The following example is used to illustrate a single dimension array. The array is 10 words long and uses Global MELSEC addresses D100 to D109. This example uses only “Standard IEC” Operators, Functions and Function Blocks.

① Create a new project and define one new POU of class “Program” using a body of language ***FBD*** and named “Data_Lookup1”

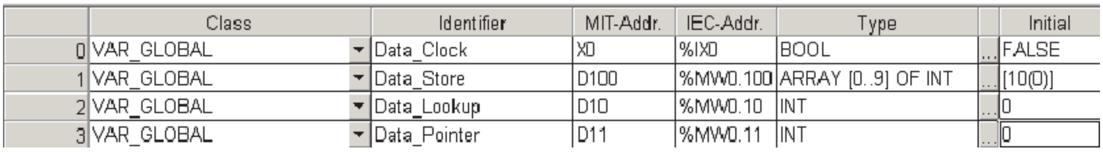
② Create a new Task in the task pool named “Main” and bind the program POU “Data_Lookup1” to it:



Arrays

Project [C:\MELSEC\GX IEC Devel
Library_Pool
Parameter
Module Configuration
Network
PLC
Task_Pool
Main (Prio = 31, Event = TRU
DUT_Pool
Global_Vars
POU_Pool
Data_Lookup1 [PRG]
Header
Body [FBD]

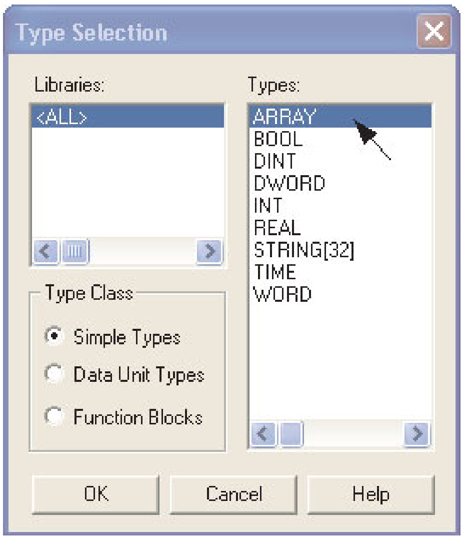
③ Open the Global Variables list and create the following entries:

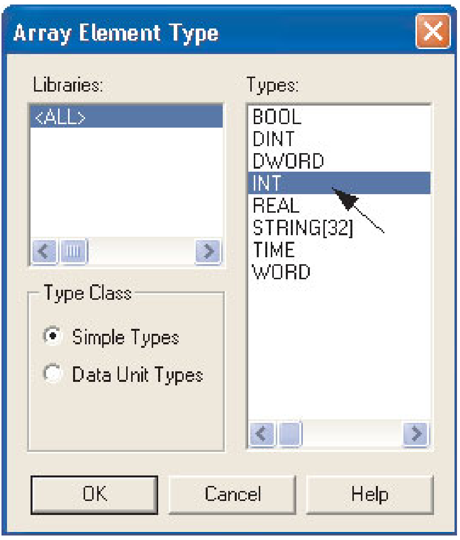


	Class	Identifier	MIT-Addr.	IEC-Addr.	Type	Initial
0	VAR_GLOBAL	Data_Clock	X0	%IX0	BOOL	FALSE
1	VAR_GLOBAL	Data_Store	D100	%MW0.100	ARRAY [0..9] OF INT	[10(0)]
2	VAR_GLOBAL	Data_Lookup	D10	%MW0.10	INT	0
3	VAR_GLOBAL	Data_Pointer	D11	%MW0.11	INT	0

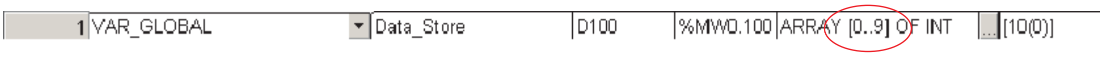
NOTE

The variable type “Array” in entered as follows:



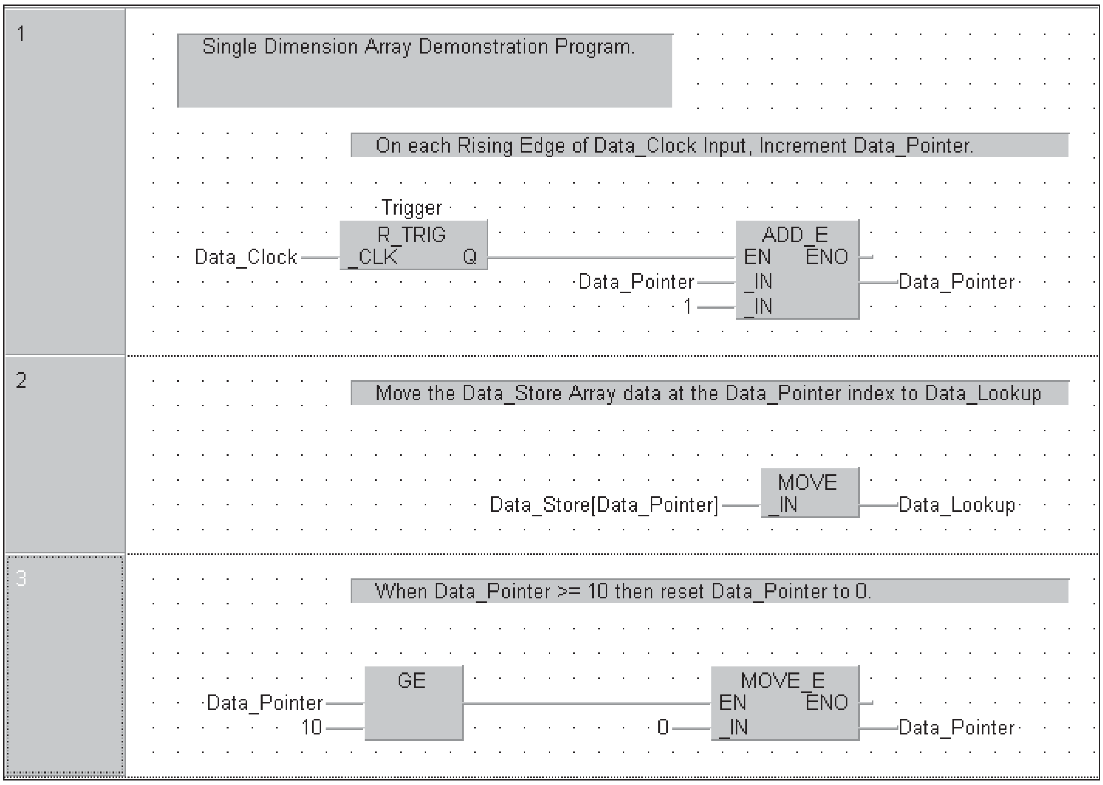


Note that when the array entry first appears, it will be dimensioned to the default value of ARRAY [0..3] OF INT. It is necessary to re dimension it to [0..9] of INT for this example, as shown below:



1	VAR_GLOBAL	Data_Store	D100	%MW0.100	ARRAY [0..9] OF INT	... [10(0)]

④ Open the Program POU “Data_Lookup1” and enter the following Function Block Diagram:



1

Single Dimension Array Demonstration Program.

On each Rising Edge of Data_Clock Input, Increment Data_Pointer.

2

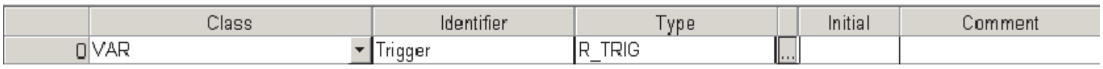
Move the Data_Store Array data at the Data_Pointer index to Data_Lookup

3

When Data_Pointer >= 10 then reset Data_Pointer to 0.

NOTE Define the ‘R_Trig’ Function block with instance name “Trigger”.

⑤ Check the Header reads as shown below:

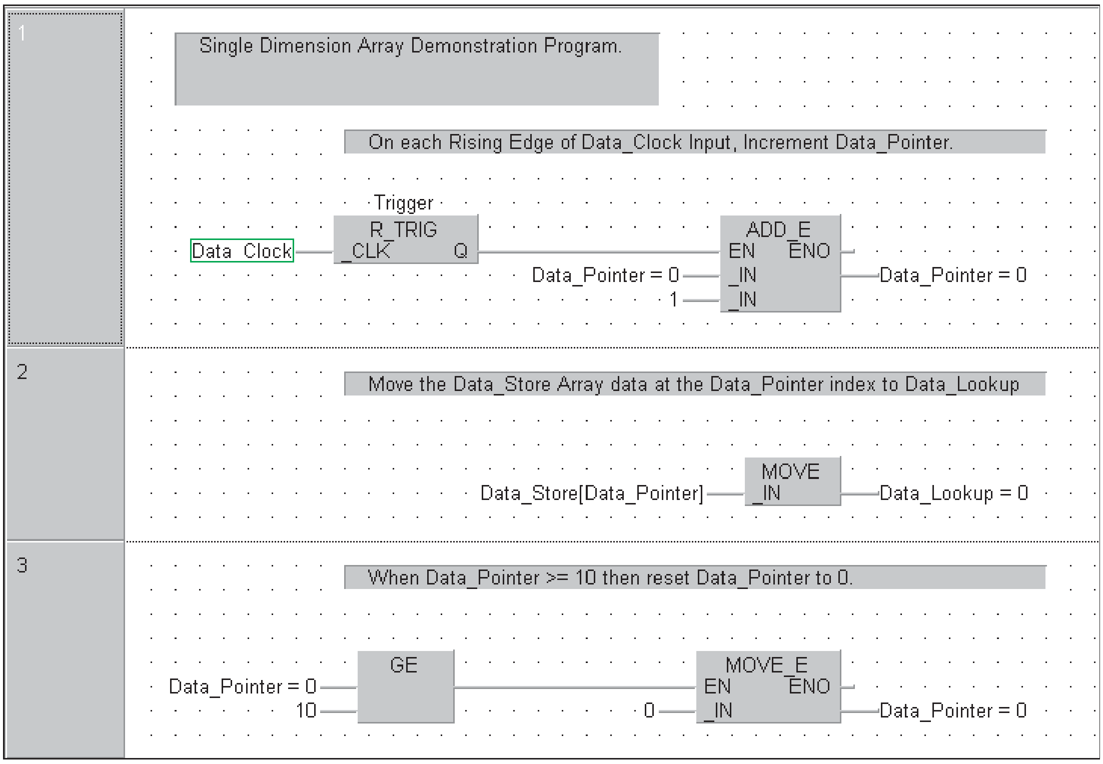


	Class	Identifier	Type		Initial	Comment
0	VAR	Trigger	R_TRIG	...		

⑥ Save the program and use ***Rebuild All*** to compile the program.

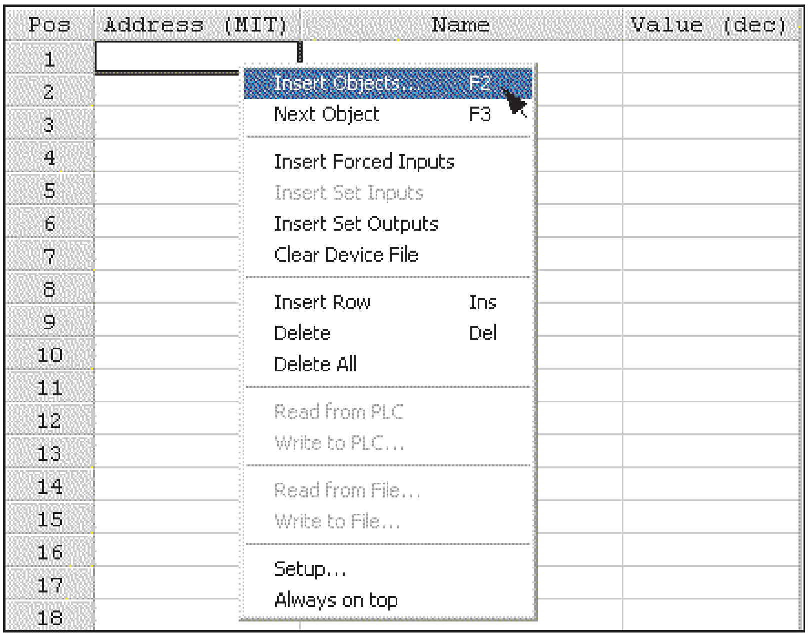
⑦ Transfer the program to the PLC.

⑧ Monitor the POU body (see next page).

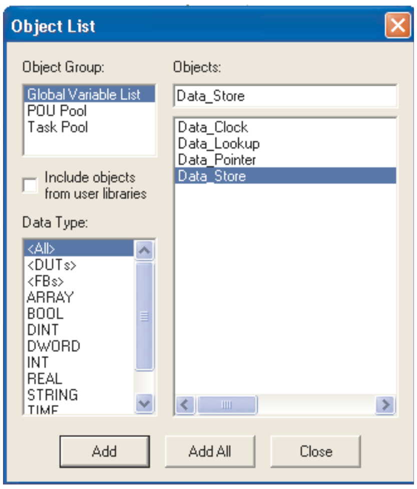

1
Single Dimension Array Demonstration Program.
On each Rising Edge of Data_Clock Input, Increment Data_Pointer.
Trigger
R_TRIG
Data_Clock
_CLK
Q
ADD_E
EN
ENO
Data_Pointer = 0
_IN
1
_IN
Data_Pointer = 0
2
Move the Data_Store Array data at the Data_Pointer index to Data_Lookup
MOVE
Data_Store[Data_Pointer]
_IN
Data_Lookup = 0
3
When Data_Pointer >= 10 then reset Data_Pointer to 0.
GE
Data_Pointer = 0
10
MOVE_E
EN
ENO
0
_IN
Data_Pointer = 0

Before the program is able to function as intended it is necessary to input data into the physical MELSEC addresses occupied by the array variables. There are two ways in which this may be achieved:

- Use the ***Device Edit*** feature from the ***Debug*** menu as previously described, using ***Insert Devices*** in the range D100 to D109, and enter any 10 random integer values between -32768 to +32767 and write them to the PLC.
- Open the ***Entry Data Monitor*** feature from the ***Online*** menu.
  – Right Click on the ***Address*** or ***Name*** column headers and select ***Insert Objects*** from the menu list as shown:



– From the resulting window select the ***Data_Store*** variable name and click ***Add***:



Object List

Object Group: Global Variable List, POU Pool, Task Pool

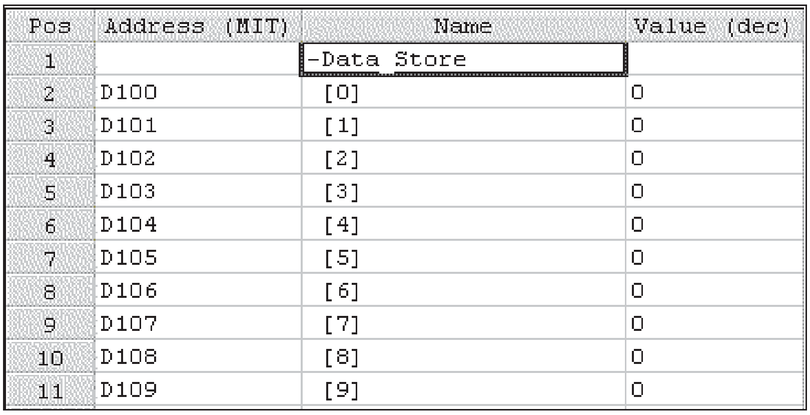
□ Include objects from user libraries

Data Type: <All>, <DUTs>, <FBs>, ARRAY, BOOL, DINT, DWORD, INT, REAL, STRING, TIME

Objects: Data_Store

Data_Clock, Data_Lookup, Data_Pointer, Data_Store

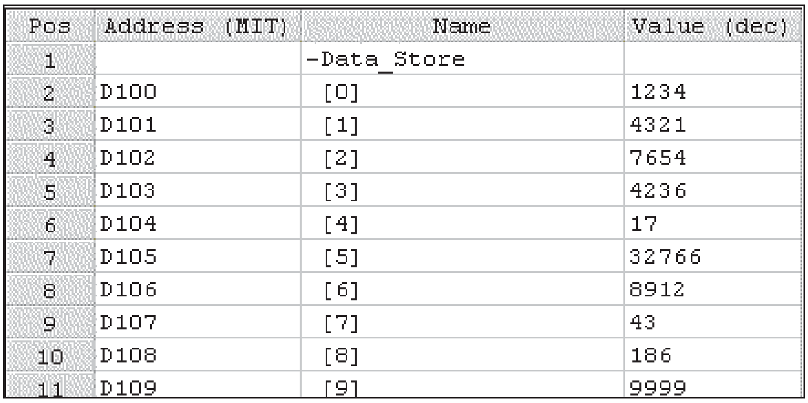
– Because the variable name “Data_Store” is an array, the system presents the entry with a “+“ prefix. Clicking on the variable name expands the array details into the table as shown:



Pos	Address (MIT)	Name	Value (dec)
1		-Data Store	
2	D100	[0]	0
3	D101	[1]	0
4	D102	[2]	0
5	D103	[3]	0
6	D104	[4]	0
7	D105	[5]	0
8	D106	[6]	0
9	D107	[7]	0
10	D108	[8]	0
11	D109	[9]	0

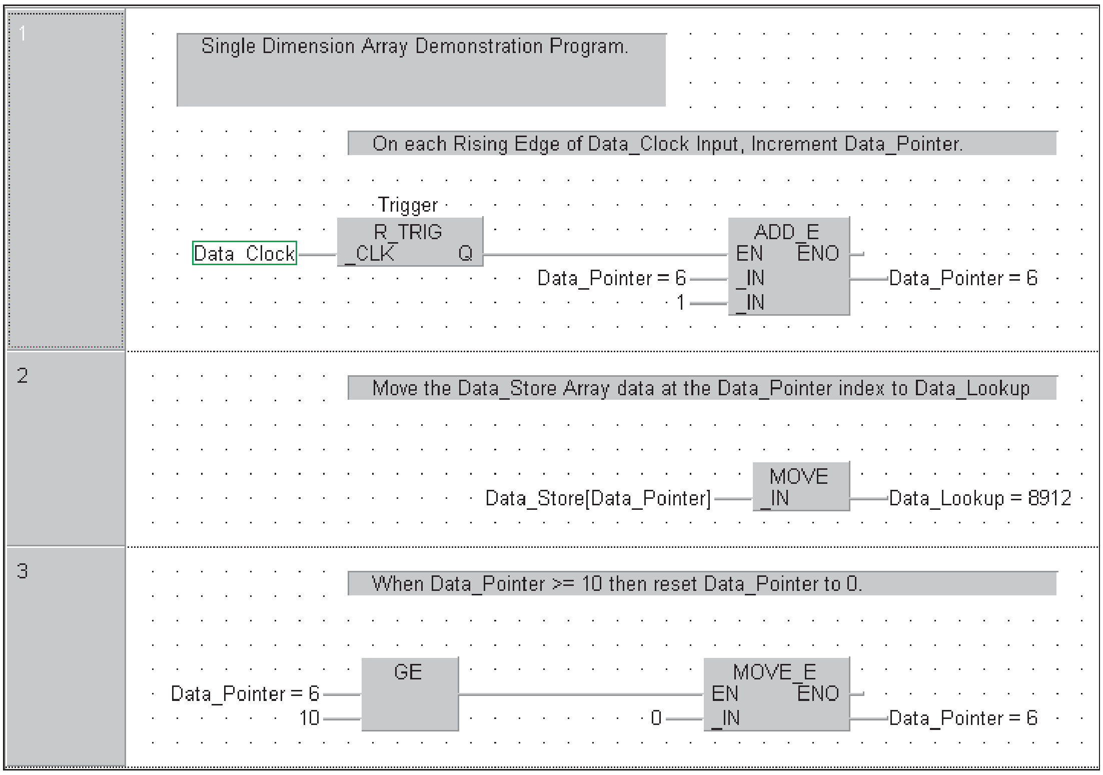
– Clicking on the “-“ Prefix collapses the array details.

– While monitoring the variable values, enter any 10 random integer values between -32768 to +32767 as shown below:



Pos	Address (MIT)	Name	Value (dec)
1		-Data_Store	
2	D100	[0]	1234
3	D101	[1]	4321
4	D102	[2]	7654
5	D103	[3]	4236
6	D104	[4]	17
7	D105	[5]	32766
8	D106	[6]	8912
9	D107	[7]	43
10	D108	[8]	186
11	D109	[9]	9999

– Switch back to monitor the body of the POU “Data_Lookup1” and observe the operation of the program, noting how the value alters on the output variable “Data_Lookup” as the data pointer increases:



1

Single Dimension Array Demonstration Program.

On each Rising Edge of Data_Clock Input, Increment Data_Pointer.

2

Move the Data_Store Array data at the Data_Pointer index to Data_Lookup

3

When Data_Pointer >= 10 then reset Data_Pointer to 0.

- The program is designed to reset the pointer to zero on the 10th element and thus will repeat scan the table with an upward increment (Index 0-9).

12 Working with Libraries

12.1 User Defined Libraries

All Functions and Function Blocks, created so far, have been resident in the current project and only available to that project.

User defined libraries, allow the creation of libraries containing user created POU's, Functions, Function Blocks etc. These libraries are available globally, i.e. can be accessed by other projects.

Therefore, engineers working with separate projects can have access to common libraries of standard circuit parts.

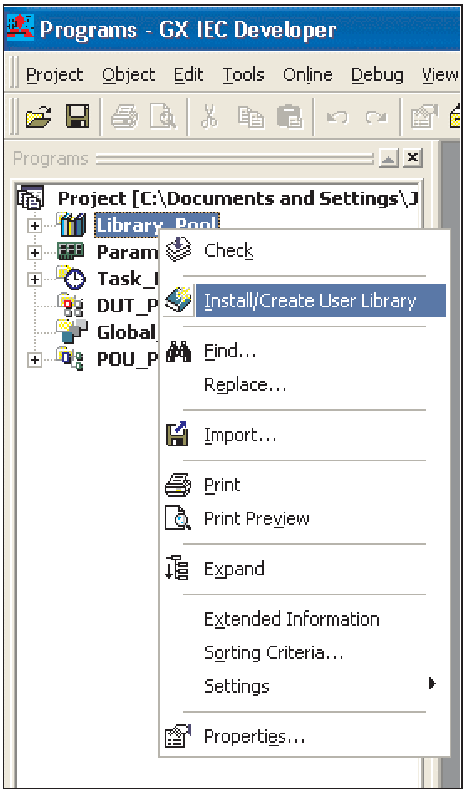
As already seen, when called program functions, the ***Standard Library*** contains IEC functions. The ***Manufacturer Library*** contains Mitsubishi functions (denoted by *_M) – M meaning manufacturer, not Mitsubishi!

Any user defined libraries will also appear on this list.

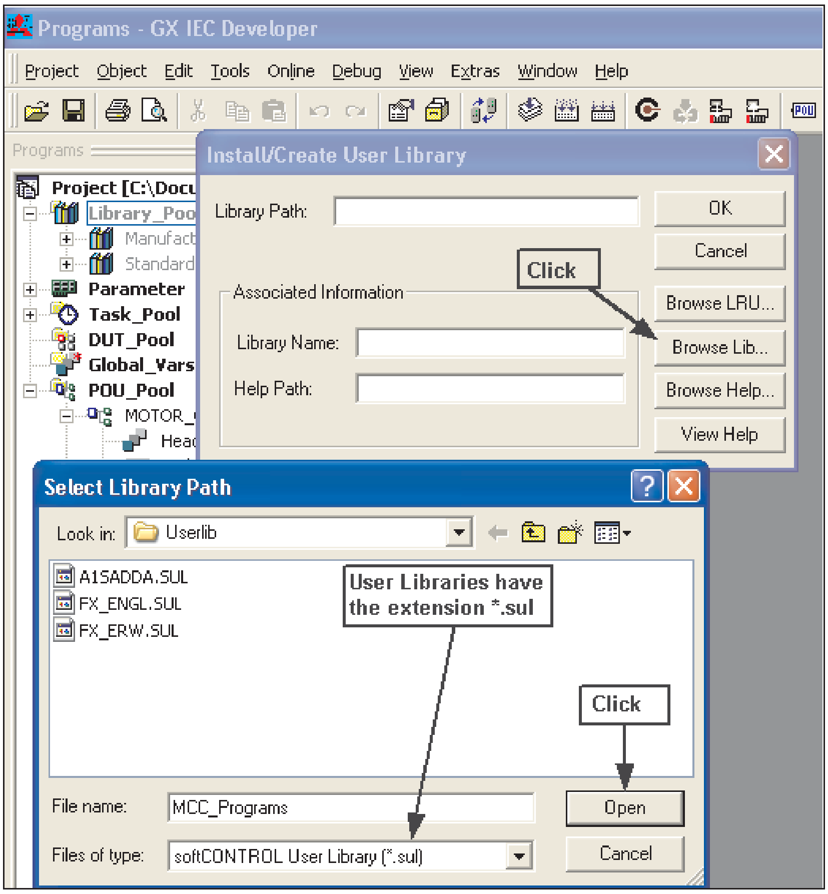
12.1.1 Example – Creating a new Library

① Assign the function block STAR_DELTA to a new library.

② Right Click the Library Pool, in the Project Navigator window and from the displayed menu select ***User Library*** and ***Install/Create Library***.



③ Click on ***Browse Lib*** and enter a file name “MCC_Programs” into the window below. The directory path can be changed if desired. In this case it is suggested that the default path is used. This being: “C:\MELSEC\GX IEC DEVELOPER 7.00\Userlib”.



Click

Look in: Userlib

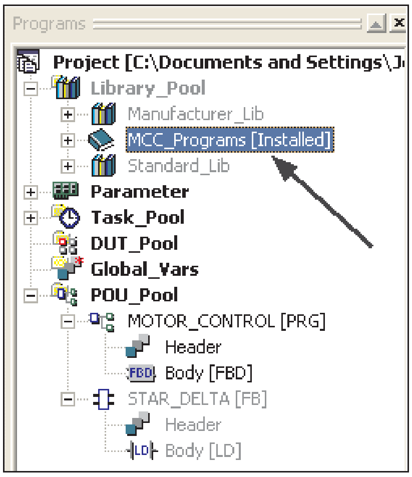
User Libraries have the extension *.sul

Click

File name: MCC_Programs

Files of type: softCONTROL User Library (*.sul)

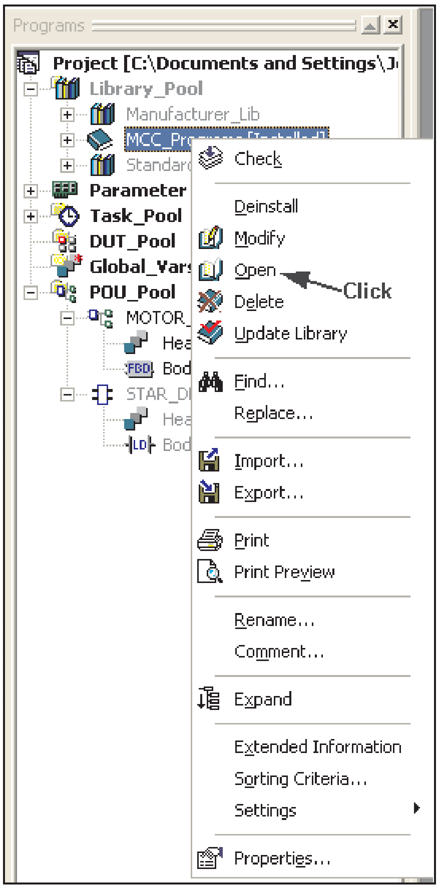
④ Click ***Open*** when done:



Notice the new Library “MCC_Programs” that is now present in the project Library Pool.

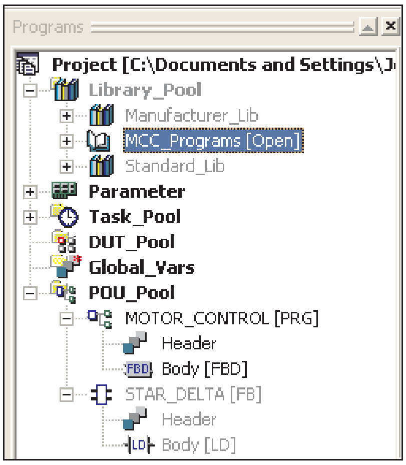
12.1.2 Opening the Library

① Open the Library by right clicking on the icon ‘MCC_Programs’ and click on ***Open*** from the menu:



Click

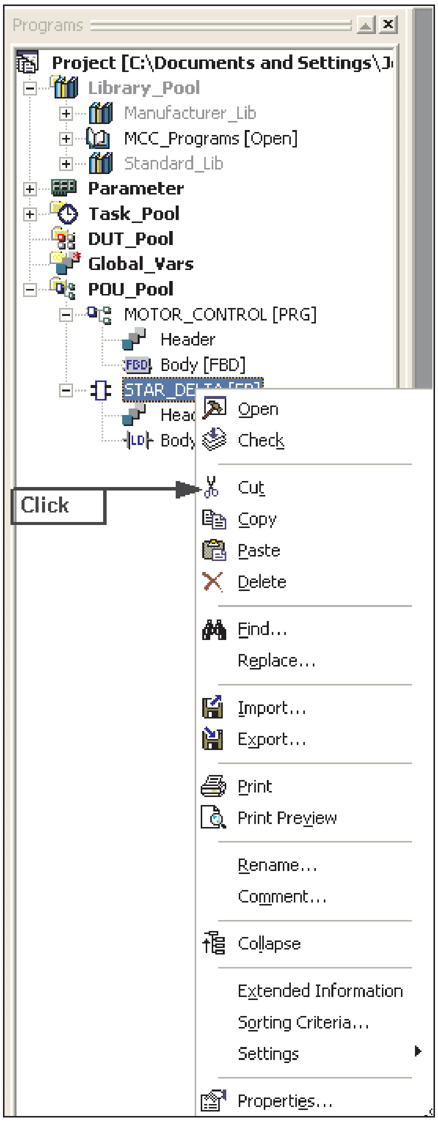
The Library is now open and may be accessed and edited:



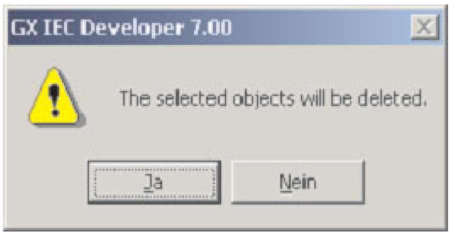
12.1.3 Moving a POU “Function Block” to an open Library

The Function Block STAR_DELTA will now be moved into the Library ‘MCC_Programs’.

① Right click on the STAR_DELTA icon in the Project navigation window and click on ***Cut***:



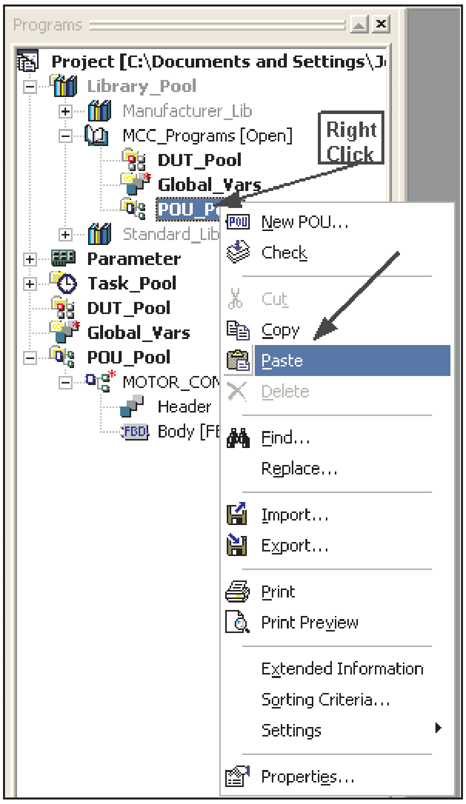
The following dialogue will be displayed:


GX IEC Developer 7.00

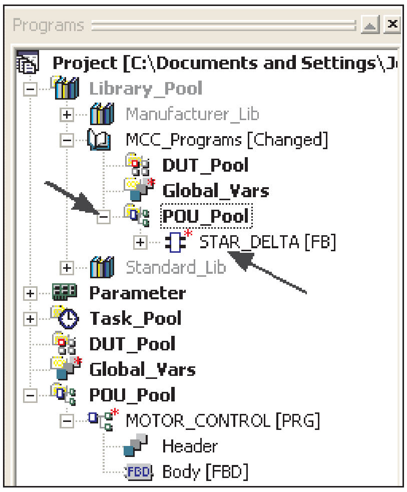
The selected objects will be deleted.

② Select ***Yes***

③ Right Click on the User Library icon and select ***Paste*** from the menu:



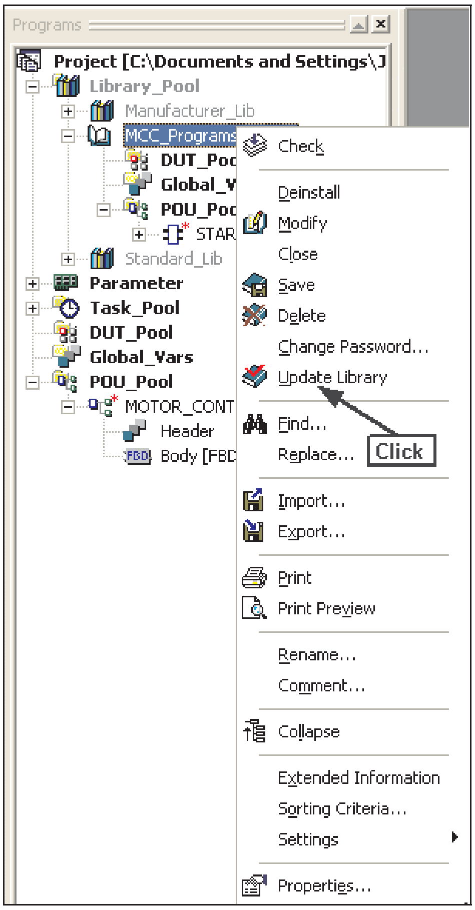
④ Click on the '+' on the new entry in the Library POU Pool to expand the 'STAR_DELTA' Function Block:



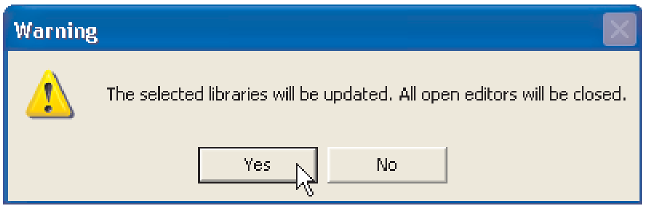
The Function Block POU, "STAR_DELTA" is now present in the Library "MCC_Programs" and no longer in the Project POU Pool.

Any POU, Function, Function Block, PRG or DUT can be added to the library in this way.

⑤ When editing of the library is complete, click ***Update Library***. This will update and close the library.

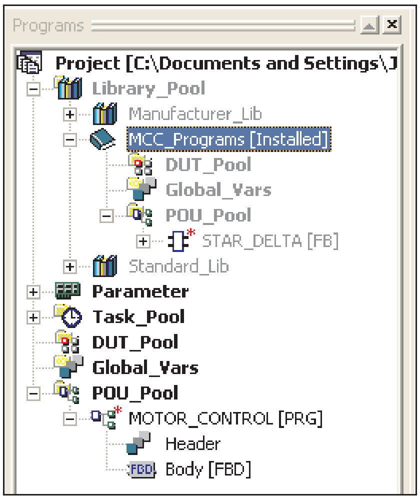


The following message will be displayed:



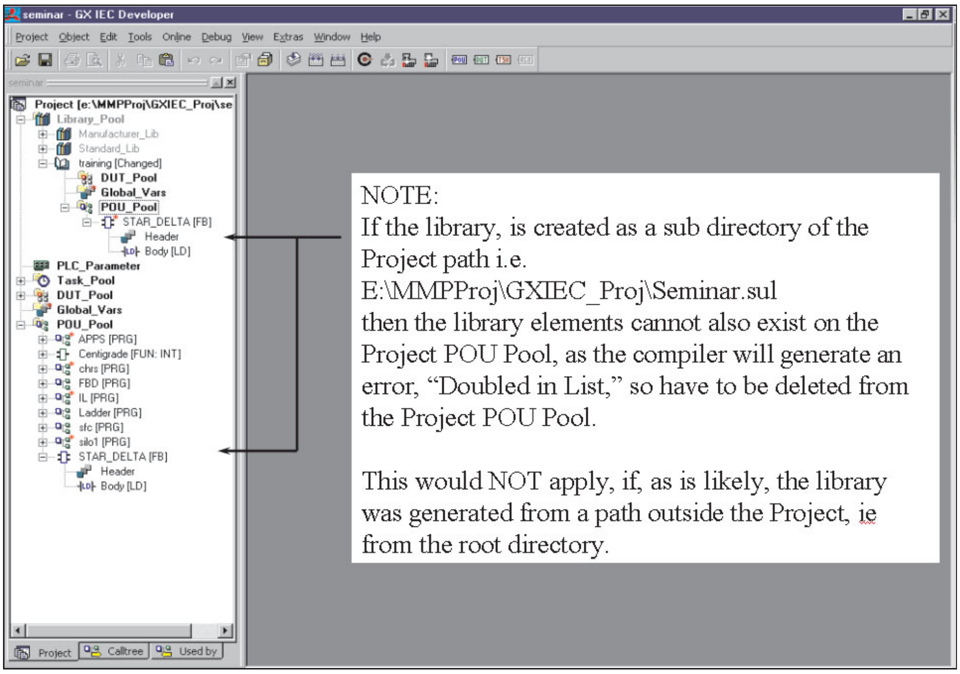
The selected libraries will be updated. All open editors will be closed.

⑥ Click ***Yes*** and the library will be updated, saved and closed.



The library is now stored in the default location of “C:\MELSEC\GX IEC DEVELOPER 7.00\Userlib” as set when creating the library.

12.2 Special Note about Libraries



NOTE:
If the library, is created as a sub directory of the Project path i.e.
E:\MMPProj\GXIEC_Proj\Seminar.sul
then the library elements cannot also exist on the Project POU Pool, as the compiler will generate an error, “Doubled in List,” so have to be deleted from the Project POU Pool.

This would NOT apply, if, as is likely, the library was generated from a path outside the Project, ie from the root directory.

12.3 Importing Libraries into Projects

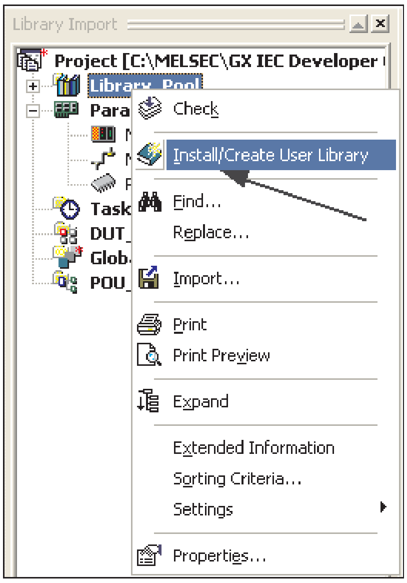
Once ‘User Libraries’ have been created, it is possible to re-use routines by importing them into other applications. Mitsubishi Electric has produced many Libraries of commonly used routines. For example, ‘Intelligent Module’ interfaces such as A/D and D/A Function Blocks containing all the code to facilitate a working interface for these and many more modules. These Function Blocks are available free on many of the Mitsubishi web sites and some are provided on the GX IEC Developer Master Disk.

The following two examples describe the methods used to import libraries into working applications.

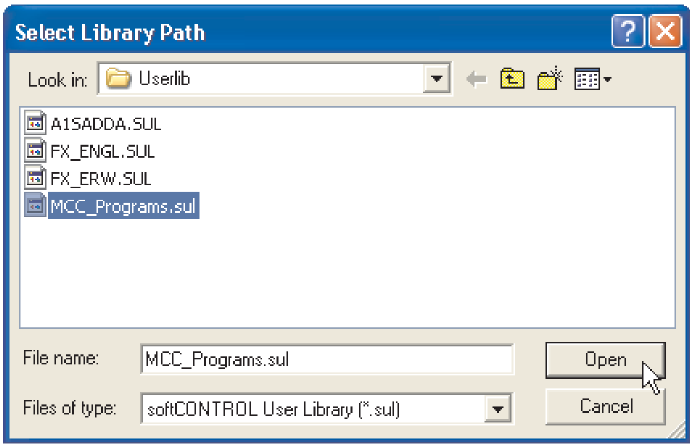
12.3.1 Import of an User Library

The previously saved library “MCC_Programs” will be imported into the current project and the Function Block contained therein will be re-used.

① Create a new empty project with no POU’s called “Library Import”.



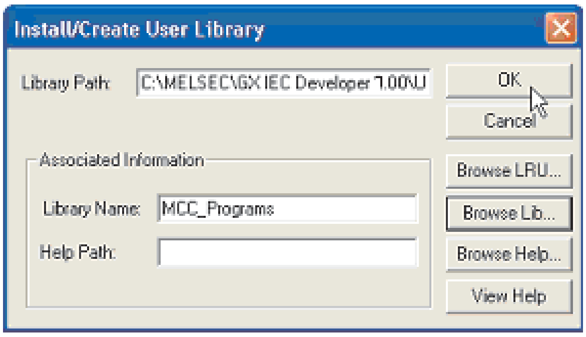
② Enter the following details into the prompt:



File name: MCC_Programs.sul

Files of type: softCONTROL User Library (*.sul)

③ Next click ***OK*** to accept the entries.



Install/Create User Library

Library Path: C:\MELSEC\GX IEC Developer 7.00\U

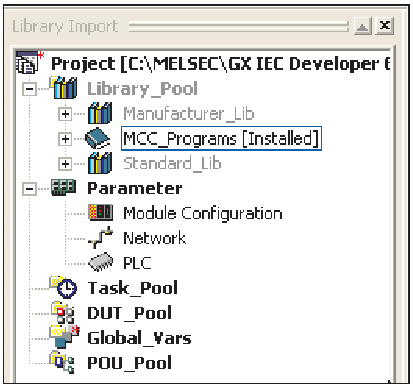
Associated Information

Library Name: MCC_Programs

Help Path:

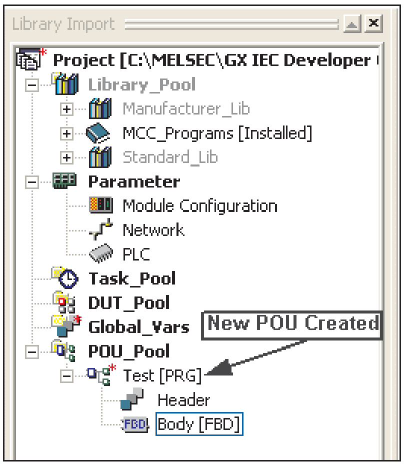
NOTE The help path is used for user help files that can be created in order to describe the operation of routines held in the library. These files can be created in MS-Word, for example in HTML format and manually saved with the reserved extension *.CHM. These files can be bound to the library by clicking ***Browse Help*** in the same manner as the ***Library Name*** selection illustrated above.

The new imported library is now installed into the application and can now be used within the project as shown:

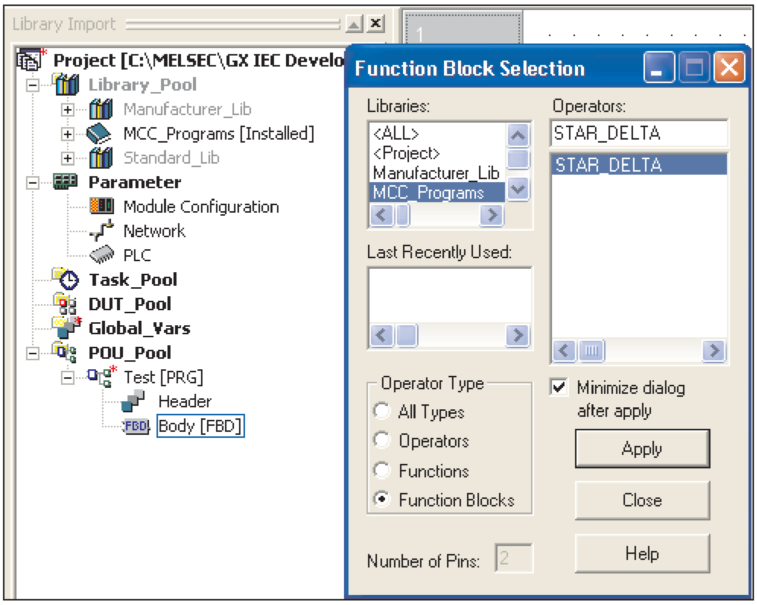


Items stored in libraries can be easily recalled and selected into a project, as shown in the following illustrations:

① Create a new POU, type: ***FBD*** and named “Test”:



② Open the new POU and select the Function Block as shown:



Function Block Selection

Libraries: <ALL> <Project> Manufacturer_Lib MCC_Programs

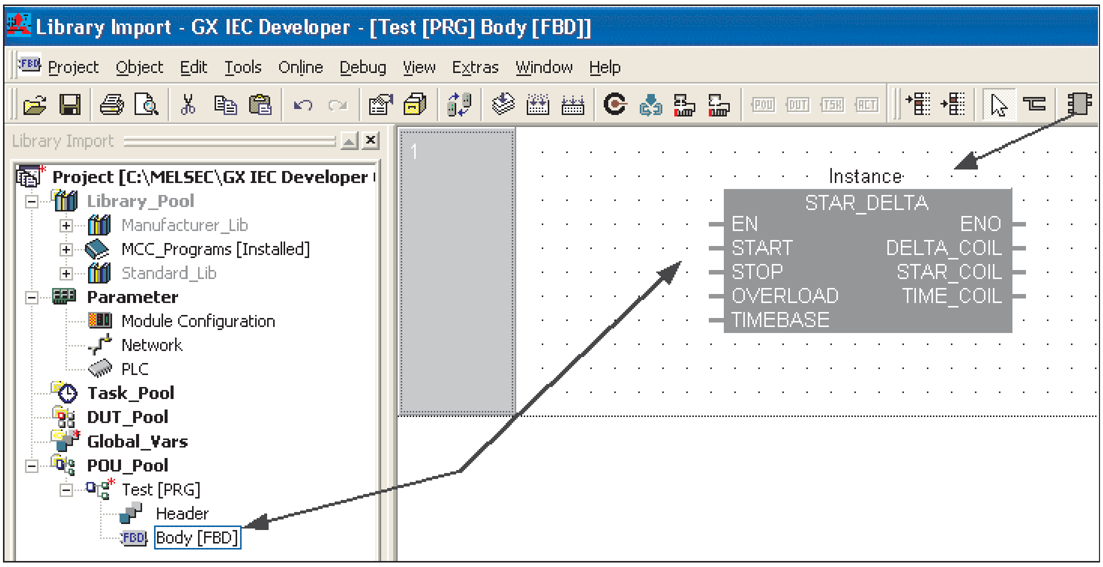
Operators: STAR_DELTA

Operator Type: ○ All Types ○ Operators ○ Functions ● Function Blocks

☑ Minimize dialog after apply

Number of Pins: 2

As can be seen the new library appears in the domain and may be selected as shown:



12.3.2 Importing a Mitsubishi Library Function Block

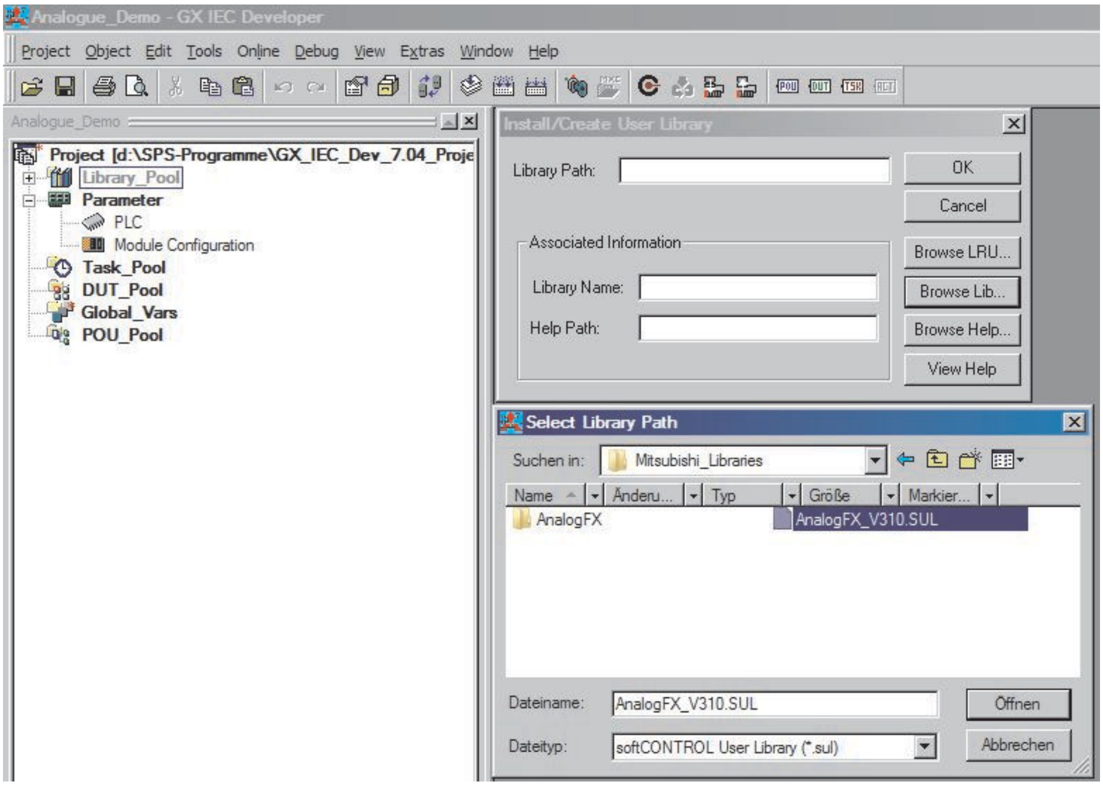
The following illustrations demonstrate the procedures required to import a Mitsubishi function block for analogue input using a module FX3U-4AD.

This function block is provided by Mitsubishi. In order for the following example to function correctly, it is necessary to install the library "AnalogFX" into the project. The library is to be found on the Mitsubishi website (www.mitsubishi-automation.com). After the installation the library can be accessed from the “Userlib” directory.

① Create a new empty project with no POU’s called “Analogue_Demo”.

② Create a new POU (Type: ***FBD***, Class: ***PRG***) and name it “Analogue_Input”

③ Right click on the ***Library_Pool*** icon and then on ***Install/Create User Library***. This opens a new dialogue window. Select ***Browse Lib***. Select the AnalogFX_V310.SUL library file and click ***Open***.



Project [d:\SPS-Programme\GX_IEC_Dev_7.04_Proje

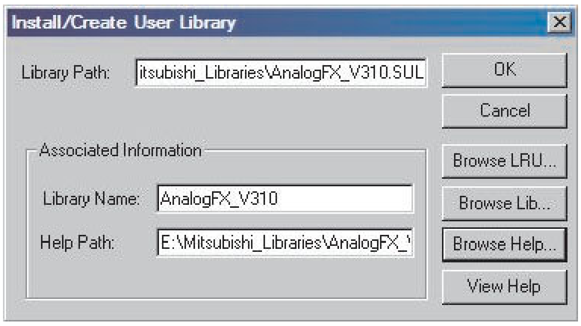
Suchen in: Mitsubishi_Libraries

Dateiname: AnalogFX_V310.SUL

Dateityp: softCONTROL User Library (*.sul)

You may also select the accompanying library help file by clicking on ***Browse Help***.

④ Click ***OK*** on the ***Install/Create User Library*** prompt:

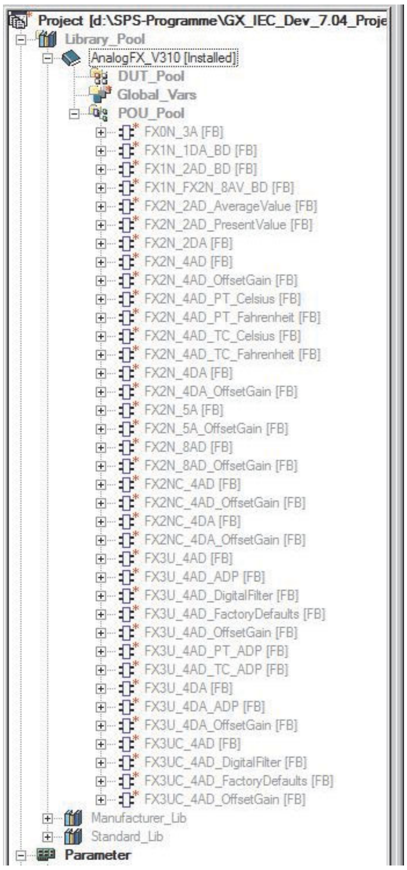


Library Path: itsubishi_Libraries\AnalogFX_V310.SUL

Library Name: AnalogFX_V310

Help Path: E:\Mitsubishi_Libraries\AnalogFX_\

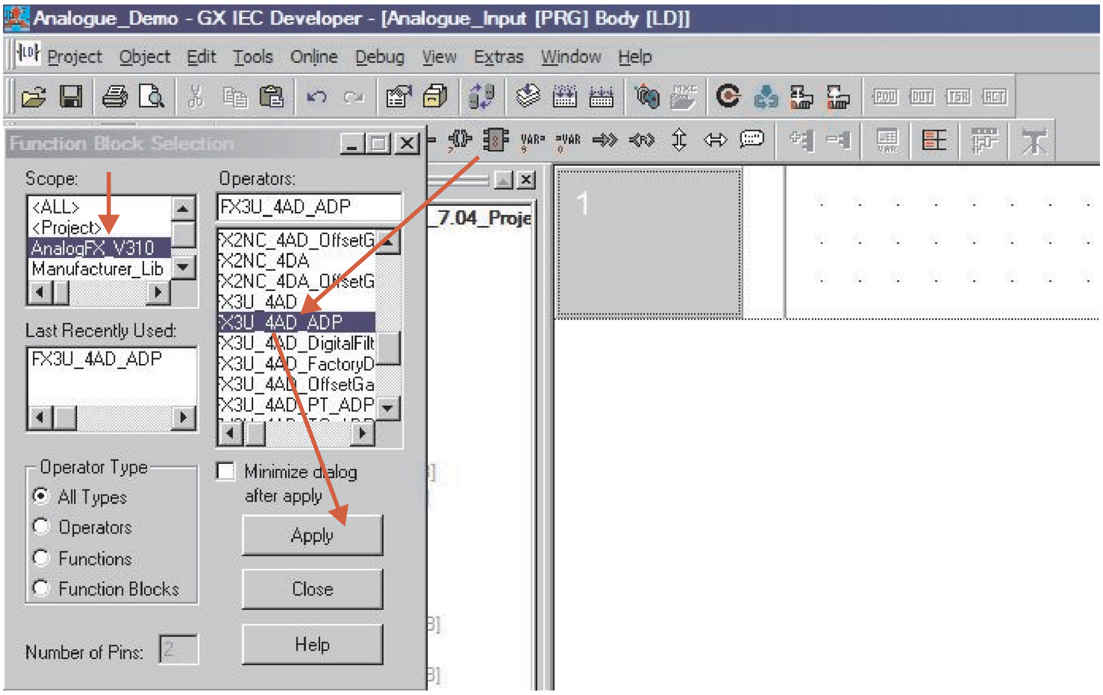
Note the new “AnalogFX_V310” library in the Project Navigation Window.



```
Project [d:\SPS-Programme\GX_IEC_Dev_7.04_Proje
  Library_Pool
    AnalogFX_V310 [Installed]
      DUT_Pool
      Global_Vars
      POU_Pool
        FX0N_3A [FB]
        FX1N_1DA_BD [FB]
        FX1N_2AD_BD [FB]
        FX1N_FX2N_8AV_BD [FB]
        FX2N_2AD_AverageValue [FB]
        FX2N_2AD_PresentValue [FB]
        FX2N_2DA [FB]
        FX2N_4AD [FB]
        FX2N_4AD_OffsetGain [FB]
        FX2N_4AD_PT_Celsius [FB]
        FX2N_4AD_PT_Fahrenheit [FB]
        FX2N_4AD_TC_Celsius [FB]
        FX2N_4AD_TC_Fahrenheit [FB]
        FX2N_4DA [FB]
        FX2N_4DA_OffsetGain [FB]
        FX2N_5A [FB]
        FX2N_5A_OffsetGain [FB]
        FX2N_8AD [FB]
        FX2N_8AD_OffsetGain [FB]
        FX2NC_4AD [FB]
        FX2NC_4AD_OffsetGain [FB]
        FX2NC_4DA [FB]
        FX2NC_4DA_OffsetGain [FB]
        FX3U_4AD [FB]
        FX3U_4AD_ADP [FB]
        FX3U_4AD_DigitalFilter [FB]
        FX3U_4AD_FactoryDefaults [FB]
        FX3U_4AD_OffsetGain [FB]
        FX3U_4AD_PT_ADP [FB]
        FX3U_4AD_TC_ADP [FB]
        FX3U_4DA [FB]
        FX3U_4DA_ADP [FB]
        FX3U_4DA_OffsetGain [FB]
        FX3UC_4AD [FB]
        FX3UC_4AD_DigitalFilter [FB]
        FX3UC_4AD_FactoryDefaults [FB]
        FX3UC_4AD_OffsetGain [FB]
    Manufacturer_Lib
    Standard_Lib
  Parameter
```

⑤ Create a new task in the task pool: “MAIN” and bind the POU “Analogue_Input” to it.

⑥ Place the FX3U_4AD_ADP function block into the POU as shown below:



Analogue_Demo - GX IEC Developer - [Analogue_Input [PRG] Body [LD]]

Function Block Selection

Scope: <ALL>, <Project>, AnalogFX_V310, Manufacturer_Lib

Operators: FX3U_4AD_ADP

```
X2NC_4AD_OffsetG
X2NC_4DA
X2NC_4DA_OffsetG
X3U_4AD
X3U_4AD_ADP
X3U_4AD_DigitalFilt
X3U_4AD_FactoryD
X3U_4AD_OffsetGa
X3U_4AD_PT_ADP
```

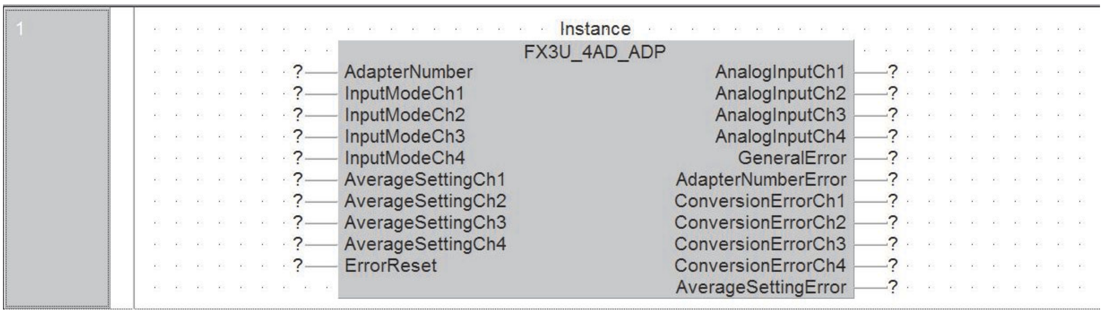
Last Recently Used: FX3U_4AD_ADP

Operator Type: ● All Types ○ Operators ○ Functions ○ Function Blocks

□ Minimize dialog after apply

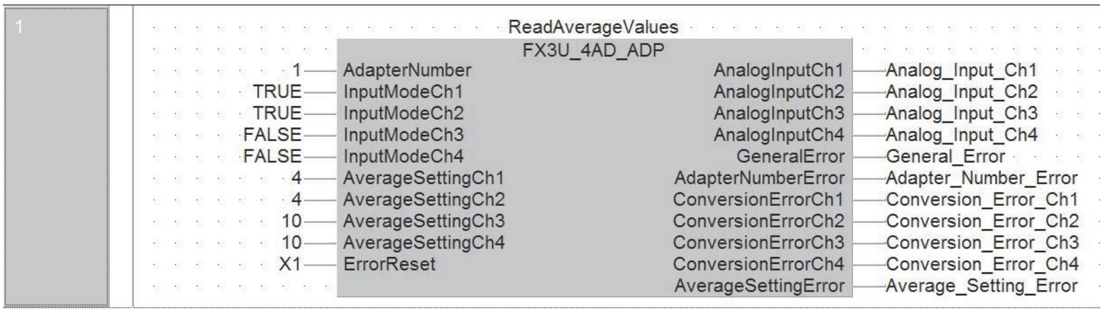
Number of Pins: 2

The function block will appear thus:



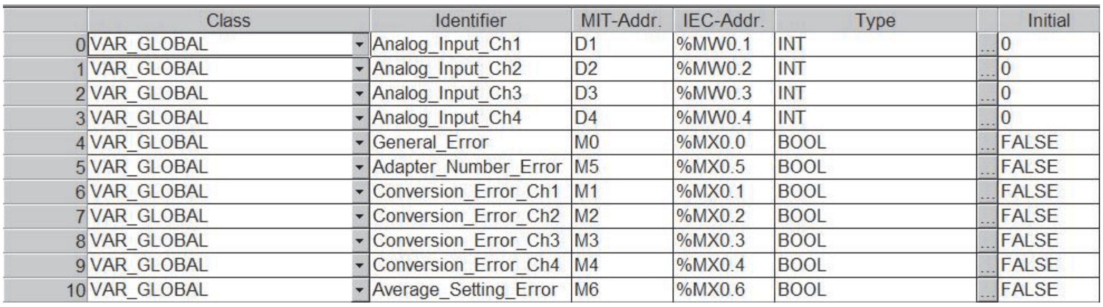
```
1                              Instance
                              FX3U_4AD_ADP
        ?── AdapterNumber                      AnalogInputCh1 ──?
        ?── InputModeCh1                       AnalogInputCh2 ──?
        ?── InputModeCh2                       AnalogInputCh3 ──?
        ?── InputModeCh3                       AnalogInputCh4 ──?
        ?── InputModeCh4                         GeneralError ──?
        ?── AverageSettingCh1              AdapterNumberError ──?
        ?── AverageSettingCh2              ConversionErrorCh1 ──?
        ?── AverageSettingCh3              ConversionErrorCh2 ──?
        ?── AverageSettingCh4              ConversionErrorCh3 ──?
        ?── ErrorReset                     ConversionErrorCh4 ──?
                                          AverageSettingError ──?
```

⑦ Define all variables as below:



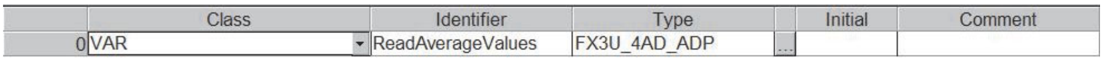
```
1                          ReadAverageValues
                              FX3U_4AD_ADP
        1── AdapterNumber                      AnalogInputCh1 ──Analog_Input_Ch1
     TRUE── InputModeCh1                       AnalogInputCh2 ──Analog_Input_Ch2
     TRUE── InputModeCh2                       AnalogInputCh3 ──Analog_Input_Ch3
    FALSE── InputModeCh3                       AnalogInputCh4 ──Analog_Input_Ch4
    FALSE── InputModeCh4                         GeneralError ──General_Error
        4── AverageSettingCh1              AdapterNumberError ──Adapter_Number_Error
        4── AverageSettingCh2              ConversionErrorCh1 ──Conversion_Error_Ch1
       10── AverageSettingCh3              ConversionErrorCh2 ──Conversion_Error_Ch2
       10── AverageSettingCh4              ConversionErrorCh3 ──Conversion_Error_Ch3
       X1── ErrorReset                     ConversionErrorCh4 ──Conversion_Error_Ch4
                                          AverageSettingError ──Average_Setting_Error
```

⑧ The outputs were registered as Global Variables.



	Class	Identifier	MIT-Addr.	IEC-Addr.	Type		Initial
0	VAR_GLOBAL	Analog_Input_Ch1	D1	%MW0.1	INT	...	0
1	VAR_GLOBAL	Analog_Input_Ch2	D2	%MW0.2	INT	...	0
2	VAR_GLOBAL	Analog_Input_Ch3	D3	%MW0.3	INT	...	0
3	VAR_GLOBAL	Analog_Input_Ch4	D4	%MW0.4	INT	...	0
4	VAR_GLOBAL	General_Error	M0	%MX0.0	BOOL	...	FALSE
5	VAR_GLOBAL	Adapter_Number_Error	M5	%MX0.5	BOOL	...	FALSE
6	VAR_GLOBAL	Conversion_Error_Ch1	M1	%MX0.1	BOOL	...	FALSE
7	VAR_GLOBAL	Conversion_Error_Ch2	M2	%MX0.2	BOOL	...	FALSE
8	VAR_GLOBAL	Conversion_Error_Ch3	M3	%MX0.3	BOOL	...	FALSE
9	VAR_GLOBAL	Conversion_Error_Ch4	M4	%MX0.4	BOOL	...	FALSE
10	VAR_GLOBAL	Average_Setting_Error	M6	%MX0.6	BOOL	...	FALSE

⑨ The instance name "ReadAverageValues" was assigned and defined as local variable.



	Class	Identifier	Type		Initial	Comment
0	VAR	ReadAverageValues	FX3U_4AD_ADP	...		

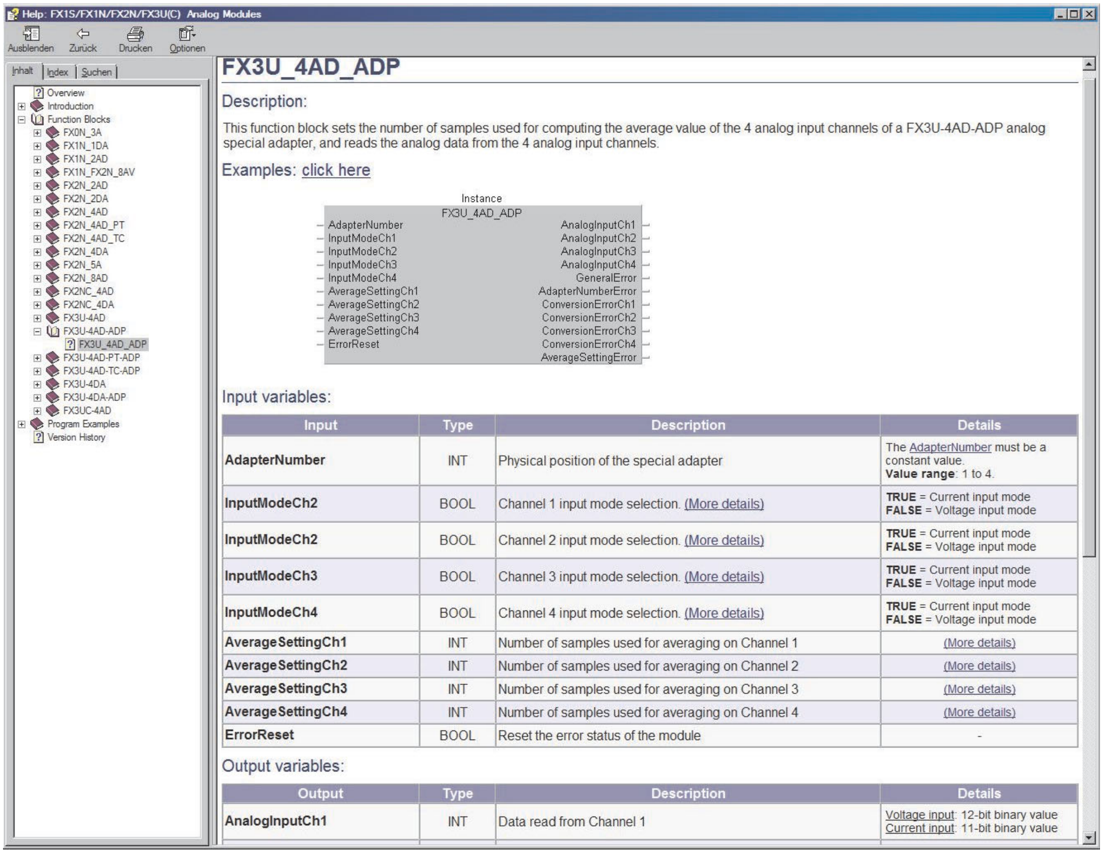
⑩ Compile and download the program to the PLC.

⑪ Monitor and test for correct operation. Observe the behaviour of the analogue outputs due to the “average settings”.

12.3.3 Library Function Block Help

Providing the accompanying Library Help file has been imported, for a full explanation with examples of all function blocks, click to highlight the function block and press the “F1” key.

For example, the help screen for the FX3U_4AD_ADP function block looks as follows:



FX3U_4AD_ADP

Description:

This function block sets the number of samples used for computing the average value of the 4 analog input channels of a FX3U-4AD-ADP analog special adapter, and reads the analog data from the 4 analog input channels.

Examples: click here

Input variables:

Input	Type	Description	Details
AdapterNumber	INT	Physical position of the special adapter	The AdapterNumber must be a constant value. **Value range**: 1 to 4.
InputModeCh2	BOOL	Channel 1 input mode selection. (More details)	**TRUE** = Current input mode **FALSE** = Voltage input mode
InputModeCh2	BOOL	Channel 2 input mode selection. (More details)	**TRUE** = Current input mode **FALSE** = Voltage input mode
InputModeCh3	BOOL	Channel 3 input mode selection. (More details)	**TRUE** = Current input mode **FALSE** = Voltage input mode
InputModeCh4	BOOL	Channel 4 input mode selection. (More details)	**TRUE** = Current input mode **FALSE** = Voltage input mode
AverageSettingCh1	INT	Number of samples used for averaging on Channel 1	(More details)
AverageSettingCh2	INT	Number of samples used for averaging on Channel 2	(More details)
AverageSettingCh3	INT	Number of samples used for averaging on Channel 3	(More details)
AverageSettingCh4	INT	Number of samples used for averaging on Channel 4	(More details)
ErrorReset	BOOL	Reset the error status of the module	-

Output variables:

Output	Type	Description	Details
AnalogInputCh1	INT	Data read from Channel 1	Voltage input: 12-bit binary value Current input: 11-bit binary value

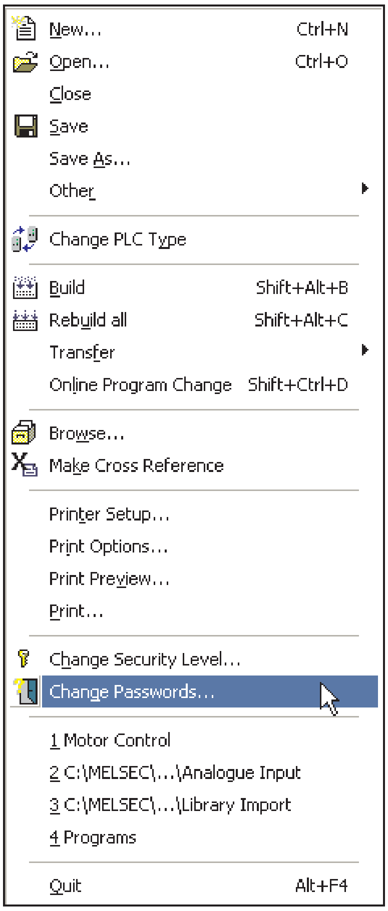
The help files cover every aspect from the setup of the FX family analogue hardware modules to use of the library function blocks.

13 Security

13.1 Password

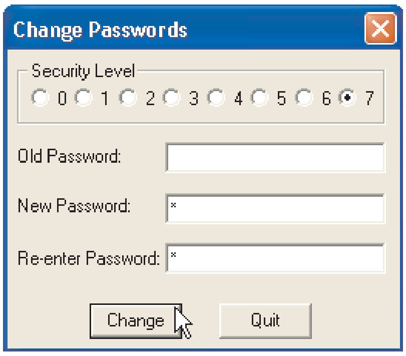
You can protect all or parts of the program with a password. You can protect against editing of program parts and also protect circuits from being viewed by others. This is particularly relevant for user defined function blocks. In addition, the PLC password (Keyword) is also available.

13.1.1 Setting the Password



Passwords can be entered and security levels can be changed, using these windows, via the ***Project*** menu.

To illustrate the operation of passwords, select ***Security Level*** 7 and enter a new password for this level (For simplicity here, press 7). Re-enter the password and click ***Change***.



Security Level
○ 0 ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ● 7

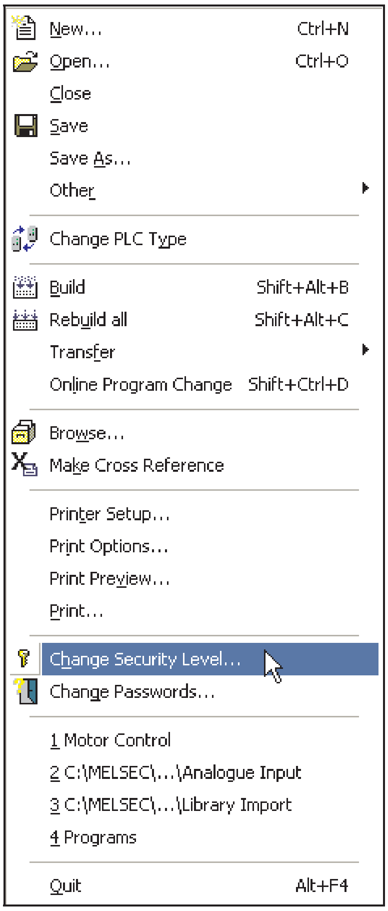
Old Password:

New Password: *

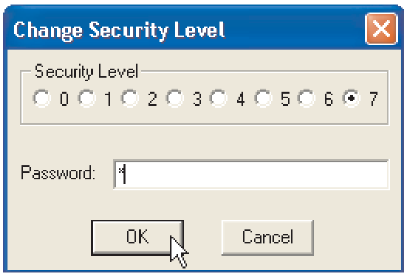
Re-enter Password: *

13.1.2 Changing the Security Level

① Select ***Change Security Level*** from the ***Project*** menu:



② Enter the password for 'Level 7' and if accepted, the user will be logged on at this level.



Security Level
○ 0 ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ● 7

Password: *

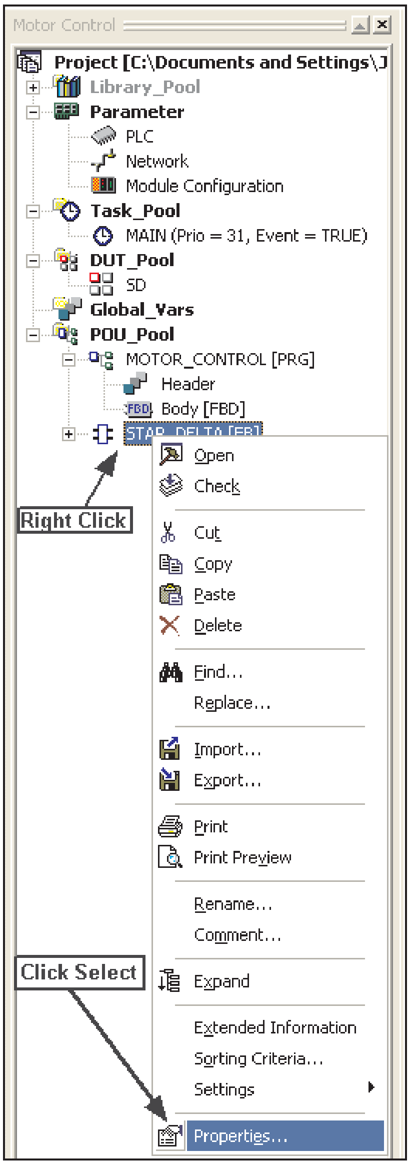
Once logged on, the security attributes for many items may be altered. For example one of the most common security options is to change access to POU's, i.e. User Functions and Function Blocks.

13.1.3 Modifying POU Password Access

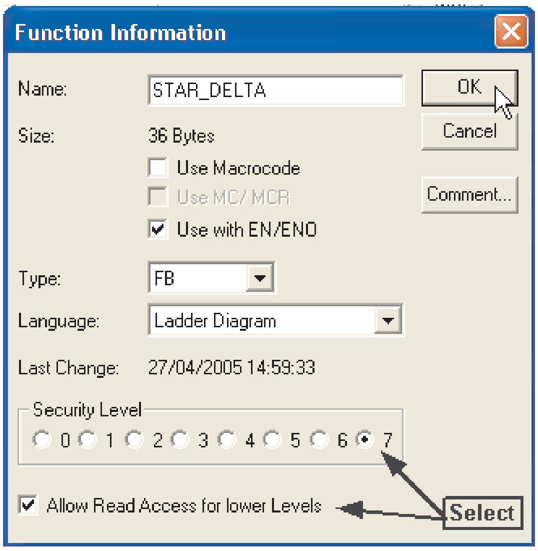
In order to protect the content or control access to User POU's the security attributes may be adjusted, whilst being logged into the security current level, as follows:

Setting Security Level

① Open the project "Motor Control" and open the header of the Function Block "STAR_DELTA":



② Adjust the Security to Level ‘7’ and click ***Allow Read Access for lower Levels***. This will allow subordinate users “Read access” only to the Header and body of the function Block:



Function Information

Name: STAR_DELTA

Size: 36 Bytes

□ Use Macrocode

□ Use MC/ MCR

☑ Use with EN/ENO

Type: FB

Language: Ladder Diagram

Last Change: 27/04/2005 14:59:33

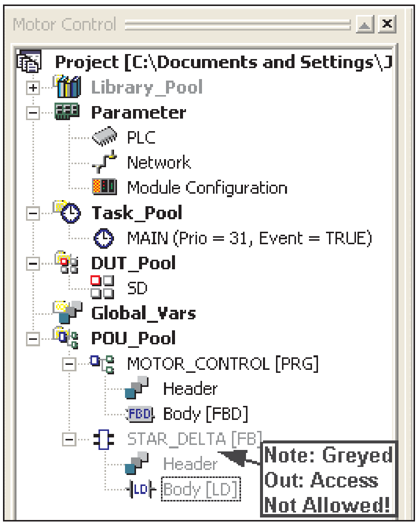
Security Level ○ 0 ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ● 7

☑ Allow Read Access for lower Levels

③ Change the security level to Level ‘0’ and access the header and body of the Function Block “STAR_DELTA”. Read access will be allowed for monitoring purposes but any alteration to the code is **not** possible.

④ Log in again to Level 7 and alter the security attributes of the Function Block “STAR_DELTA” so that read access is **NOT** allowed for lower levels.

⑤ Change the security level to ‘0’ and try to access the body of the Function Block “STAR_DELTA”. The Header and Body of the POU will be greyed out with access to the POU completely blocked:



Motor Control

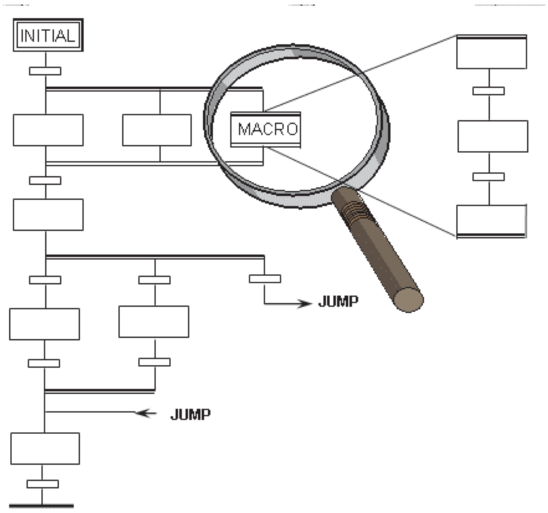
```
Project [C:\Documents and Settings\J
  Library_Pool
  Parameter
    PLC
    Network
    Module Configuration
  Task_Pool
    MAIN (Prio = 31, Event = TRUE)
  DUT_Pool
    SD
  Global_Vars
  POU_Pool
    MOTOR_CONTROL [PRG]
      Header
      Body [FBD]
    STAR_DELTA [FB]
      Header
      Body [LD]
```

Note: Greyed Out: Access Not Allowed!

Access attributes for <u>any</u> individual object or complete folder in the ‘Project Navigation Window’ above can be individually set, allowing higher degrees of flexibility in the program security settings.

14 Sequential Function Chart - SFC

14.1 What is SFC?

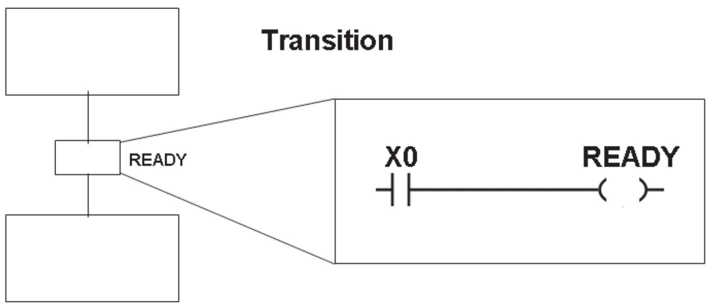


- The “Sequential Function Chart” editor is a guided editor.
- Graphical Flowchart representation.
- Based on the French Grafcet (IEC 848)
- SFC is a structural language which divides the process into steps and transitions.
- The steps “hide” actions (no POUs) and / or directly switched bit operands.
- Transitions always contain one link/network which activates the progression instruction (name of the transition).

  (It is also possible to use a discrete address instead of a name.)
- Actions can be created in every editor, except SFC.
- Transitions can be created in every editor, except SFC.
- The SFC code resides in the Micro-computer area of the plc, so allocate memory space in PLC Parameters (A series only).

14.2 SFC Elements

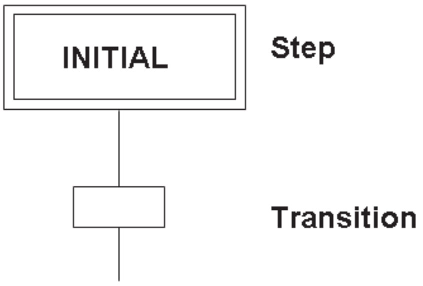
14.2.1 SFC Transitions



- Transitions represent a link which starts progression.
- They can be created in every IEC editor.
- Except in SFC.
- It is also possible to use a bit directly instead of the name READY.

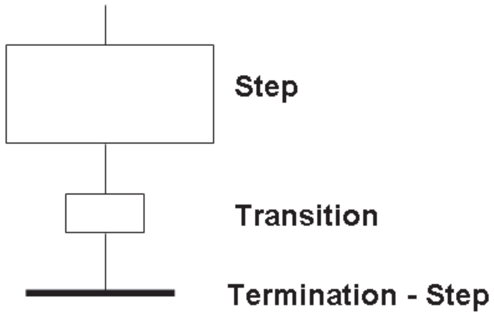
14.2.2 Initial Step

SFC programs begin with an Initial Step function which indicates the start of a sequence:

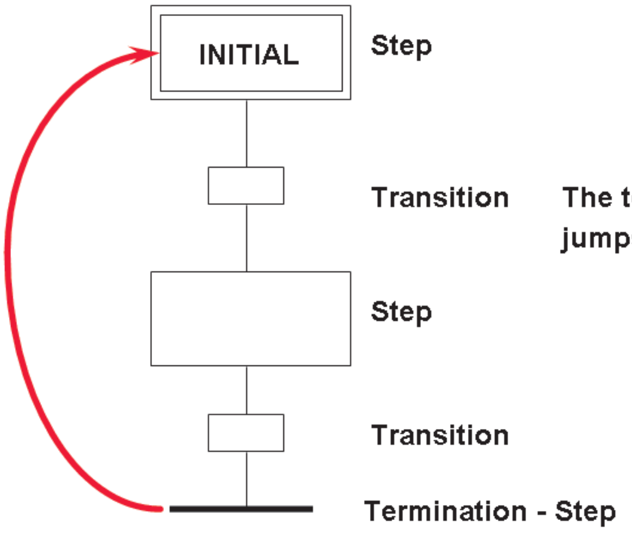


14.2.3 Termination Step

All Sequences finish with a Termination Step:



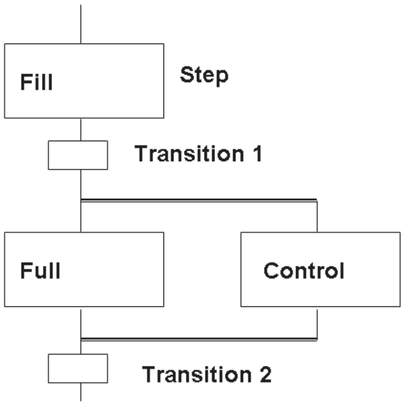
Initialisation / Termination - Step



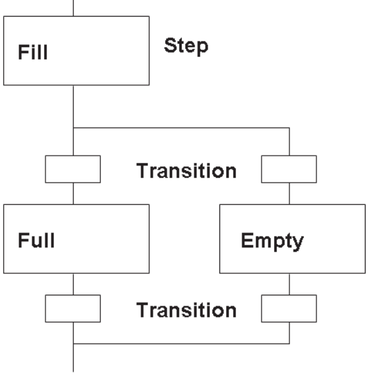
The termination-step automatically jumps to the initialisation-step.

14.3 SFC configuration examples

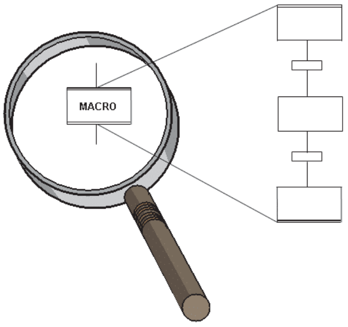
Parallel Branch



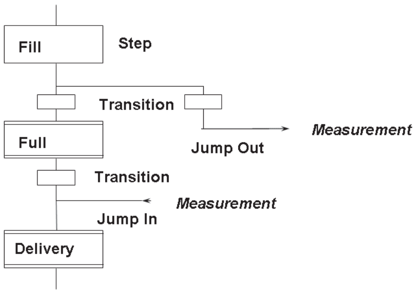
Selective Branch



Macro - Step

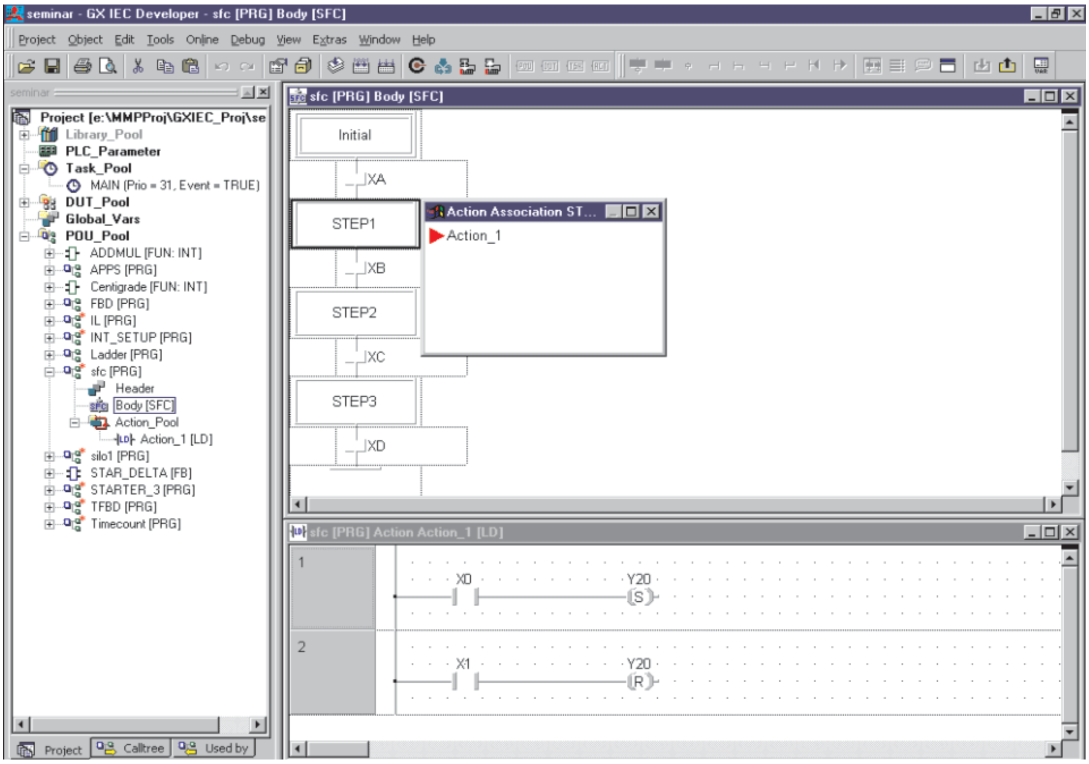


SFC Jump

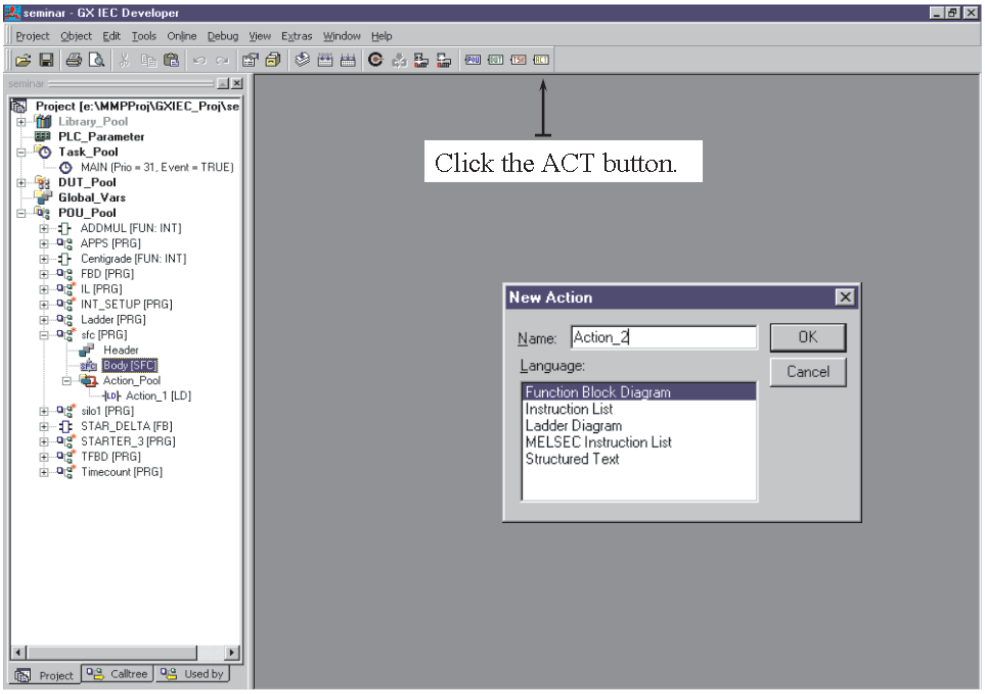


14.4 SFC Actions

Each step has associated actions. An action is simply a program, as for a POU. Each action has associated logic written in either, IEC LD, IL, FBD or ST:



New Actions are created by clicking on the ***ACT*** button on the toolbar. Select the required editor, as for POUs:

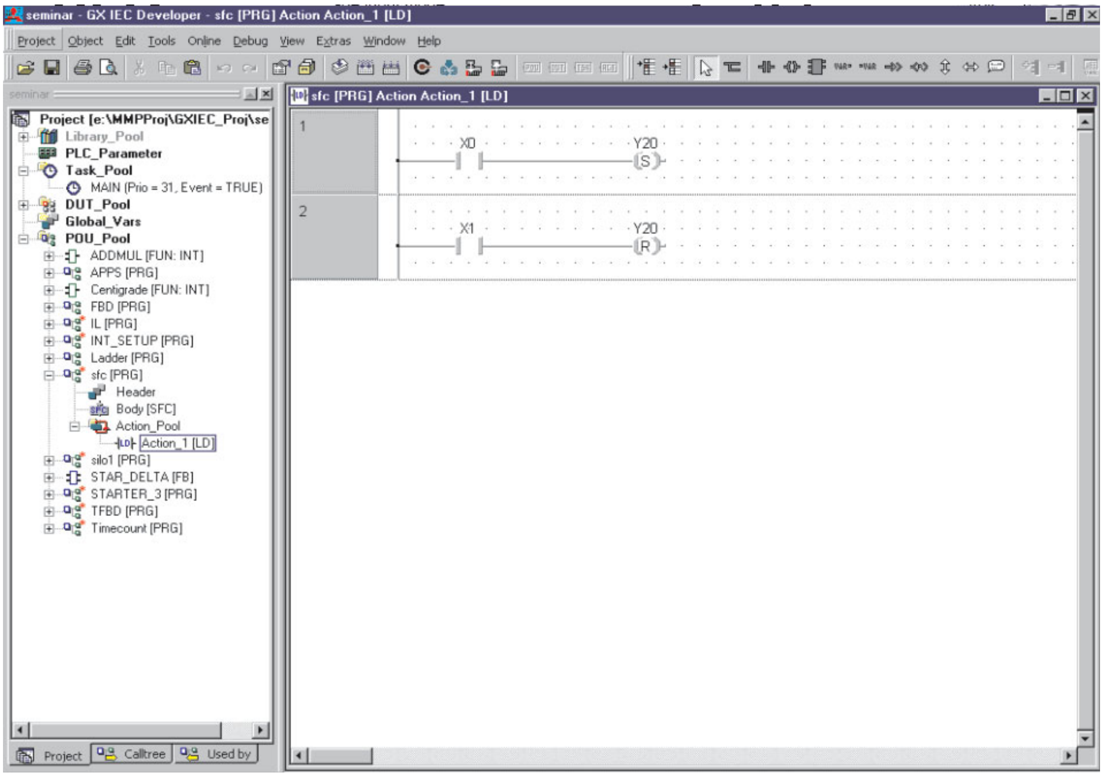


Click the ACT button.

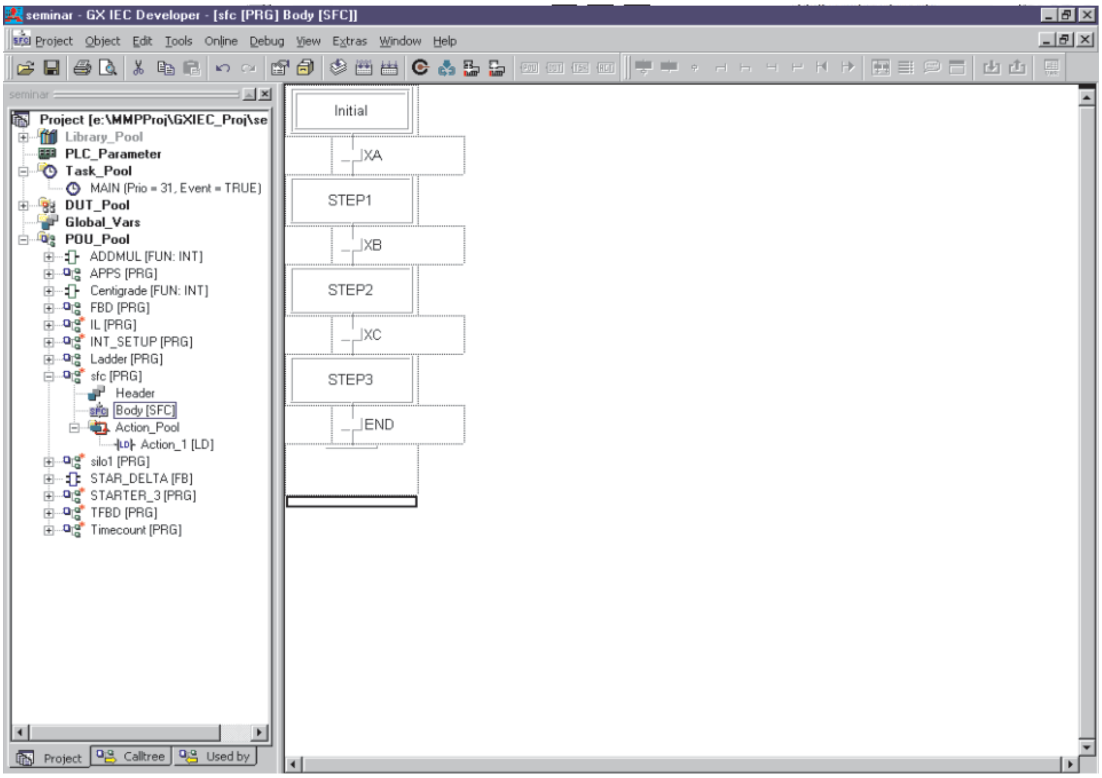
Name: Action_2

Language: Function Block Diagram

Actions can be programs within their own right. Action_1 may be a complete ladder interlocking routine, consisting of many networks

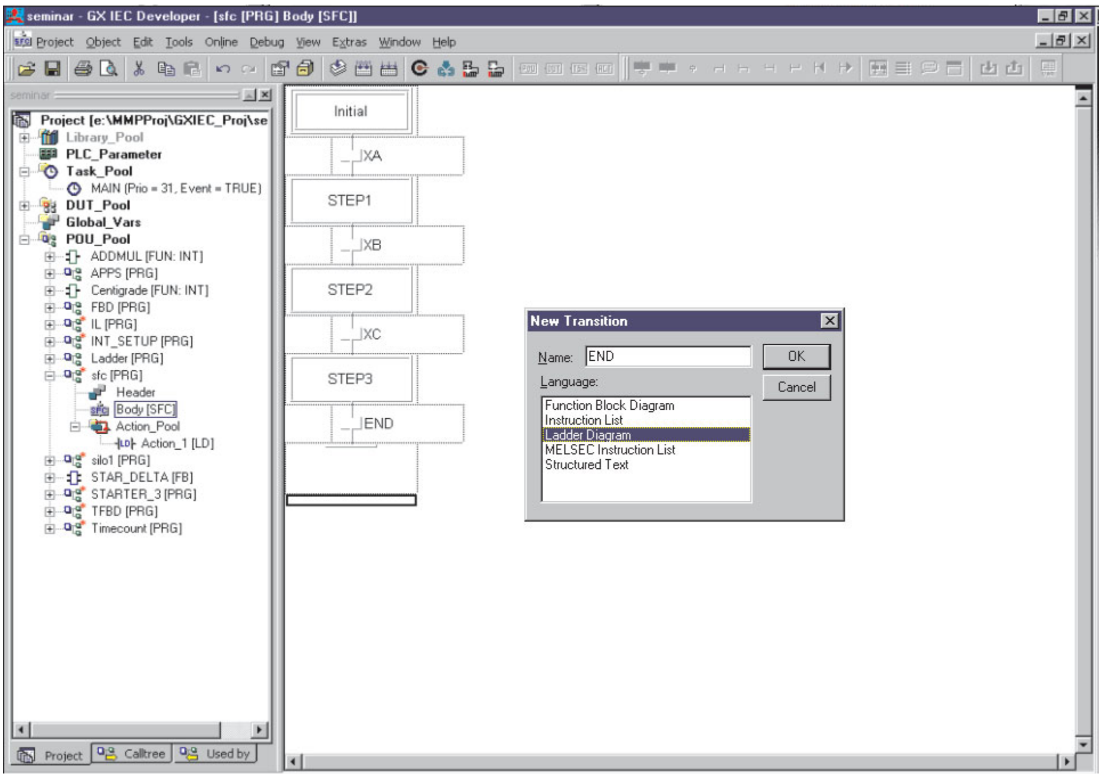


Each Transition can be a simple device i.e. Mitsubishi address X0, or an identifier name, or more complex, as a single network program written in either IEC, IL, LD, Structured Text or FBD:



14.5 Complex Transitions

To program a complex transition, input a Transition name and hit the enter key. Choose the required editor, as for Actions:



seminar - GX IEC Developer - [sfc [PRG] Body [SFC]]

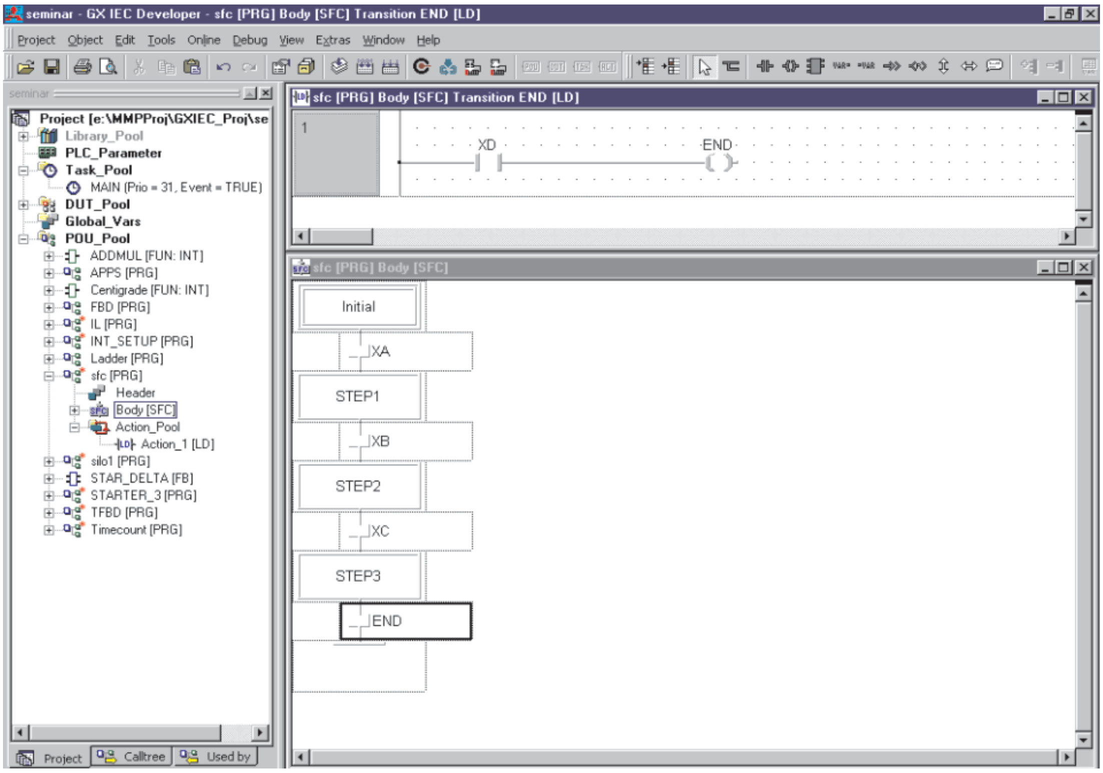
New Transition

Name: END

Language:

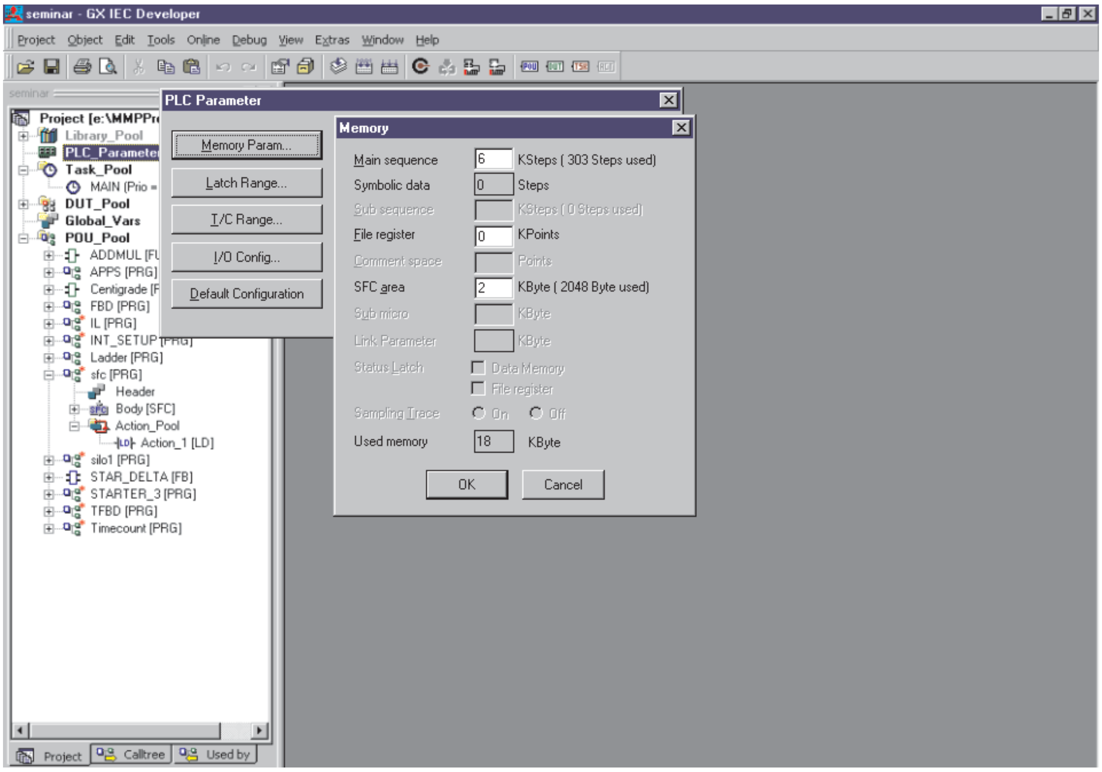
Function Block Diagram
Instruction List
Ladder Diagram
MELSEC Instruction List
Structured Text

The transition could be a complex expression but it only consists of one network:



seminar - GX IEC Developer - sfc [PRG] Body [SFC] Transition END [LD]

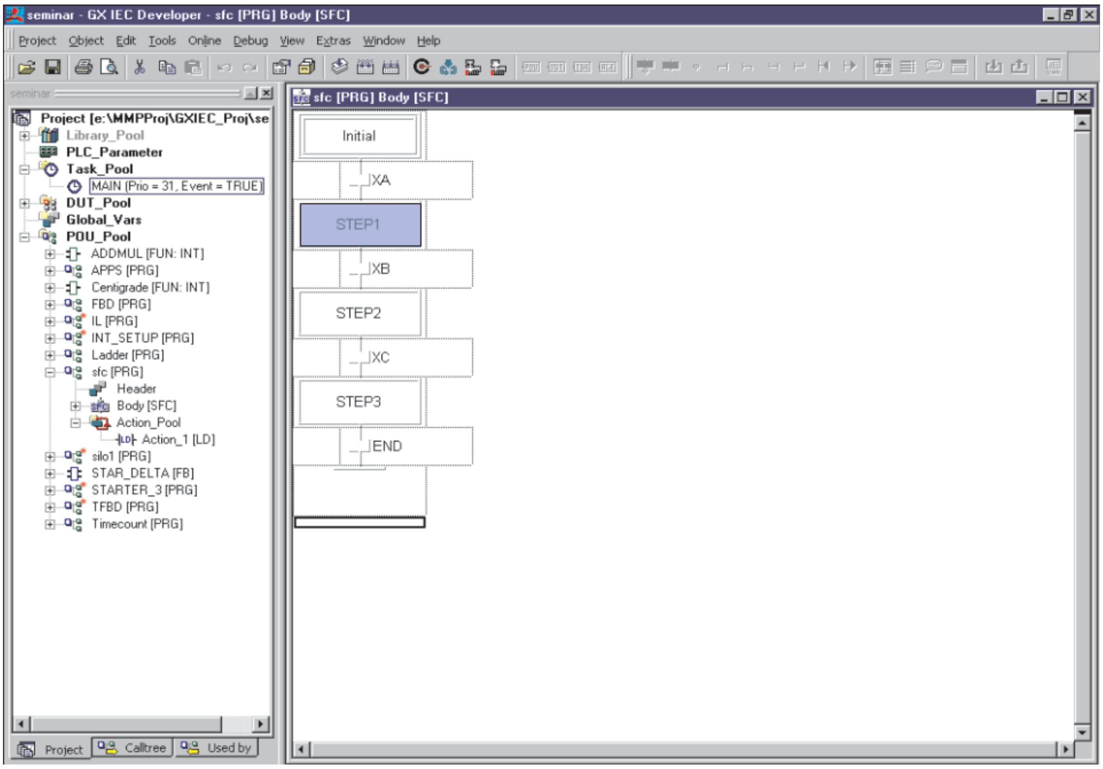
For A(ns) Series PLC's, SFC's reside in the micro computer area of the memory cassette. This area must be allocated from PLC Parameters / Memory, as shown below:



Main sequence	6	KSteps (303 Steps used)
Symbolic data	0	Steps
Sub sequence		KSteps (0 Steps used)
File register	0	KPoints
Comment space		Points
SFC area	2	KByte (2048 Byte used)
Sub micro		KByte
Link Parameter		KByte
Status Latch	□ Data Memory □ File register	
Sampling Trace	○ On ○ Off	
Used memory	18	KByte

This is not the case for Q series, as the MELSEC System Q supports SFC's in the program area. Also for FX range, SFC's actually compile to STL code in the program area.

One popular feature of SFC's, is that in monitor mode, the current step is highlighted. This means for fault finding purposes, engineers can see exactly how far the sequence has progressed and can investigate accordingly:



15 IEC Instruction List

- The “Instruction List” editor is a free text editor.
- No line addresses are released.
- Functions and function blocks can be called.
- In addition to the IEC networks MELSEC networks can be included.
- Comments can be included within (* *)
- By means of the Windows functionality a program can be written for example in WinWord and then be copied via the clip board into GX IEC Developer.

15.1 Example of IEC Instruction List (IL)

```
LD      X4        (* Interrogation X4 *)
ANDN    M5        (* ANDN M5 *)
ST      Y20       (* Assignment OUT to Y20 *)

LD      TEST      (* Load TEST into accu *)
BCD_TO_INT        (* Convert accu *)
ST      RESULT    (* Write accu to RESULT *)
```

15.1.1 Some useful tips

To Perform : “ + D0 D1 D2 ” in IEC IL, becomes:

```
LD      D0
ADD     D1
ST      D2
```

To Perform : “ + D0 D1 D2 ” and then “ + D2 K50 D3 ” becomes:

```
LD      D0
ADD     D1,D2,50
ST      D3
```

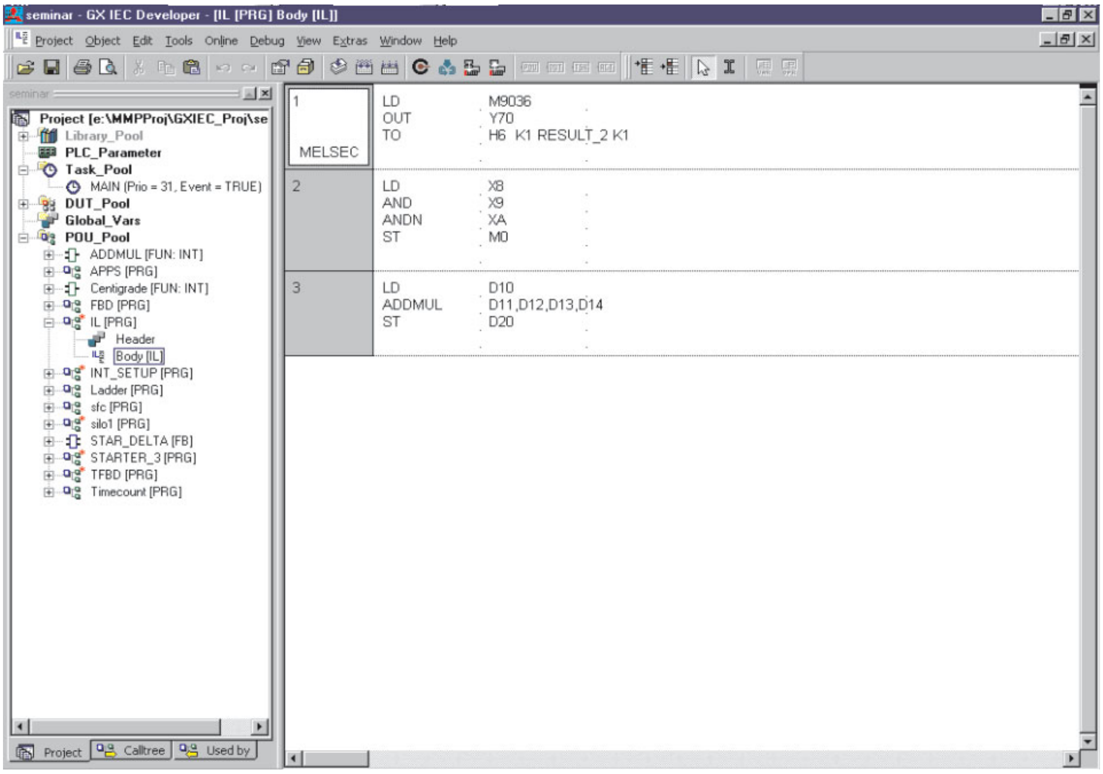
Use of an “_E” function can simplify still further. To Perform : “ + D0 D1 D2 ” and then “ + D2 K50 D3 ” from a conditional input X0 becomes:

```
LD      X0
ADD_E   D0,D1,D2,50,D3
```

This is because the ADD_E function has an Enable Output (ENO) feature.

15.2 Mixing IEC IL and Melsec IL in POUs

Both IEC IL and Melsec IL networks can be incorporated into the same POU. This is achieved, by highlighting the current network, selecting from the Edit Menu, ***New Network*** then ***Melsec Before*** from the ***Options*** list:



```
1        LD       M9036
         OUT      Y70
MELSEC   TO       H6  K1 RESULT_2 K1

2        LD       X8
         AND      X9
         ANDN     XA
         ST       M0

3        LD       D10
         ADDMUL   D11,D12,D13,D14
         ST       D20
```

16 IEC Structured Text

ST is a high level textual editor, which has the appearance of PASCAL but is a dedicated language for industrial control applications.

POUs, Functions and Function Blocks can be created using ST.

IEC Structured Text example:

IF …..THEN ….. ELSE conditions
CASE ...ELSE .... END_CASE structures
REPEAT
RETURN
Expression Evaluation
Variable Declaration etc

Complex mathematical expressions can be realised using these operators, in a few lines of text.

16.1 Structured Text Operators

Operator	Description	Precedence
(….)	Parenthesised expression	Highest
Function(….)	Parameter list of a function, function evaluation	
**	Exponentiation, ie raising to a power	
-	Negation	
NOT	Boolean compliment	
*	Multiplication	
/	Division	
MOD	Modulus operation	
+	Addition	
-	Subtraction	
<,>,<=,>=	Comparison operators	
=	Equality	
<>	Non equality	
AND, &	Boolean AND	
XOR	Boolean exclusive OR	
OR	Boolean OR	Lowest

16.2 Structured Text Program Example

A new Function Block will be constructed to perform a simple “Centigrade to Fahrenheit” conversion similar to that used in a previous example, in order to illustrate the use of the ‘Structured Text’ language editor.

The formula used is as follows:

$$Fahrenheit = \frac{Celsius \times 9}{5} + 32$$

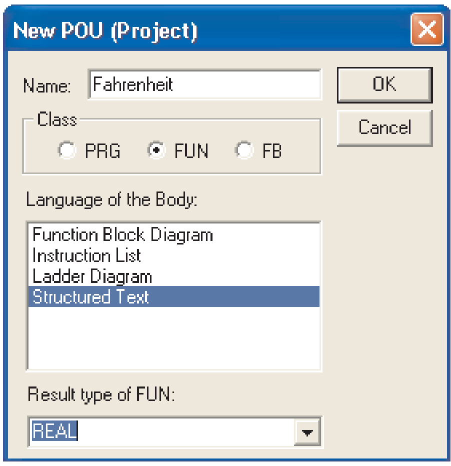
The input and result variables will be in Floating Point (REAL) format.

NOTE

For the FX range of PLCs, floating point calculation is only possible with the main units of the FX2N, FX2NC, FX3G, FX3U, and FX3UC series.

① Create a new project called "Structured_Text".

② Create a new POU named "Fahrenheit", of Class: ***FUN***, Result Type: ***REAL***, with a language of “ST” (***Structured Text***):



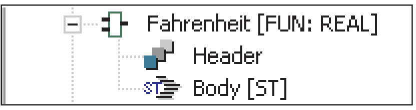
New POU (Project)

Name: Fahrenheit

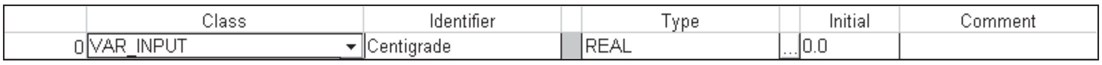
Class: ○ PRG ● FUN ○ FB

Language of the Body:
- Function Block Diagram
- Instruction List
- Ladder Diagram
- Structured Text

Result type of FUN: REAL



③ Create an entry in the header (LVL) of the Function “Fahrenheit”:

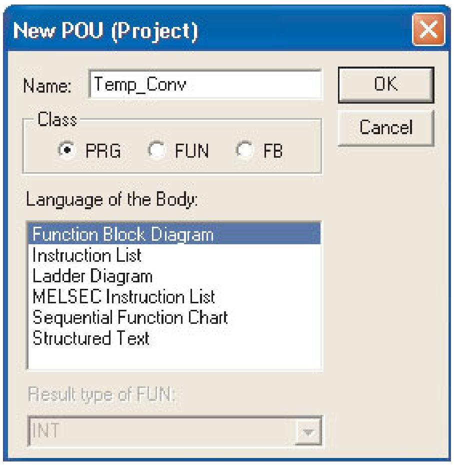


	Class	Identifier	Type	Initial	Comment
0	VAR_INPUT	Centigrade	REAL	0.0	

④ Open the Body of the Function “Fahrenheit” and enter the following simple ST program:

Fahrenheit := (Centigrade*9.0/5.0+32.0);

⑤ Create a new POU with a name “Temp_Conv”, Class: ***PRG***, Language: ***Function Block Diagram***



New POU (Project)

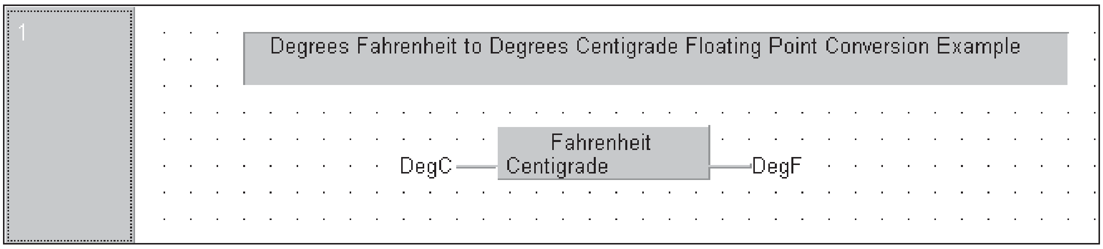
Name: Temp_Conv

Class: ● PRG ○ FUN ○ FB

Language of the Body:
- Function Block Diagram
- Instruction List
- Ladder Diagram
- MELSEC Instruction List
- Sequential Function Chart
- Structured Text

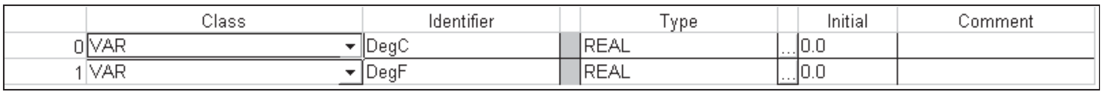
Result type of FUN: INT

⑥ Open the body of the program POU “Temp_Conv” and enter the following program example:



Degrees Fahrenheit to Degrees Centigrade Floating Point Conversion Example

⑦ Edit the LVL (Header) of the POU “Temp_Conv” to include 2 local variables as shown below:

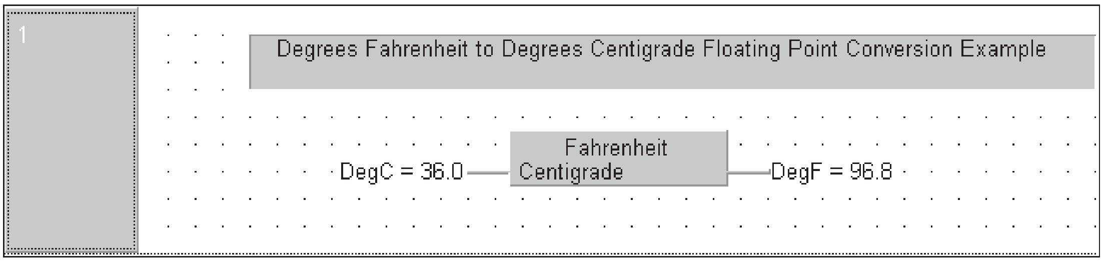


	Class	Identifier	Type	Initial	Comment
0	VAR	DegC	REAL	0.0	
1	VAR	DegF	REAL	0.0	

⑧ Close all open editors, compile the project using “Rebuild All”. Save and download to the PLC.

⑨ Monitor the program body of “Temp_Conv” and observe the values on screen.

⑩ Force new values into the input variable “DegC” of the equation by double clicking on the variable Tag Name.



Degrees Fahrenheit to Degrees Centigrade Floating Point Conversion Example

DegC = 36.0

DegF = 96.8

NOTE

In this example, Local Variables are used to directly enter values via the GX IEC Developer programming / monitoring interface; normally values are entered via Global Variables.

17 PROFIBUS/DP Communication

The open PROFIBUS/DP network enables extremely fast data exchange with a very wide variety of slave devices, including remote digital I/Os, remote analog I/Os, frequency inverters and a range of other devices from third-party manufacturers. Of course, PROFIBUS/DP slaves from MITSUBISHI ELECTRIC can also be connected to master devices from other manufacturers.

The installation of remote digital or analog I/Os helps to reduce costs for wiring.

Structure

The maximum coverage of a bus segment is 1200 m (at a maximum of 93.75 kbit/s). Up to 3 repeaters are allowed. Thus the maximum distance between 2 stations is calculated with 4800 m.

Cable types

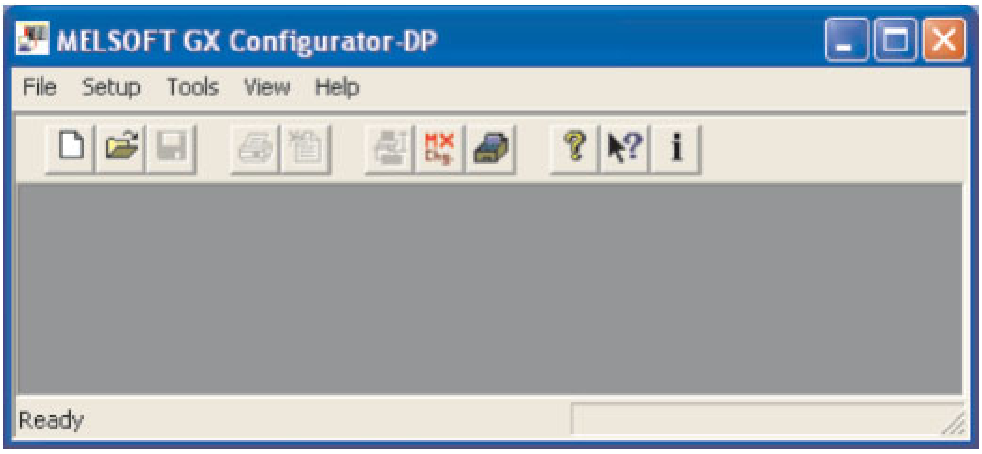
To help reduce costs PROFIBUS/DP uses RS 485 technology with shielded 2-wire cabling.

17.1 Configuring the PROFIBUS/DP Network

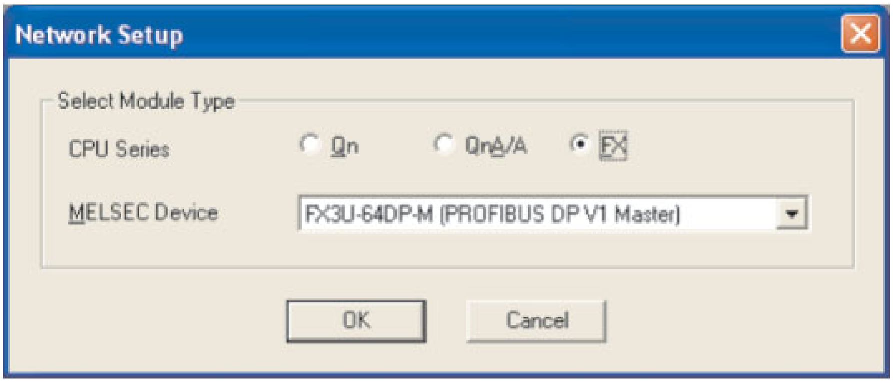
In combination with the software GX Configurator DP the FX3U-64DP-M master unit as well as master modules from the A series or the MELSEC System Q give you user-friendly plug-and-play technology. The configuration software is self-explanatory, using a graphical model for setting up the network. You simply select the slave unit, assign the station numbers and specify where the information is stored in the master station.

In this chapter the configuration of a PROFIBUS/ DP master module FX3U-64DP-M installed in a FX3U base unit is shown. Connected to the master module is a slave station consisting of digital and analog modules of the MELSEC ST series. For more information of the ST series please refer to the Technical Catalogue Networks, art.-no. 136730.

① Start GX Configurator DP and open a new project.



② In the dialog ***Network Setup*** *select* ***FX***. As ***MELSEC Device*** FX3U-64DP-M is automatically entered.



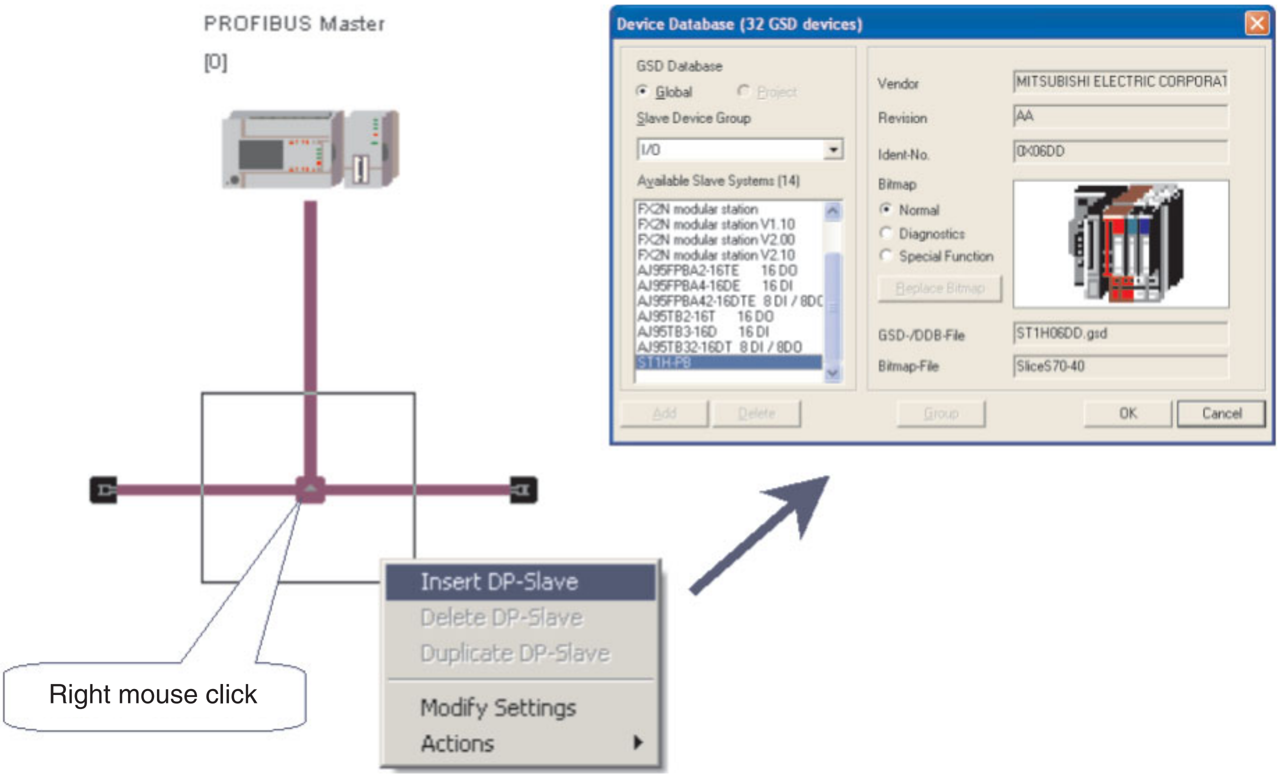
Network Setup

Select Module Type

CPU Series ○ Qn ○ QnA/A ● FX

MELSEC Device FX3U-64DP-M (PROFIBUS DP V1 Master)

③ Insert DP-Slave in empty project.



Device Database (32 GSD devices)

GSD Database ● Global ○ Project

Slave Device Group: I/0

Available Slave Systems (14)

FX2N modular station
FX2N modular station V1.10
FX2N modular station V2.00
FX2N modular station V2.10
AJ95FPBA2-16TE 16 DO
AJ95FPBA4-16DE 16 DI
AJ95FPBA42-16DTE 8 DI / 8DO
AJ95TB2-16T 16 DO
AJ95TB3-16D 16 DI
AJ95TB32-16DT 8 DI / 8DO
ST1H-PB

Vendor: MITSUBISHI ELECTRIC CORPORAT

Revision: AA

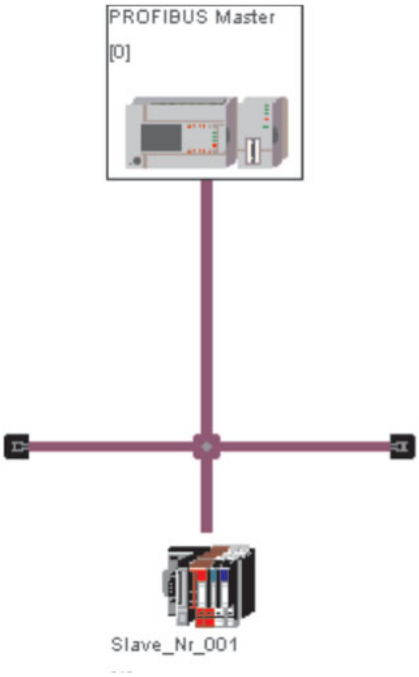
Ident-No.: 0X06DD

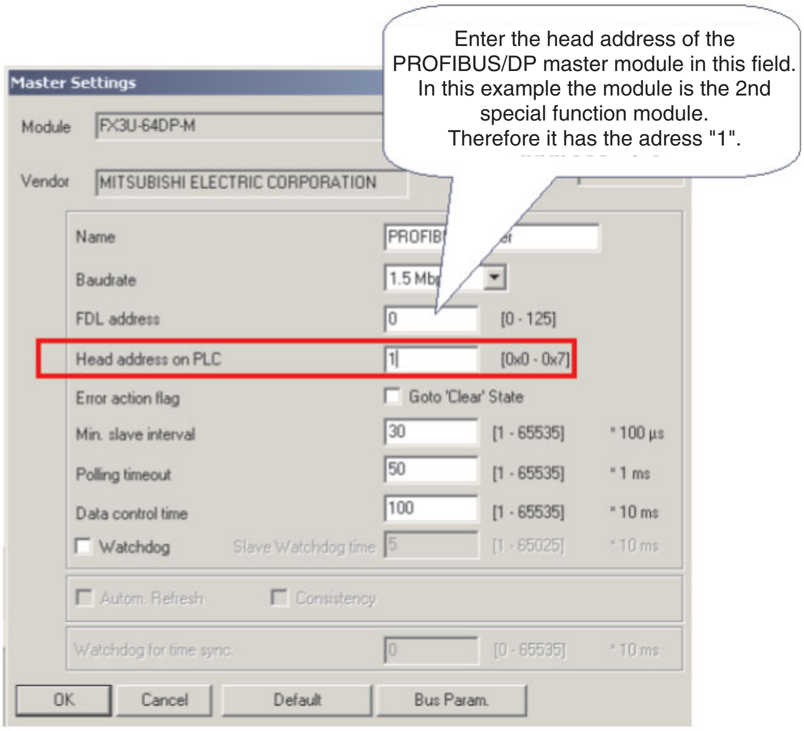
Bitmap: ● Normal ○ Diagnostics ○ Special Function

GSD-/DDB-File: ST1H06DD.gsd

Bitmap-File: SliceS70-40

④ Define the head address of the master module.




Master Settings

Module: FX3U-64DP-M

Vendor: MITSUBISHI ELECTRIC CORPORATION

Name: PROFIB…er

Baudrate: 1.5 Mbp…

FDL address: 0 [0 - 125]

Head address on PLC: 1 [0x0 - 0x7]

□ Error action flag: Goto 'Clear' State

Min. slave interval: 30 [1 - 65535] * 100 µs

Polling timeout: 50 [1 - 65535] * 1 ms

Data control time: 100 [1 - 65535] * 10 ms

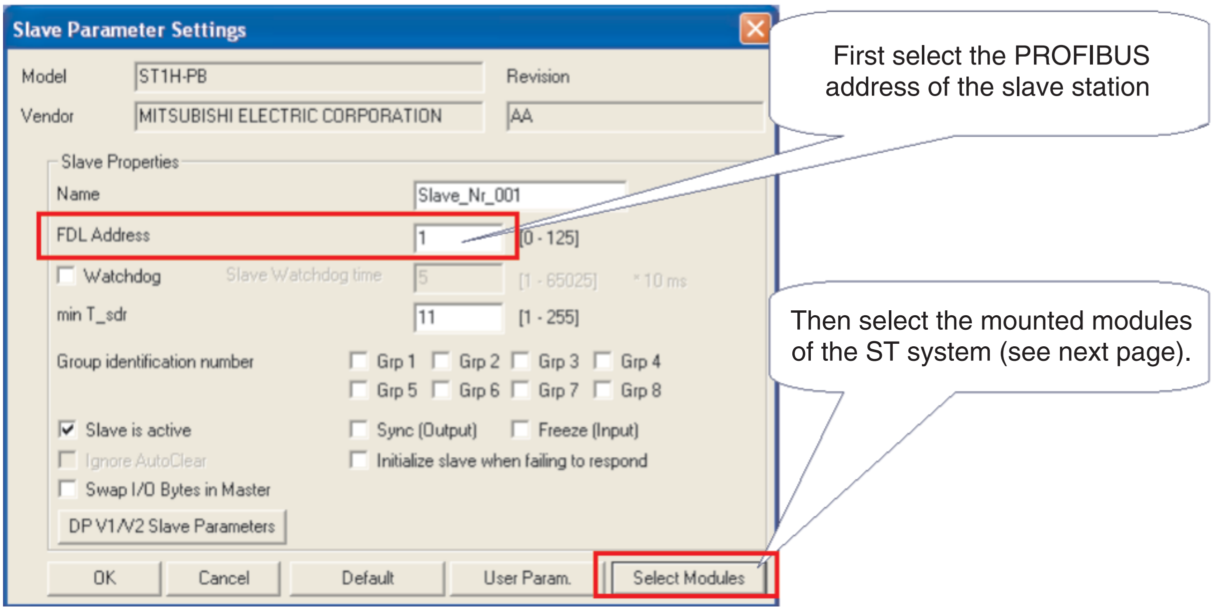
□ Watchdog — Slave Watchdog time: 5 [1 - 65025] * 10 ms

□ Autom. Refresh □ Consistency

Watchdog for time sync.: 0 [0 - 65535] * 10 ms

Enter the head address of the PROFIBUS/DP master module in this field. In this example the module is the 2nd special function module. Therefore it has the adress "1".

⑤ Configure the slave station. In this example it is a head module of the MELSEC ST series (ST1H-PB).


Slave Parameter Settings

Model: ST1H-PB — Revision: AA

Vendor: MITSUBISHI ELECTRIC CORPORATION

Slave Properties

Name: Slave_Nr_001

FDL Address: 1 [0 - 125]

□ Watchdog — Slave Watchdog time: 5 [1 - 65025] * 10 ms

min T_sdr: 11 [1 - 255]

Group identification number: □ Grp 1 □ Grp 2 □ Grp 3 □ Grp 4 □ Grp 5 □ Grp 6 □ Grp 7 □ Grp 8

☑ Slave is active □ Sync (Output) □ Freeze (Input)

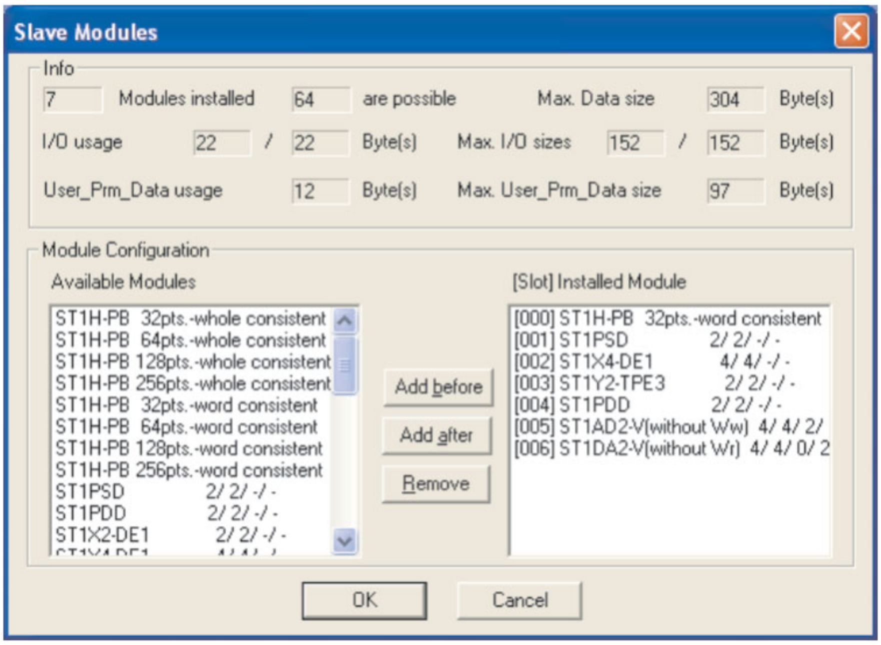
□ Ignore AutoClear □ Initialize slave when failing to respond

□ Swap I/O Bytes in Master

First select the PROFIBUS address of the slave station

Then select the mounted modules of the ST system (see next page).

⑥ Select modules



Slave Modules

Info

7	Modules installed	64	are possible	Max. Data size	304	Byte(s)
I/O usage	22 /	22	Byte(s)	Max. I/O sizes	152 / 152	Byte(s)
User_Prm_Data usage		12	Byte(s)	Max. User_Prm_Data size	97	Byte(s)

Module Configuration

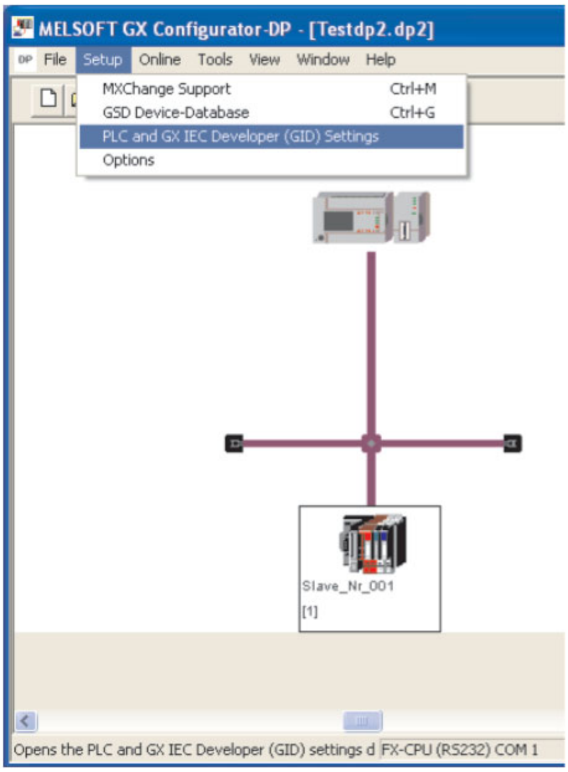
Available Modules

```
ST1H-PB  32pts.-whole consistent
ST1H-PB  64pts.-whole consistent
ST1H-PB 128pts.-whole consistent
ST1H-PB 256pts.-whole consistent
ST1H-PB  32pts.-word consistent
ST1H-PB  64pts.-word consistent
ST1H-PB 128pts.-word consistent
ST1H-PB 256pts.-word consistent
ST1PSD          2/ 2/ -/ -
ST1PDD          2/ 2/ -/ -
ST1X2-DE1        2/ 2/ -/ -
```

[Slot] Installed Module

```
[000] ST1H-PB  32pts.-word consistent
[001] ST1PSD            2/ 2/ -/ -
[002] ST1X4-DE1          4/ 4/ -/ -
[003] ST1Y2-TPE3          2/ 2/ -/ -
[004] ST1PDD            2/ 2/ -/ -
[005] ST1AD2-V(without Ww)  4/ 4/ 2/
[006] ST1DA2-V(without Wr)  4/ 4/ 0/ 2
```

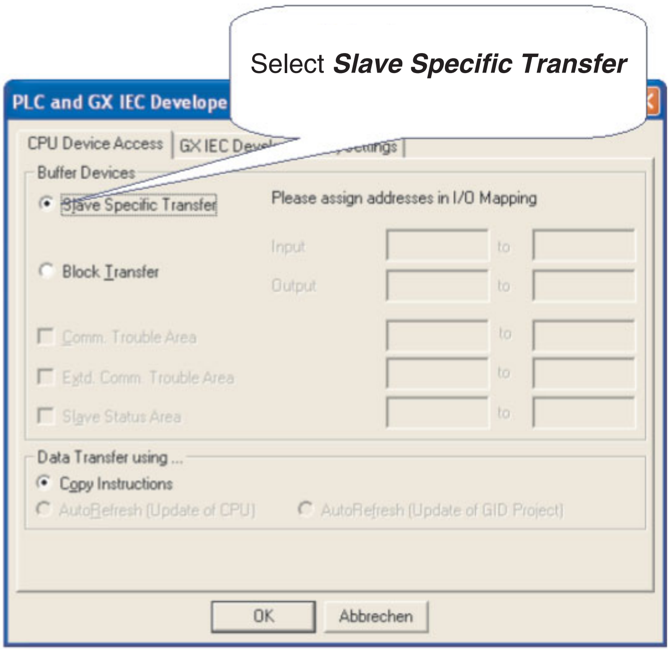
⑦ Make PLC settings for input and output devices.



MELSOFT GX Configurator-DP - [Testdp2.dp2]

File Setup Online Tools View Window Help
MXChange Support Ctrl+M
GSD Device-Database Ctrl+G
PLC and GX IEC Developer (GID) Settings
Options
Slave_Nr_001
[1]

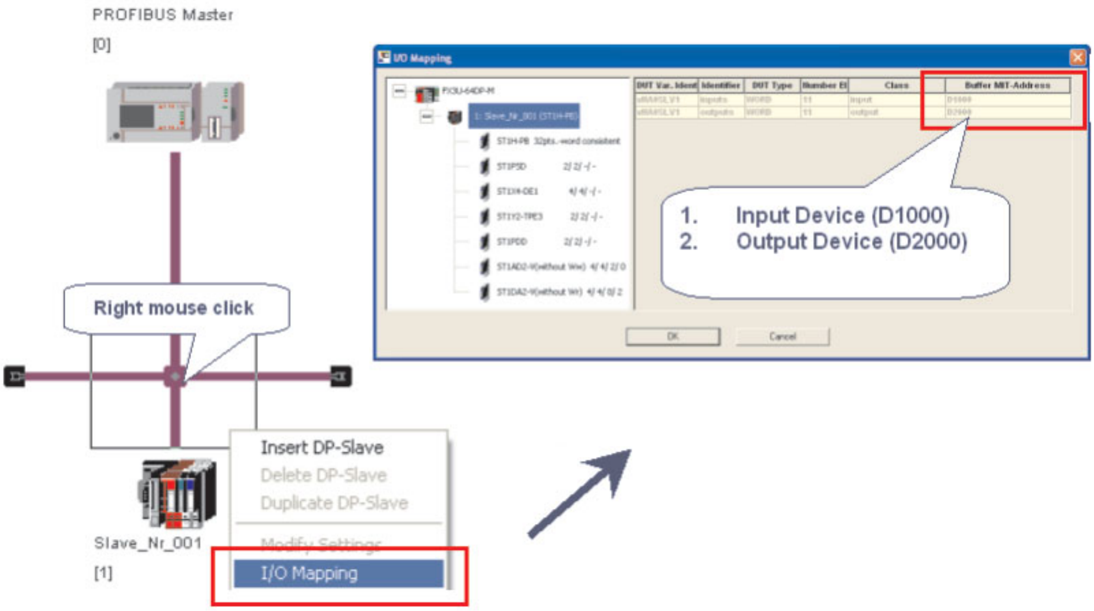
Opens the PLC and GX IEC Developer (GID) settings d FX-CPU (RS232) COM 1



Select ***Slave Specific Transfer***

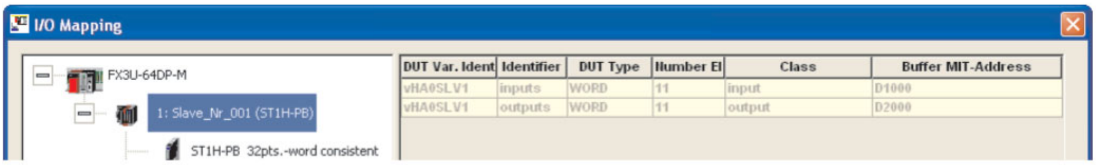
PLC and GX IEC Develope

⑧ Slave Specific Transfer

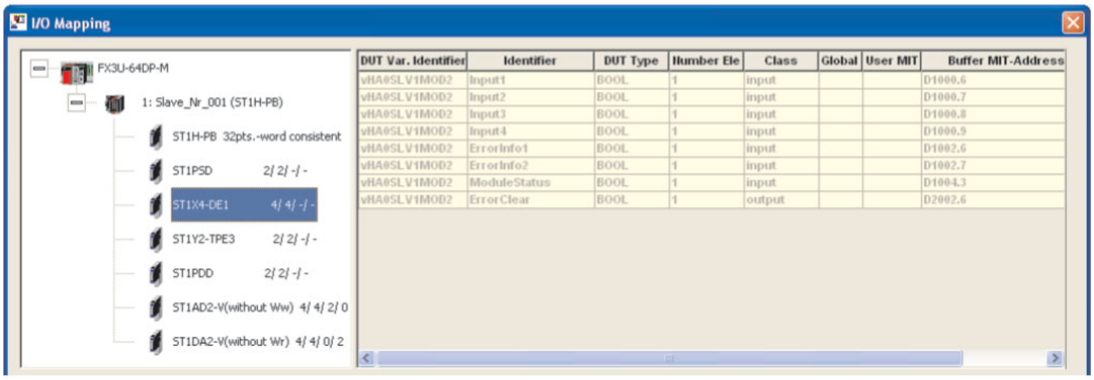


1. Input Device (D1000)
2. Output Device (D2000)

⑨ I/O mapping

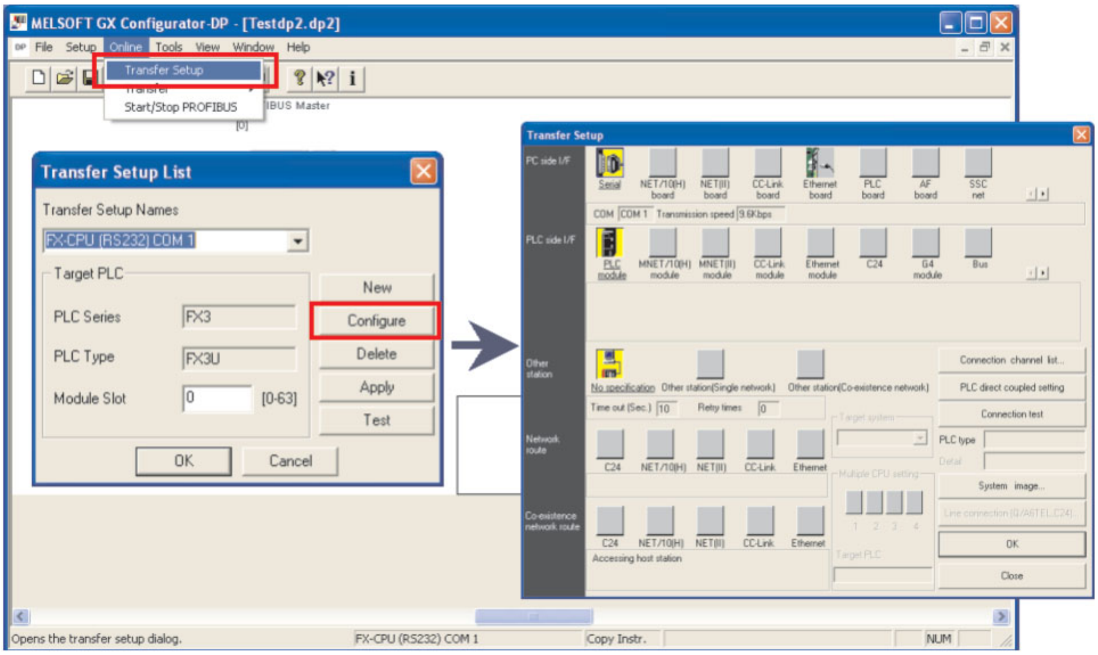


DUT Var. Ident	Identifier	DUT Type	Number El	Class	Buffer MIT-Address
vHA0SLV1	inputs	WORD	11	input	D1000
vHA0SLV1	outputs	WORD	11	output	D2000


I/O Mapping
FX3U-64DP-M
1: Slave_Nr_001 (ST1H-PB)
ST1H-PB 32pts.-word consistent
ST1PSD 2/ 2/ -/ -
ST1X4-DE1 4/ 4/ -/ -
ST1Y2-TPE3 2/ 2/ -/ -
ST1PDD 2/ 2/ -/ -
ST1AD2-V(without Ww) 4/ 4/ 2/ 0
ST1DA2-V(without Wr) 4/ 4/ 0/ 2

DUT Var. Identifier	Identifier	DUT Type	Number Ele	Class	Global	User MIT	Buffer MIT-Address
vHA0SLV1MOD2	Input1	BOOL	1	input			D1000.6
vHA0SLV1MOD2	Input2	BOOL	1	input			D1000.7
vHA0SLV1MOD2	Input3	BOOL	1	input			D1000.8
vHA0SLV1MOD2	Input4	BOOL	1	input			D1000.9
vHA0SLV1MOD2	ErrorInfo1	BOOL	1	input			D1002.6
vHA0SLV1MOD2	ErrorInfo2	BOOL	1	input			D1002.7
vHA0SLV1MOD2	ModuleStatus	BOOL	1	input			D1004.3
vHA0SLV1MOD2	ErrorClear	BOOL	1	output			D2002.6

⑩ Before download please select ***Transfer Setup***



Transfer Setup Names: FX-CPU (RS232) COM 1

Target PLC

PLC Series: FX3

PLC Type: FX3U

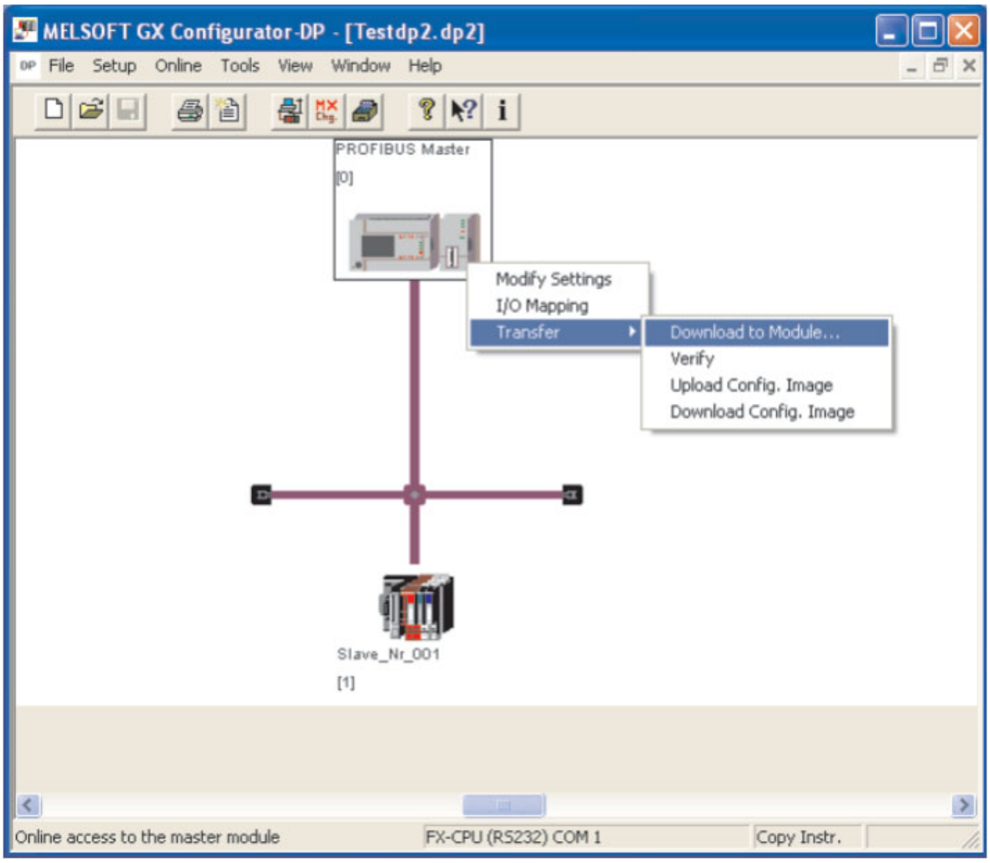
Module Slot: 0 [0-63]

COM: COM 1 Transmission speed: 9.6Kbps

Time out (Sec.): 10 Retry times: 0

Opens the transfer setup dialog. | FX-CPU (RS232) COM 1 | Copy Instr. | NUM

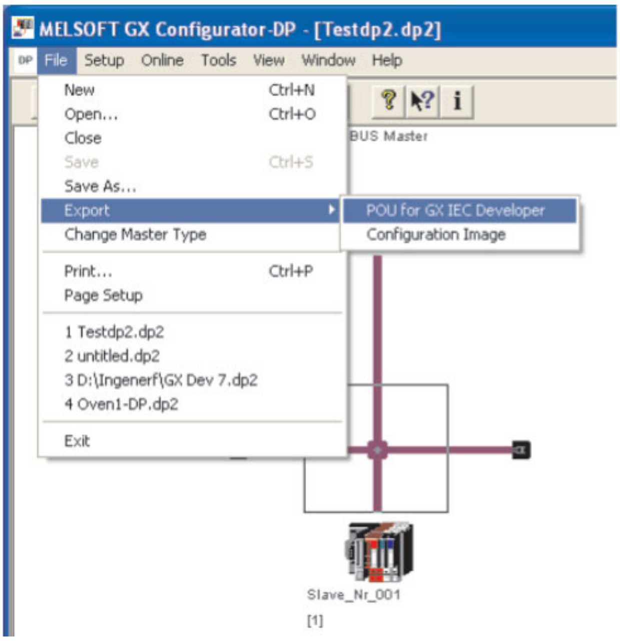
⑪ Transfer configuration to PROFIBUS/DP master module.

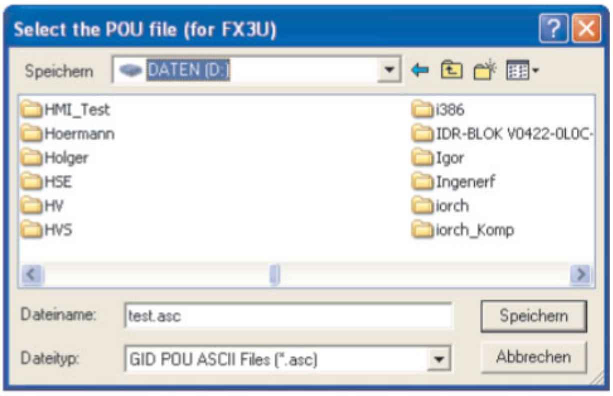


Online access to the master module | FX-CPU (RS232) COM 1 | Copy Instr.

⑫ POU for GX IEC Developer

The created POU can be exported to the GX IEC Developer project. This POU will initialize the PROFIBUS/DP master module in the PLC program.





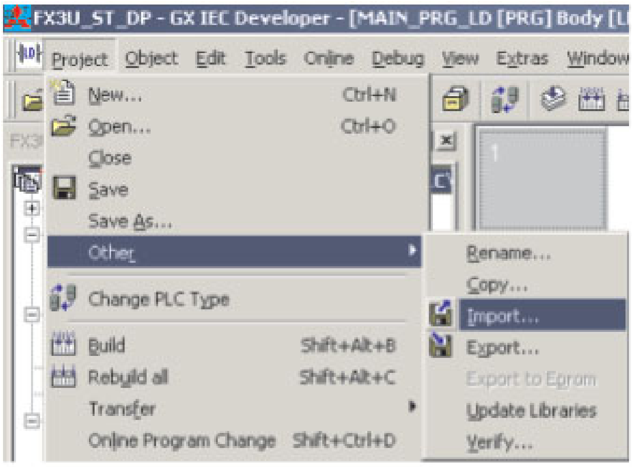
Speichern: DATEN (D:)

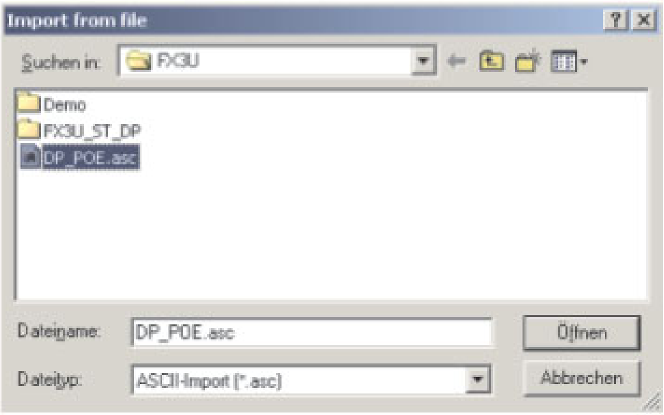
Dateiname: test.asc

Dateityp: GID POU ASCII Files (*.asc)

⑬ Import of the POU in the GX IEC Developer project.

(A new project with the correct CPU has already been created and saved.)

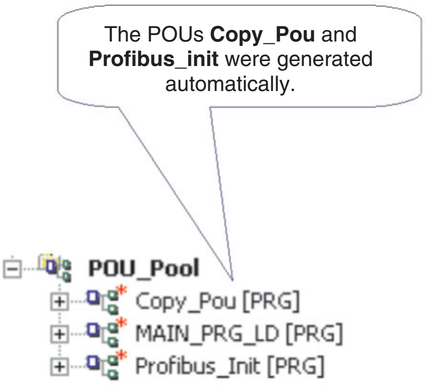




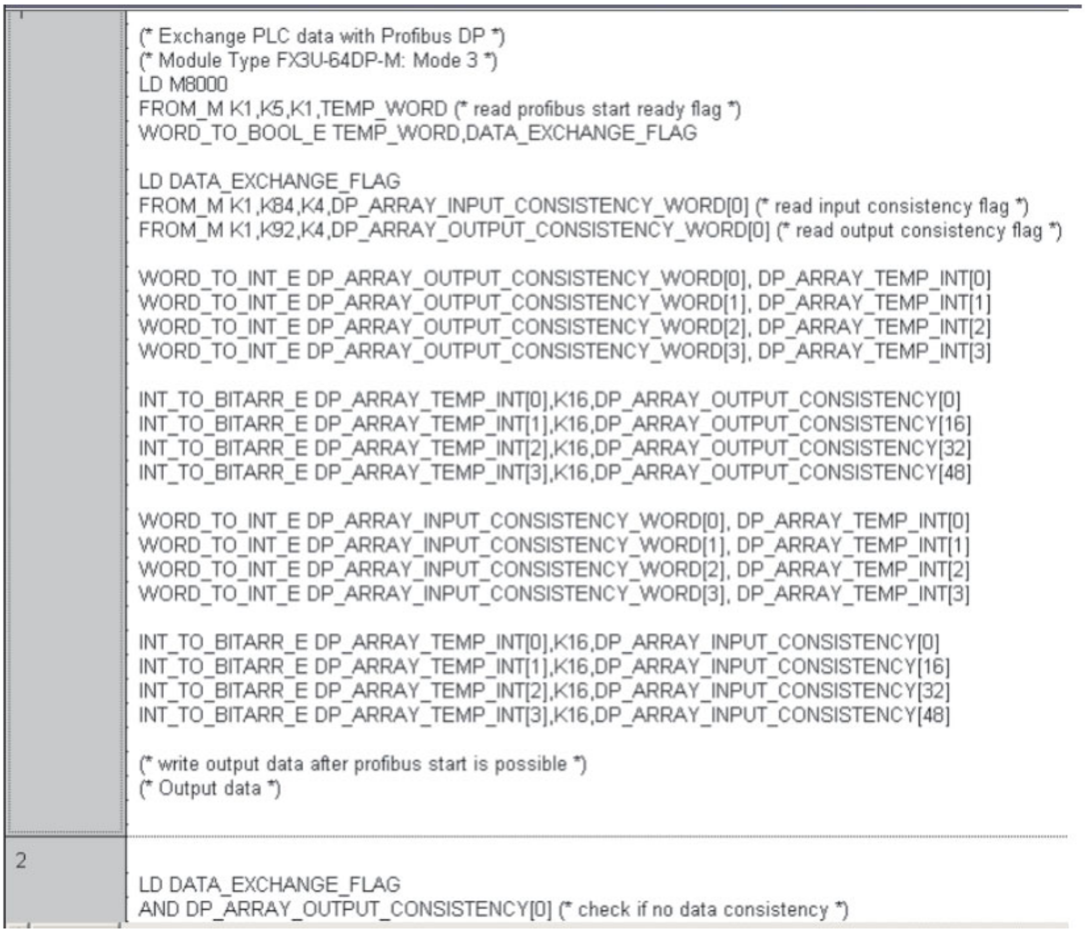
Suchen in: FX3U

Dateiname: DP_POE.asc

Dateityp: ASCII-Import (*.asc)



The POUs **Copy_Pou** and **Profibus_init** were generated automatically.



```
(* Exchange PLC data with Profibus DP *)
(* Module Type FX3U-64DP-M: Mode 3 *)
LD M8000
FROM_M K1,K5,K1,TEMP_WORD (* read profibus start ready flag *)
WORD_TO_BOOL_E TEMP_WORD,DATA_EXCHANGE_FLAG

LD DATA_EXCHANGE_FLAG
FROM_M K1,K84,K4,DP_ARRAY_INPUT_CONSISTENCY_WORD[0] (* read input consistency flag *)
FROM_M K1,K92,K4,DP_ARRAY_OUTPUT_CONSISTENCY_WORD[0] (* read output consistency flag *)

WORD_TO_INT_E DP_ARRAY_OUTPUT_CONSISTENCY_WORD[0], DP_ARRAY_TEMP_INT[0]
WORD_TO_INT_E DP_ARRAY_OUTPUT_CONSISTENCY_WORD[1], DP_ARRAY_TEMP_INT[1]
WORD_TO_INT_E DP_ARRAY_OUTPUT_CONSISTENCY_WORD[2], DP_ARRAY_TEMP_INT[2]
WORD_TO_INT_E DP_ARRAY_OUTPUT_CONSISTENCY_WORD[3], DP_ARRAY_TEMP_INT[3]

INT_TO_BITARR_E DP_ARRAY_TEMP_INT[0],K16,DP_ARRAY_OUTPUT_CONSISTENCY[0]
INT_TO_BITARR_E DP_ARRAY_TEMP_INT[1],K16,DP_ARRAY_OUTPUT_CONSISTENCY[16]
INT_TO_BITARR_E DP_ARRAY_TEMP_INT[2],K16,DP_ARRAY_OUTPUT_CONSISTENCY[32]
INT_TO_BITARR_E DP_ARRAY_TEMP_INT[3],K16,DP_ARRAY_OUTPUT_CONSISTENCY[48]

WORD_TO_INT_E DP_ARRAY_INPUT_CONSISTENCY_WORD[0], DP_ARRAY_TEMP_INT[0]
WORD_TO_INT_E DP_ARRAY_INPUT_CONSISTENCY_WORD[1], DP_ARRAY_TEMP_INT[1]
WORD_TO_INT_E DP_ARRAY_INPUT_CONSISTENCY_WORD[2], DP_ARRAY_TEMP_INT[2]
WORD_TO_INT_E DP_ARRAY_INPUT_CONSISTENCY_WORD[3], DP_ARRAY_TEMP_INT[3]

INT_TO_BITARR_E DP_ARRAY_TEMP_INT[0],K16,DP_ARRAY_INPUT_CONSISTENCY[0]
INT_TO_BITARR_E DP_ARRAY_TEMP_INT[1],K16,DP_ARRAY_INPUT_CONSISTENCY[16]
INT_TO_BITARR_E DP_ARRAY_TEMP_INT[2],K16,DP_ARRAY_INPUT_CONSISTENCY[32]
INT_TO_BITARR_E DP_ARRAY_TEMP_INT[3],K16,DP_ARRAY_INPUT_CONSISTENCY[48]

(* write output data after profibus start is possible *)
(* Output data *)

2
LD DATA_EXCHANGE_FLAG
AND DP_ARRAY_OUTPUT_CONSISTENCY[0] (* check if no data consistency *)
```

⑭ Rebuild the GX IEC Developer project and transfer it to the FX3U. After restarting the PLC the PROFIBUS communication will start.

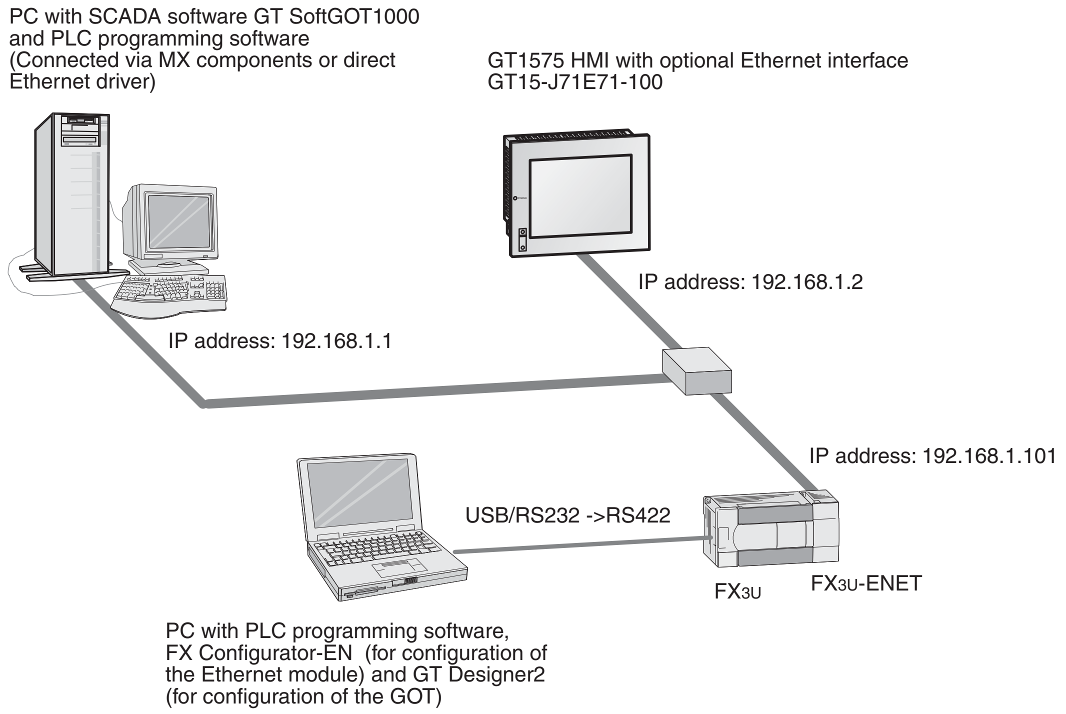
18 Ethernet Communications

This section provides a step-by-step guide to setting up a Ethernet module FX3U-ENET using FX Configurator-EN.

As an example, this section will show how to set up a module for allowing TCP/IP communications between a FX3U, a SCADA PC and a GT15 HMI*. Also shown is how the programming software can be configured to communicate with the FX3U via Ethernet once the settings have been made.

The diagram below shows the layout of the example Ethernet network. Proposed IP addresses are shown next to the Ethernet nodes.

Please note that more attention is given to the set up of the PLC than the PC or HMI, as the user may require more specific settings than this section covers.

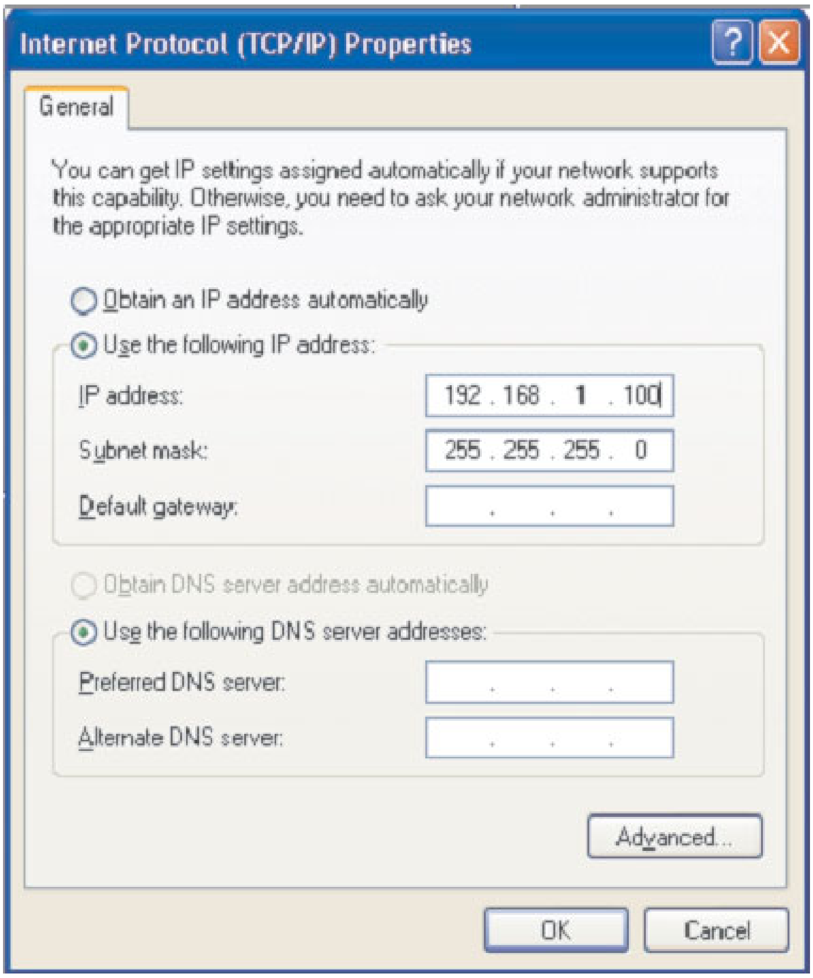

PC with SCADA software GT SoftGOT1000
and PLC programming software
(Connected via MX components or direct
Ethernet driver)
GT1575 HMI with optional Ethernet interface
GT15-J71E71-100
IP address: 192.168.1.2
IP address: 192.168.1.1
IP address: 192.168.1.101
USB/RS232 ->RS422
FX3U
FX3U-ENET
PC with PLC programming software,
FX Configurator-EN (for configuration of
the Ethernet module) and GT Designer2
(for configuration of the GOT)

* For the case that a HMI of the E1000 series is used instead of a GOT, the settings in the software E-Designer are shown in section 18.5.

18.1 Configuring the PC on the Ethernet

Open the Network properties of Windows, and assign an IP address and subnet mask in the TCP/IP properties dialogue for the Ethernet network adapter to be used in the PC.

Please note that after changing IP address, the PC may require a restart.



Internet Protocol (TCP/IP) Properties

General

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

○ Obtain an IP address automatically

● Use the following IP address:

IP address: 192 . 168 . 1 . 100

Subnet mask: 255 . 255 . 255 . 0

Default gateway: . . .

○ Obtain DNS server address automatically

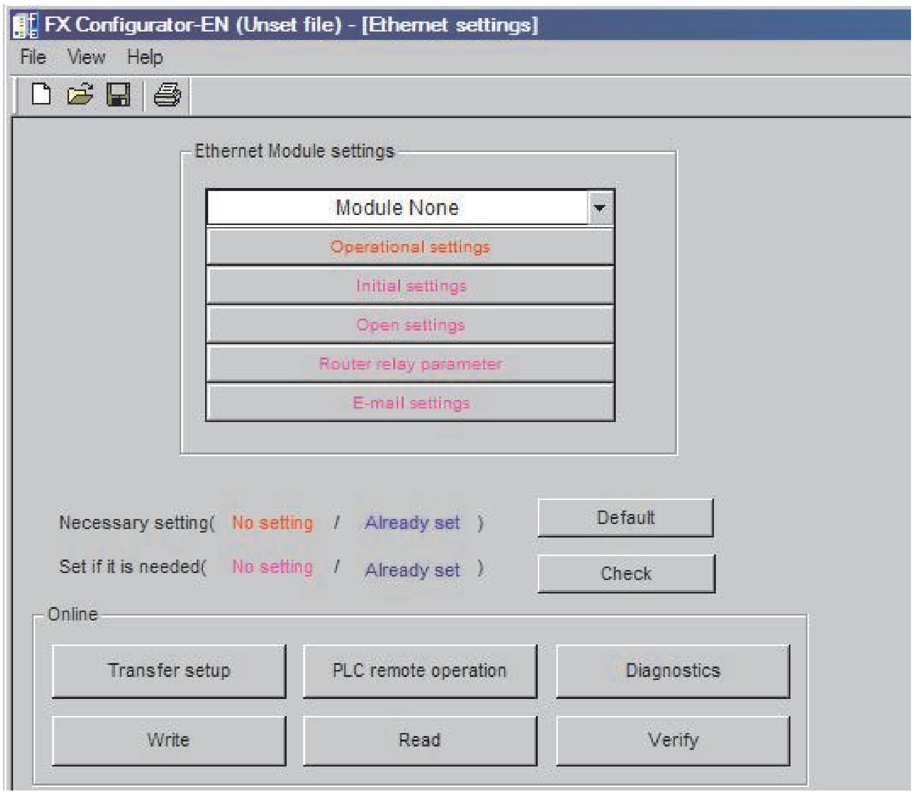
● Use the following DNS server addresses:

Preferred DNS server: . . .

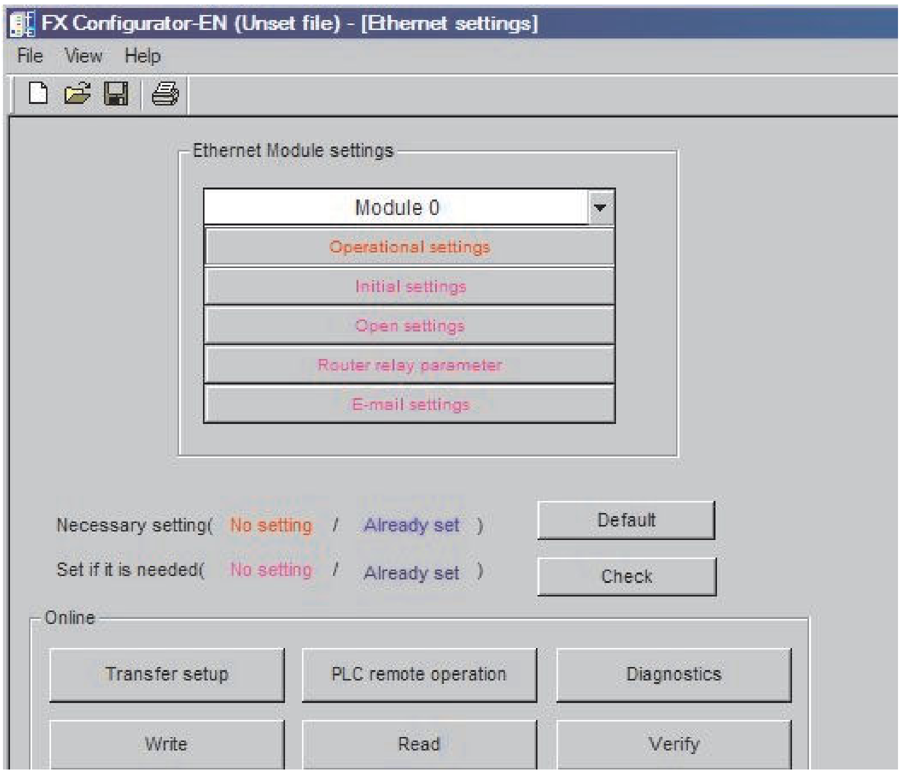
Alternate DNS server: . . .

18.2 Configuring the FX3U-ENET by FX Configurator-EN

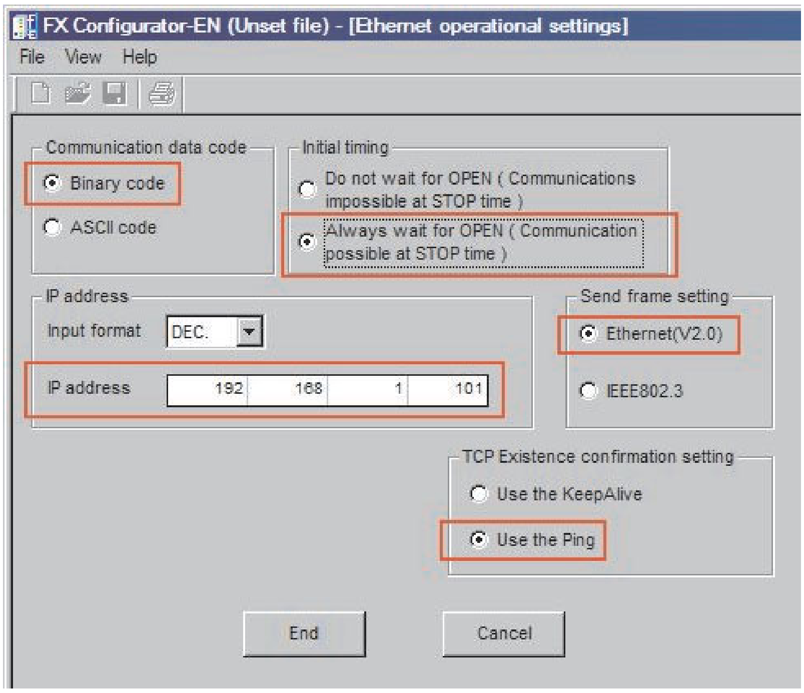
① Open the FX Configurator-EN and start the setting of the ETHERNET module FX3U-ENET.



② Now select the special function module address of the FX3U-ENET. Special function modules connected to the right side of the base unit are counted from left to right. If the FX3U-ENET is the first special function module select **Module 0**.



③ Open the **Operational Settings** and take over the settings shown below in red frames. The IP address 192.168.1.101 of the FX3U-ENET corresponds to the requirements of your network if your network IP is 192.168.1.1.



FX Configurator-EN (Unset file) - [Ethernet operational settings]

Communication data code
● Binary code
○ ASCII code

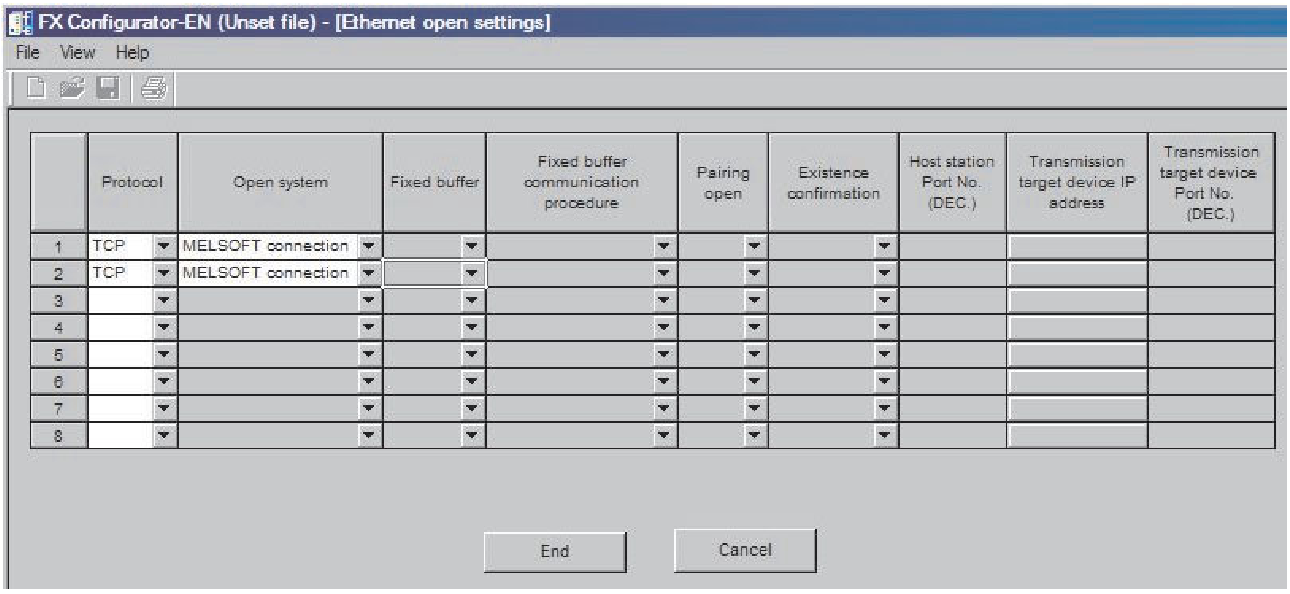
Initial timing
○ Do not wait for OPEN (Communications impossible at STOP time)
● Always wait for OPEN (Communication possible at STOP time)

IP address
Input format DEC.
IP address 192 168 1 101

Send frame setting
● Ethernet(V2.0)
○ IEEE802.3

TCP Existence confirmation setting
○ Use the KeepAlive
● Use the Ping

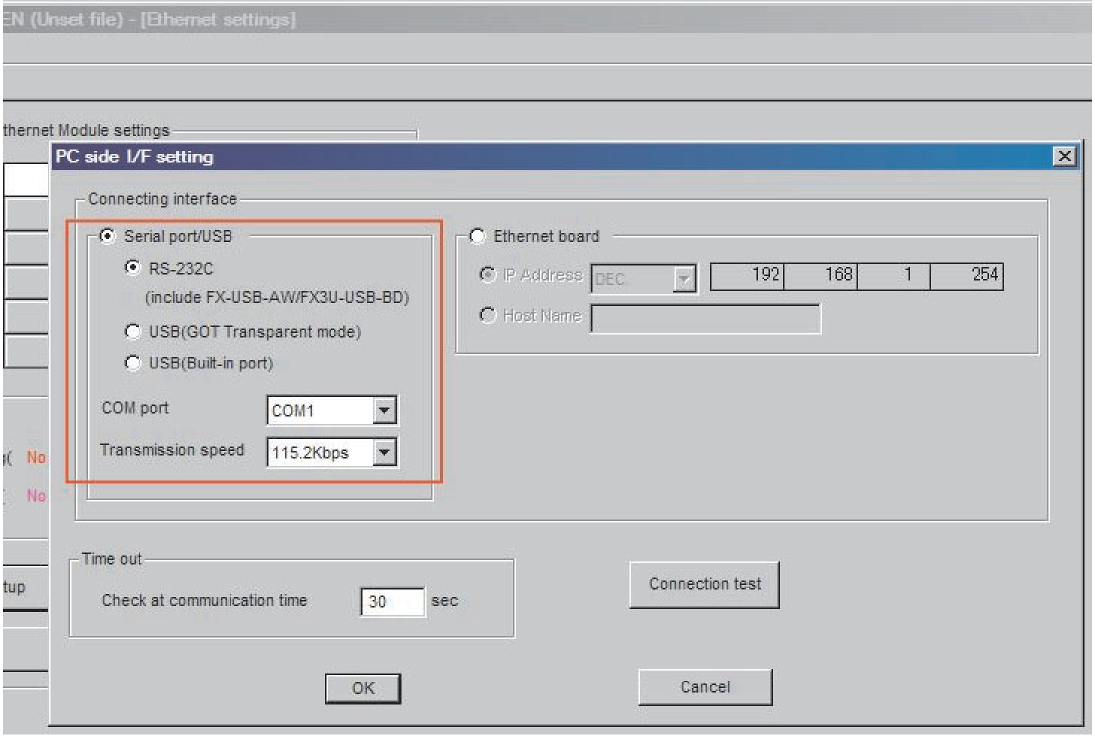
④ Next, open the **Open Settings** and take over the following settings.



FX Configurator-EN (Unset file) - [Ethernet open settings]

	Protocol	Open system	Fixed buffer	Fixed buffer communication procedure	Pairing open	Existence confirmation	Host station Port No. (DEC.)	Transmission target device IP address	Transmission target device Port No. (DEC.)
1	TCP	MELSOFT connection							
2	TCP	MELSOFT connection							
3									
4									
5									
6									
7									
8									

⑤ In the **Ethernet Module settings**, click on ***Transfer Setup*** and take over the settings shown below in the red frame.



PC side I/F setting

Connecting interface

● Serial port/USB

● RS-232C
(include FX-USB-AW/FX3U-USB-BD)

○ USB(GOT Transparent mode)

○ USB(Built-in port)

COM port: COM1

Transmission speed: 115.2Kbps

○ Ethernet board

IP Address: DEC 192 168 1 254

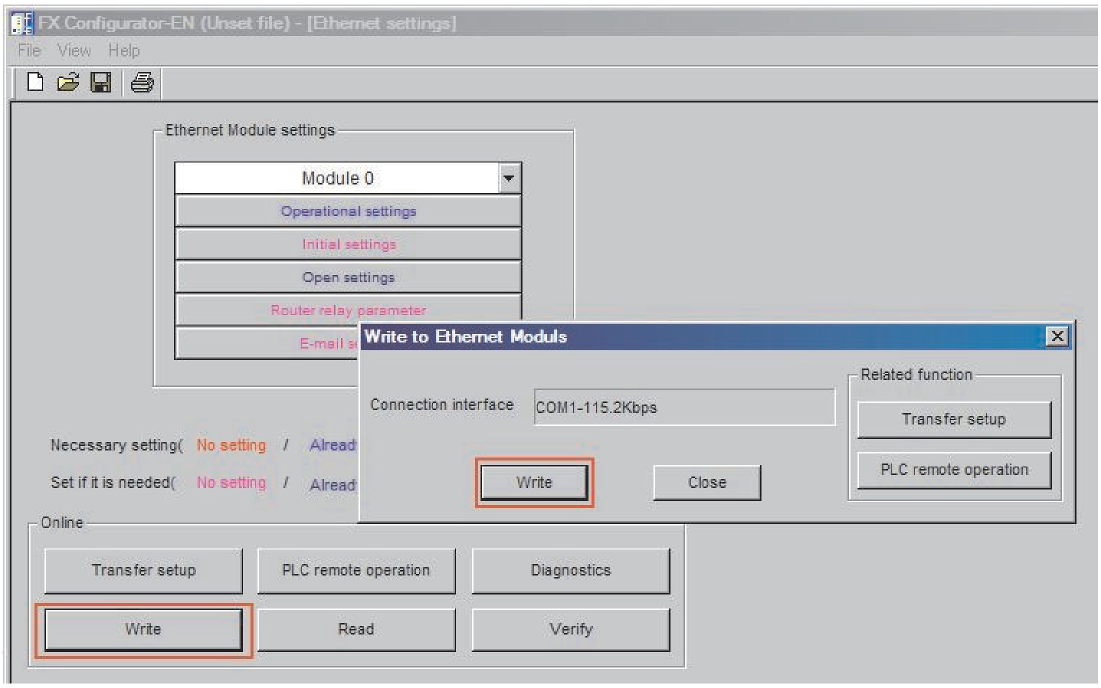
Host Name:

Time out

Check at communication time: 30 sec

⑥ Click on ***Write*** in the **Ethernet Module settings**. As you see, the transfer speed for COM1 is set to 115.2 Kbps.

In the dialogue window **Write to Ethernet Module** click on ***Write*** and transfer your settings to the PLC. Confirm displayed messages with ***YES*** resp. ***OK***.



Write to Ethernet Moduls

Connection interface: COM1-115.2Kbps

Necessary setting(No setting / Alread

Set if it is needed(No setting / Alread

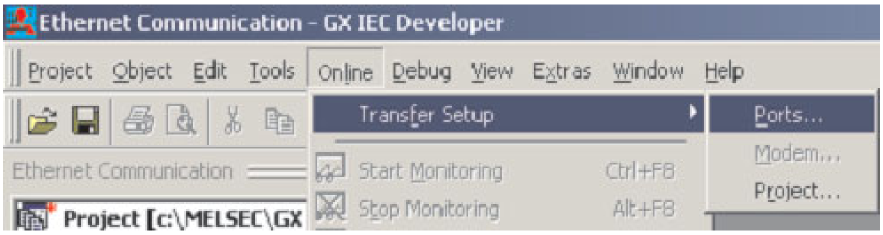
Now you can confirm the completion of the initial processing by issuing a PING command to the FX3U-ENET. The ping command is provided by Microsoft® Windows. Shown below is an example for normal completion.

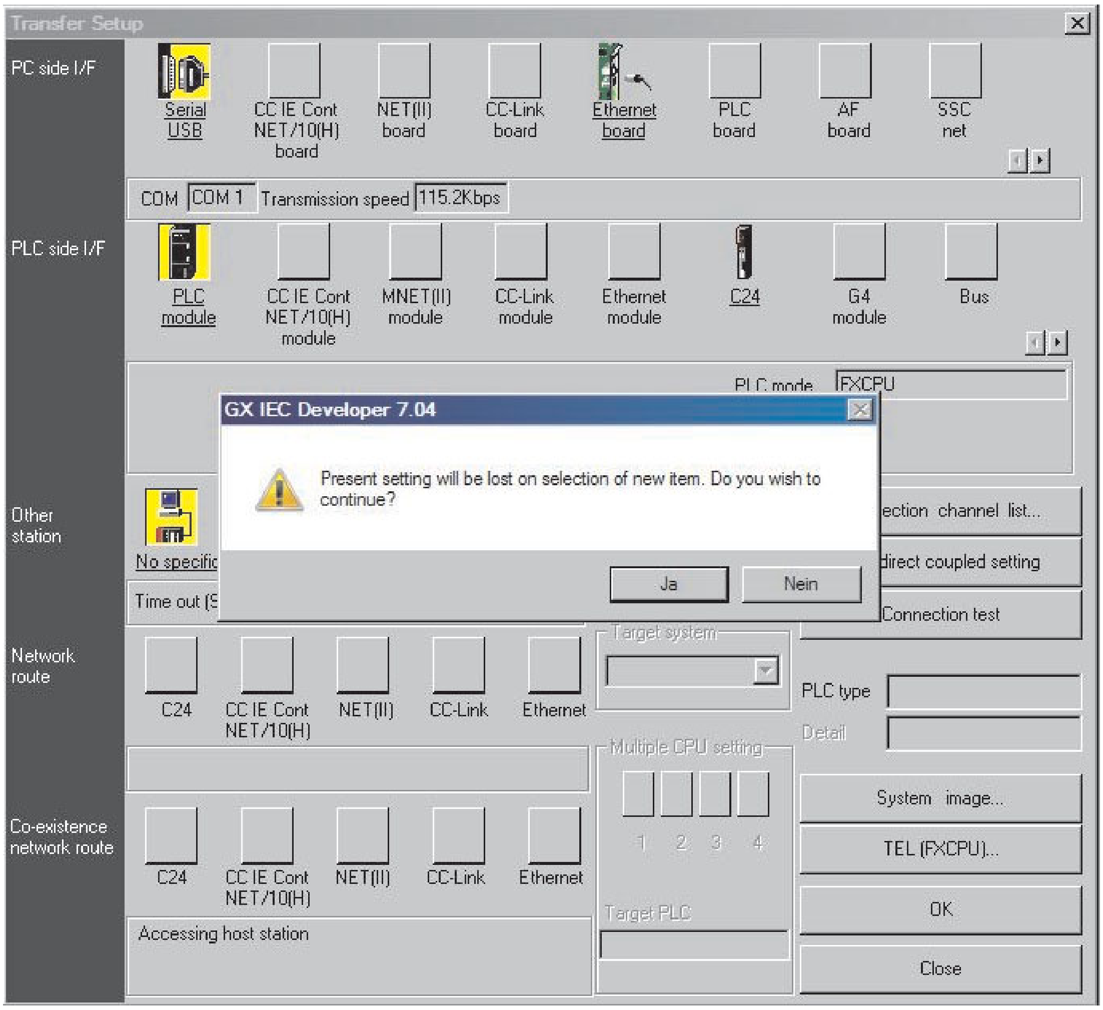
```
C:\>ping 192.0.1.254…Remark: Execute the ping command
Pinging 192.168.1.101 with 32 bytes of data:
Reply from 192.168.1.101: bytes=32 time=1ms TTL=250
Reply from 192.168.1.101: bytes=32 time=1ms TTL=250
Reply from 192.168.1.101: bytes=32 time=1ms TTL=250
Reply from 192.168.1.101: bytes=32 time=1ms TTL=250
Ping statistics for 192.168.1.101:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss)
Approximate round trip times in milli-seconds:
Minimum = 1ms, Maximum = 1ms, Average = 1ms
```

18.3 Configuring GX IEC Developer to access the PLC on Ethernet

Please make the following settings in order to access the PLC via an Ethernet network and an Ethernet interface module.

① From the **Online** menu, select ***Transfer Setup*** and then ***Ports***:





COM COM 1 Transmission speed 115.2Kbps

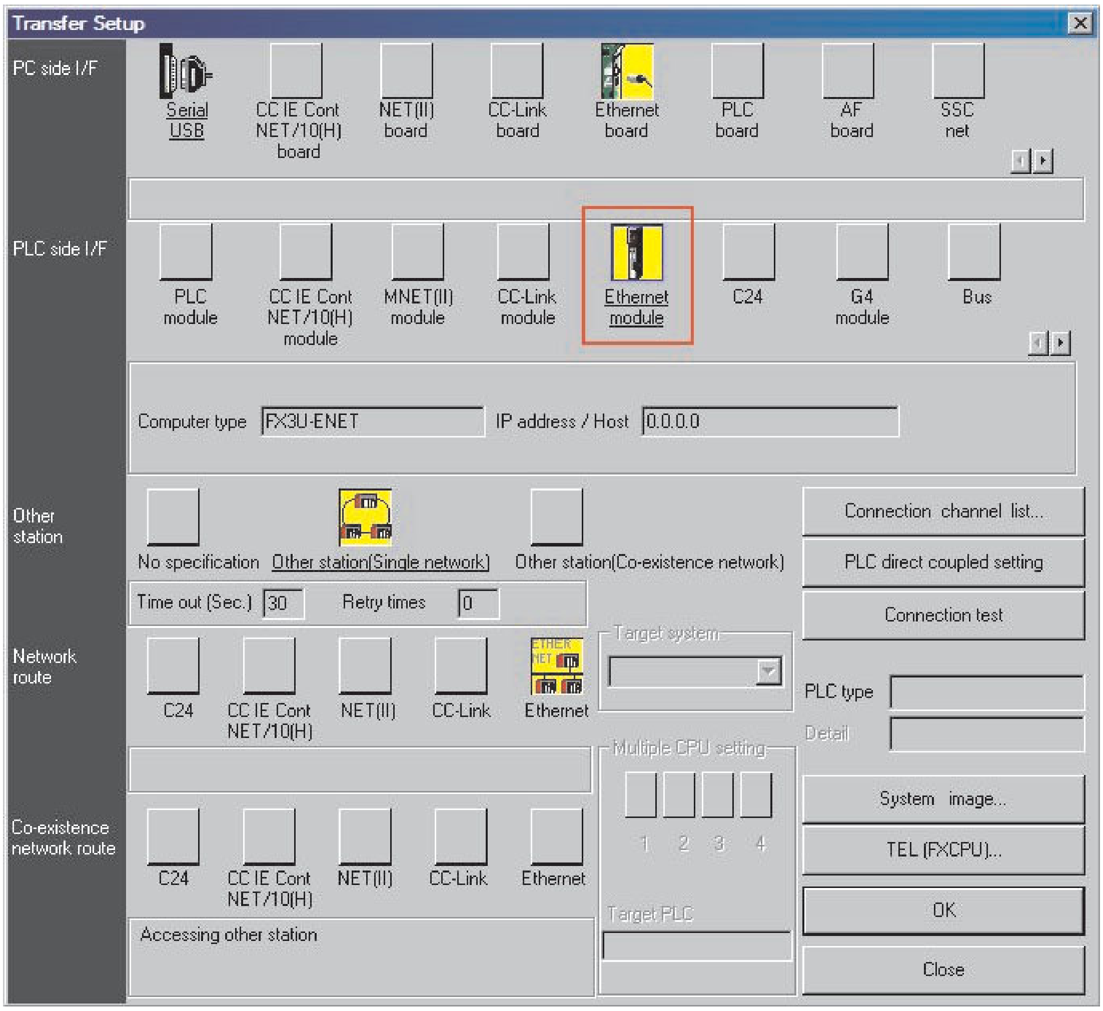
PLC mode FXCPU

Time out (S

Accessing host station

Present setting will be lost on selection of new item. Do you wish to continue?

② The default connection is for the **PC Side I/F** to use serial connection to the PLC CPU module. Change the **PC Side I/F** to ***Ethernet board*** by clicking on it as shown above, and saying ***Yes*** to the question about present setting will be lost (i.e. the setting of serial to CPU).



Computer type: FX3U-ENET

IP address / Host: 0.0.0.0

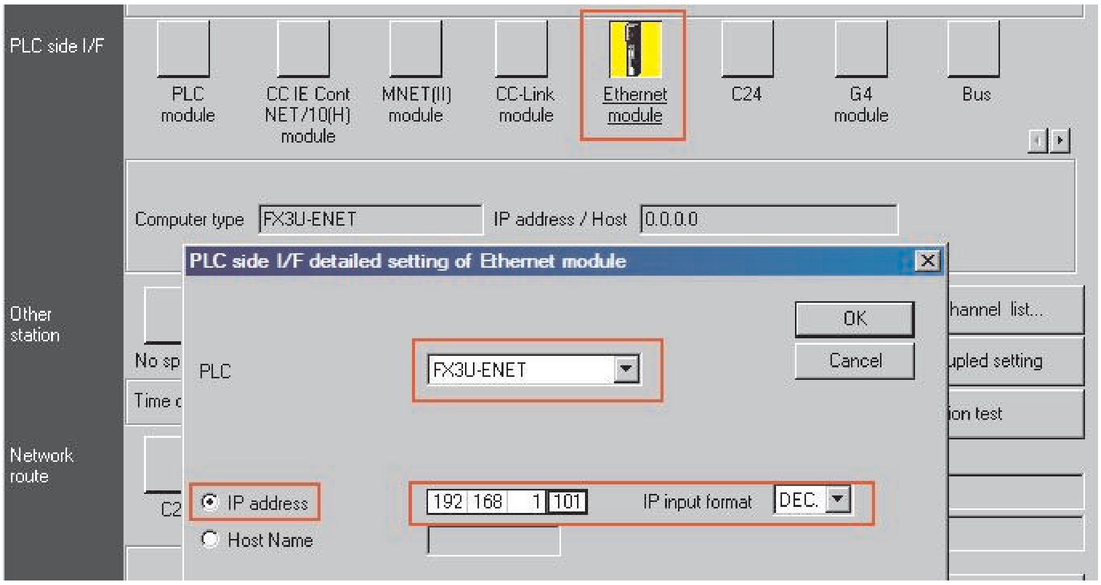
Time out (Sec.): 30

Retry times: 0

③ Next, double click on ***Ethernet module*** under **PLC side I/F** as shown above. This will open up the dialogue to allow the settings for the Ethernet interface module used.

NOTE

There is no need to specify a port number, as the programming software will use a MELSOFT Protocol dedicated port by default.



Computer type: FX3U-ENET

IP address / Host: 0.0.0.0

PLC: FX3U-ENET

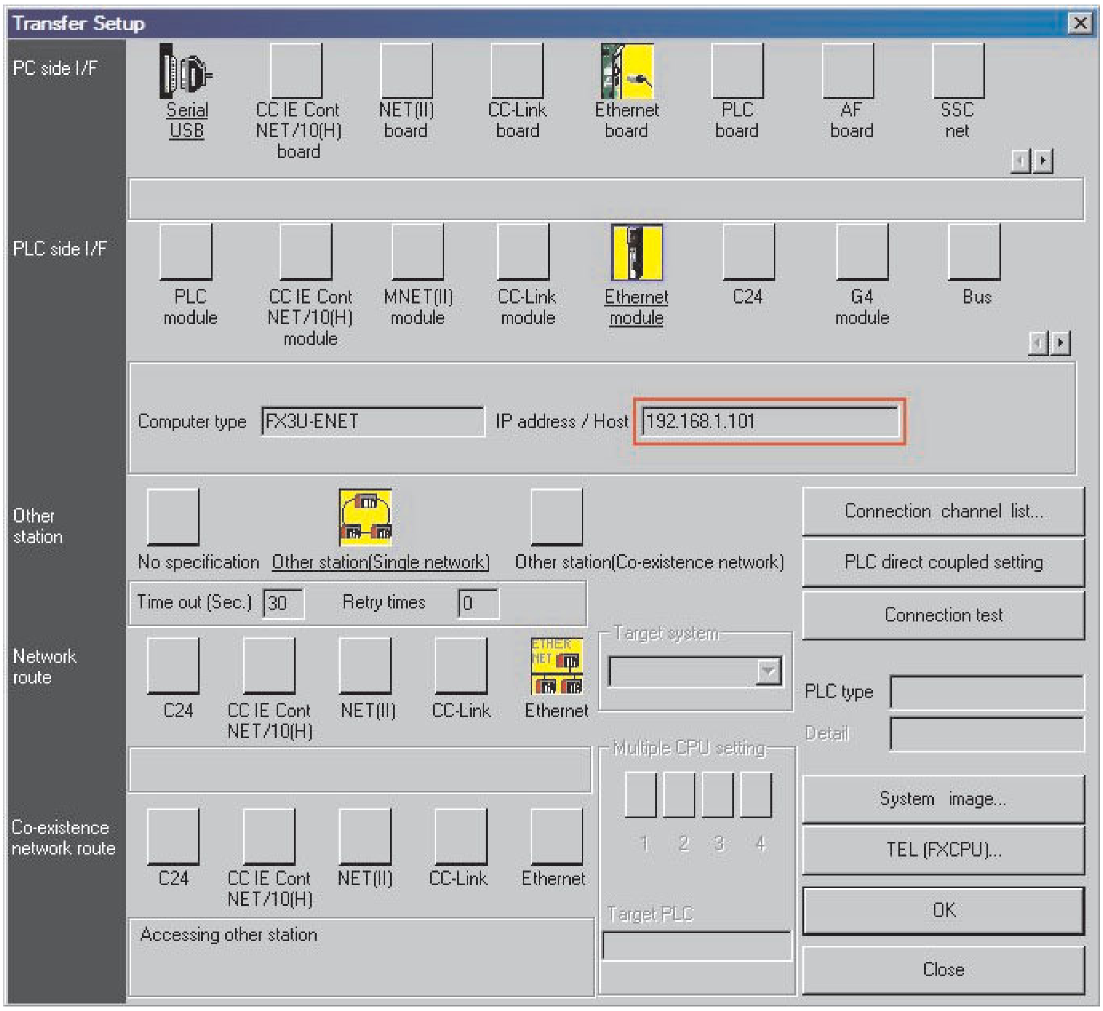
● IP address: 192 168 1 101

IP input format: DEC.

④ Click ***OK*** when done.

This will complete the setting, making the dialogue look as shown below.

⑤ Click ***Connection test*** to confirm the settings are correct. Then click ***OK*** when finished.



Transfer Setup

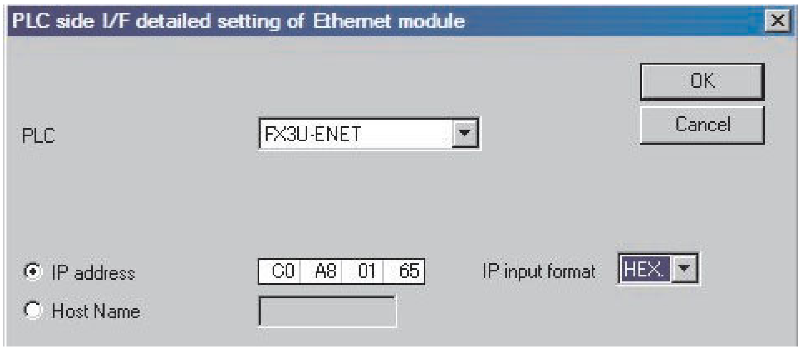
Computer type: FX3U-ENET

IP address / Host: 192.168.1.101

Time out (Sec.): 30

Retry times: 0

NOTE The IP address can be entered also in hexadecimal format. This option is shown in the following two figures.



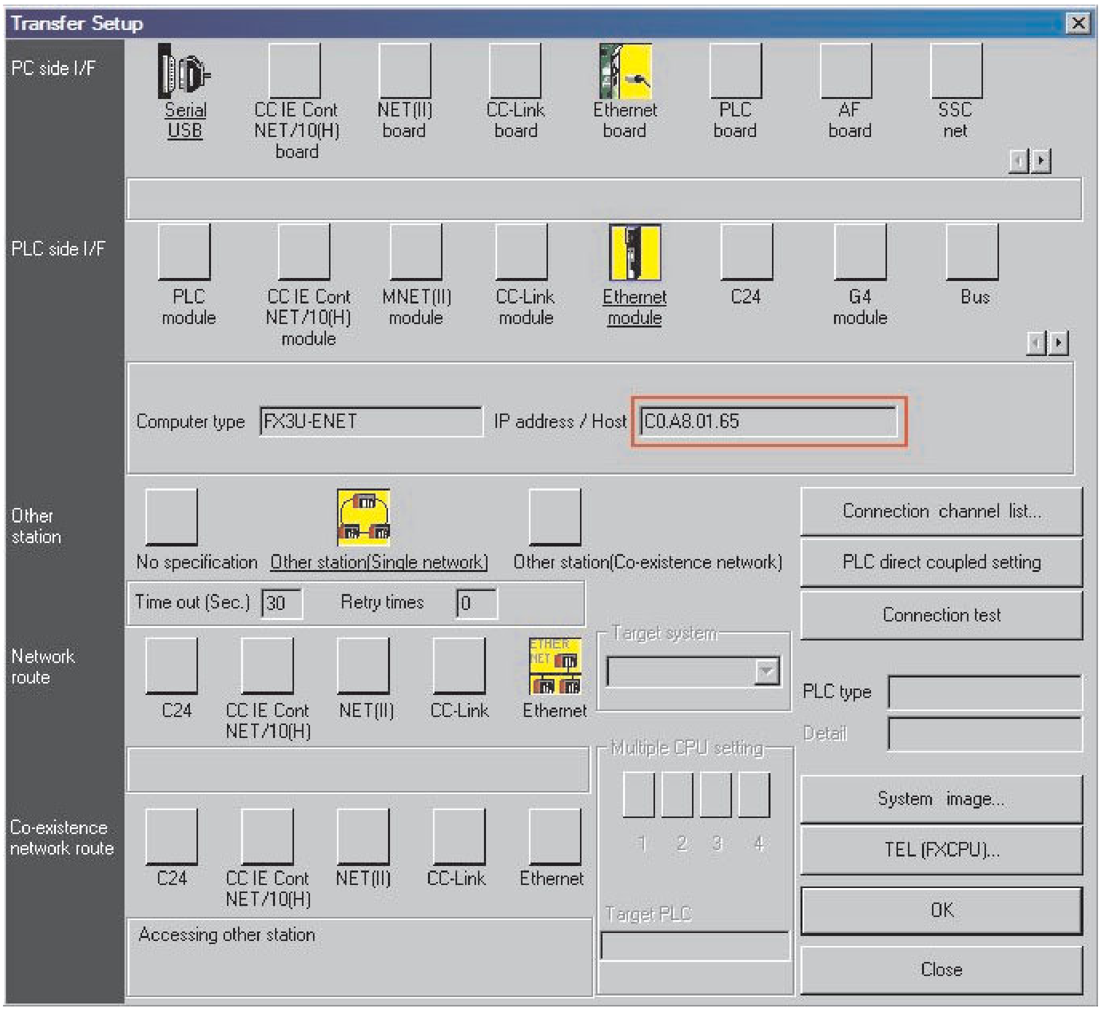
PLC side I/F detailed setting of Ethernet module

PLC: FX3U-ENET

● IP address: C0 A8 01 65

○ Host Name

IP input format: HEX



Computer type: FX3U-ENET

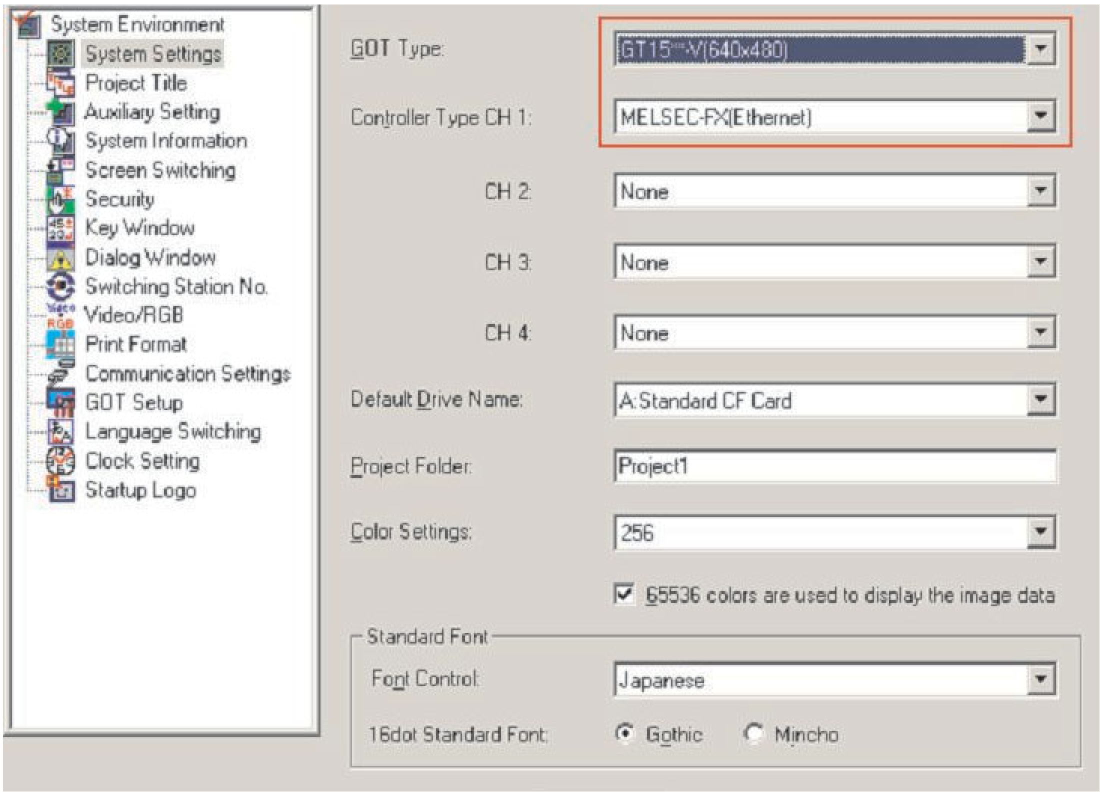
IP address / Host: C0.A8.01.65

Time out (Sec.): 30

Retry times: 0

18.4 Setting up a HMI of the GOT1000 Series (GT12, GT15 or GT16)

① Please start GT-Designer2 and open a new project. In the Project Navigator window, double click on **System Settings** and make the settings shown below.



GOT Type: GT15**-V(640x480)

Controller Type CH 1: MELSEC-FX(Ethernet)

CH 2: None

CH 3: None

CH 4: None

Default Drive Name: A:Standard CF Card

Project Folder: Project1

Color Settings: 256

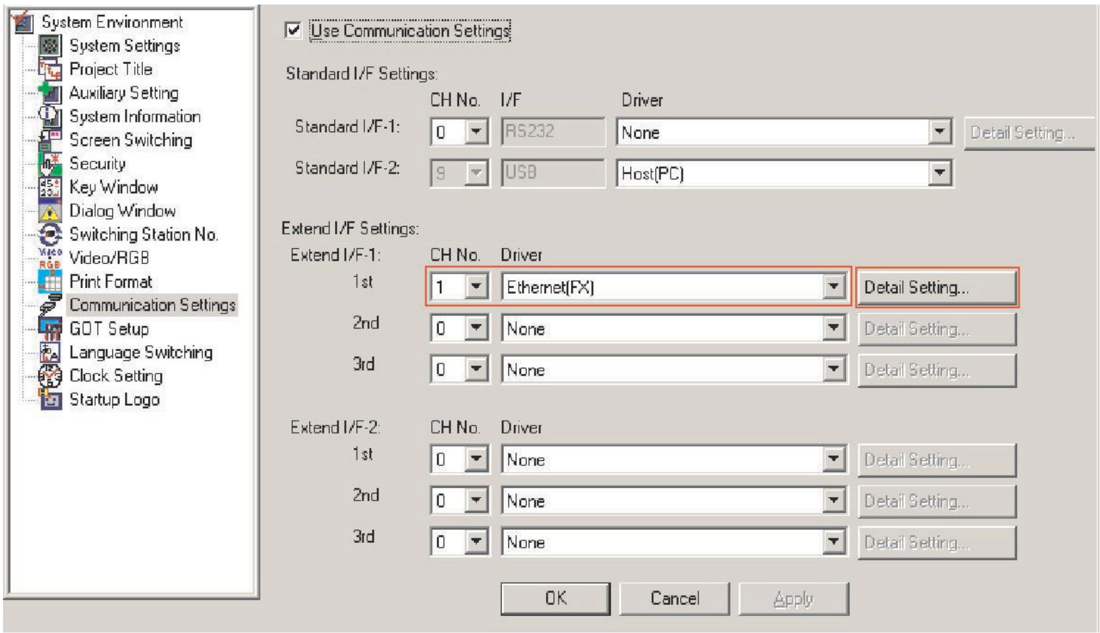
☑ 65536 colors are used to display the image data

Standard Font

Font Control: Japanese

16dot Standard Font: ● Gothic ○ Mincho

② In the **System Enviroment**, double click on **Communication Settings** and make the settings shown below.



☑ Use Communication Settings

Standard I/F Settings:

	CH No.	I/F	Driver	
Standard I/F-1:	0	RS232	None	Detail Setting...
Standard I/F-2:	9	USB	Host(PC)	

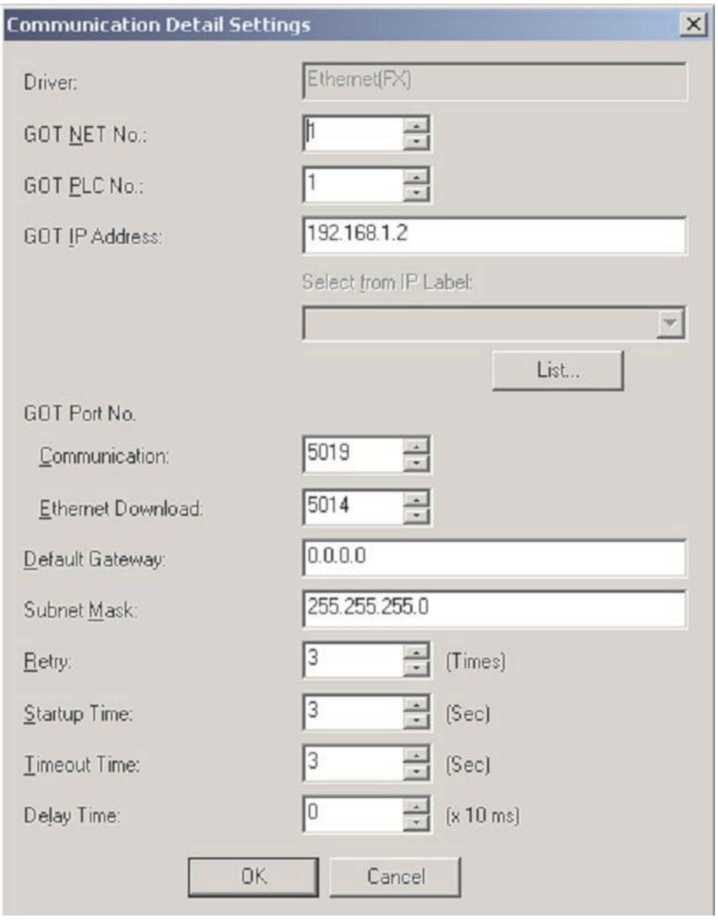
Extend I/F Settings:

Extend I/F-1:	CH No.	Driver	
1st	1	Ethernet(FX)	Detail Setting...
2nd	0	None	Detail Setting...
3rd	0	None	Detail Setting...

Extend I/F-2:	CH No.	Driver	
1st	0	None	Detail Setting...
2nd	0	None	Detail Setting...
3rd	0	None	Detail Setting...

③ Then, in the **Communication Settings** window, click on ***Detail Setting***.

④ As shown on the right, enter the details of the network in use and the IP address of the GOT.



Communication Detail Settings

Driver: Ethernet(FX)

GOT NET No.: 1

GOT PLC No.: 1

GOT IP Address: 192.168.1.2

Select from IP Label:

GOT Port No.

Communication: 5019

Ethernet Download: 5014

Default Gateway: 0.0.0.0

Subnet Mask: 255.255.255.0

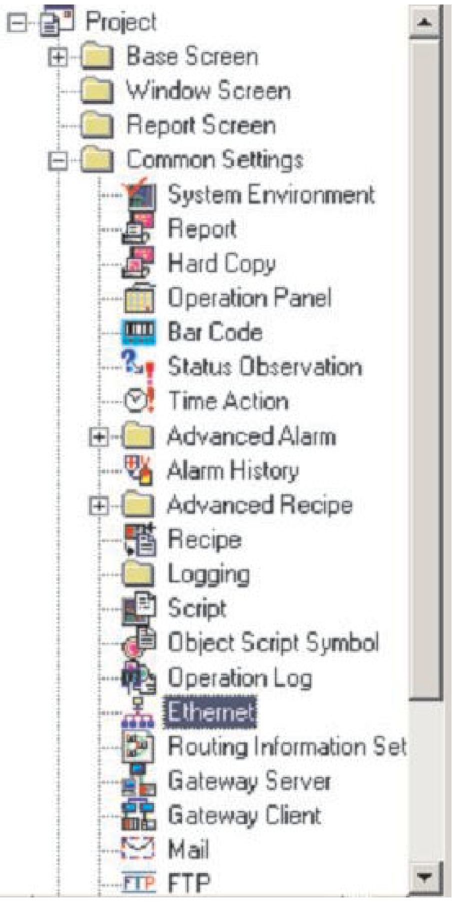
Retry: 3 (Times)

Startup Time: 3 (Sec)

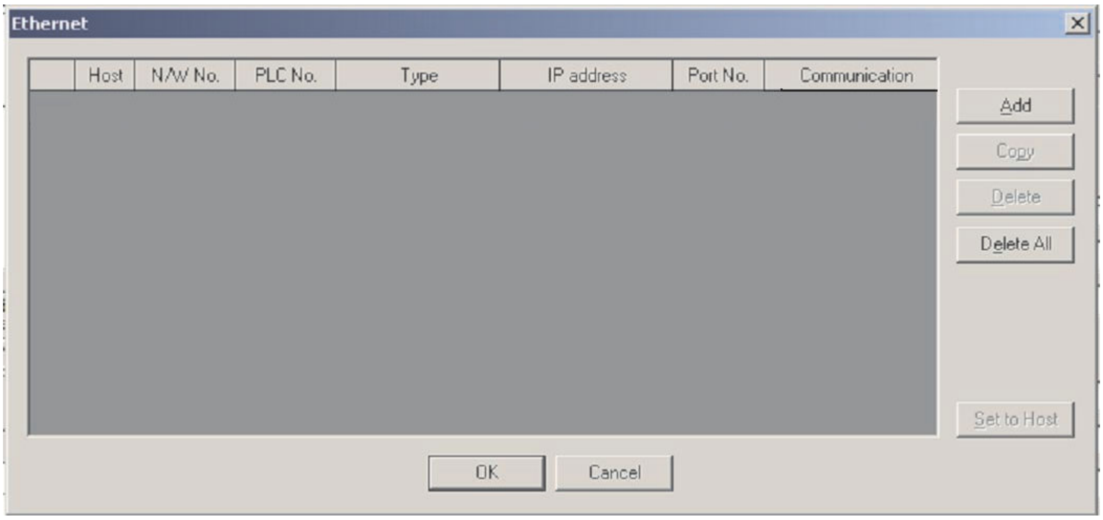
Timeout Time: 3 (Sec)

Delay Time: 0 (x 10 ms)

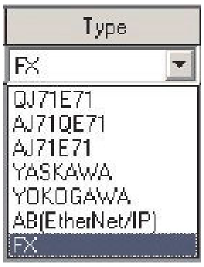
⑤ In the Project Navigator, select:
Common Settings -> **Ethernet**
to set up the connected PLC and the associated IP address.



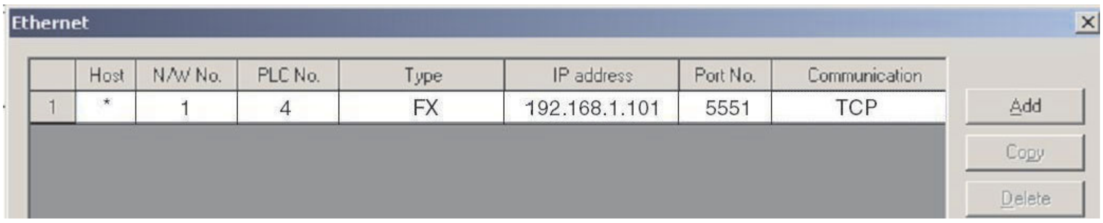
⑥ The following dialogue window will be displayed. Click on ***Add***.



⑦ Click on the **Type** column and select the type of PLC.



⑧ When selecting the PLC, certain settings (e.g. the Port No.) are taken over as defaults. Please make the remaining settings.

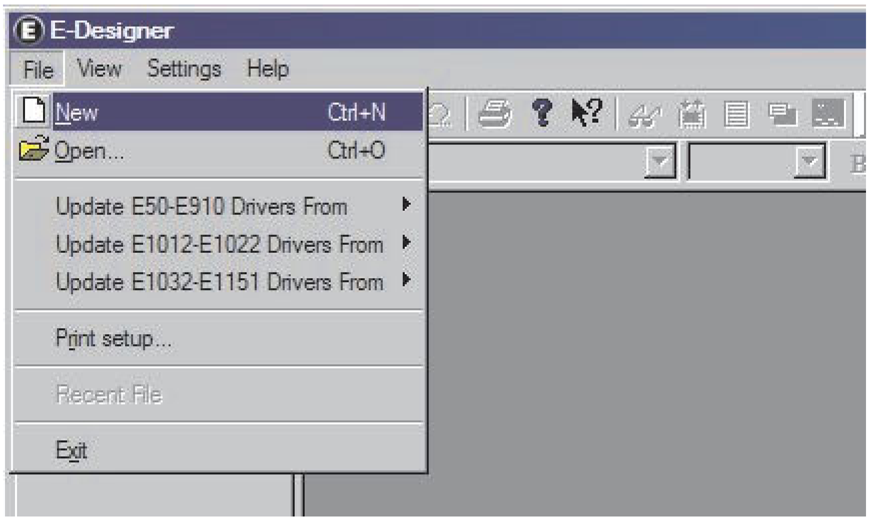


	Host	N/W No.	PLC No.	Type	IP address	Port No.	Communication
1	*	1	4	FX	192.168.1.101	5551	TCP

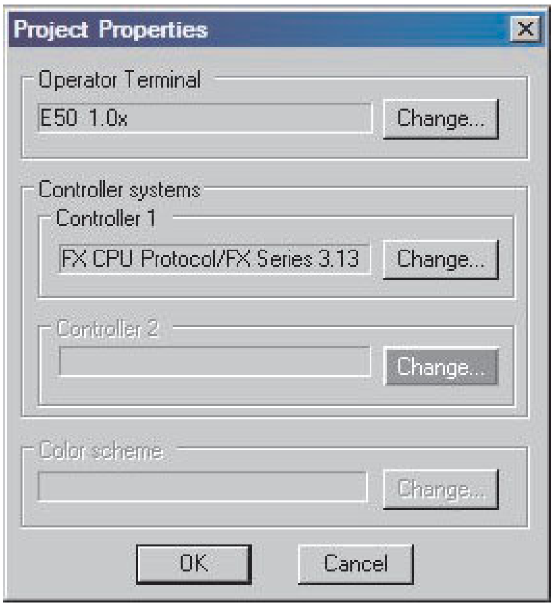
18.5 Setting up a HMI of the E1000 series

The settings shown in this section are only necessary when a HMI of the E1000 series is connected to the network instead of a GOT series HMI.

① Please open a new E-Designer project.


E-Designer
File
View
Settings
Help
New Ctrl+N
Open... Ctrl+O
Update E50-E910 Drivers From
Update E1012-E1022 Drivers From
Update E1032-E1151 Drivers From
Print setup...
Recent File
Exit

② Next, a dialogue window allowing the settings for the HMI used and the connected PLC is opened.



Operator Terminal: E50 1.0x

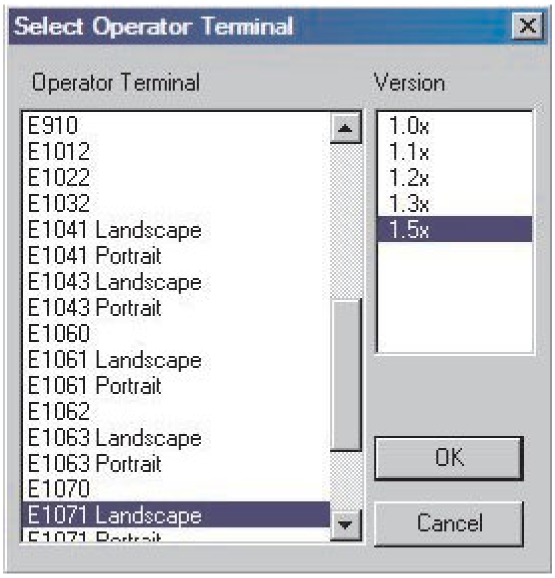
Controller systems

Controller 1: FX CPU Protocol/FX Series 3.13

Controller 2:

Color scheme:

③ Select the operator terminal.



Select Operator Terminal

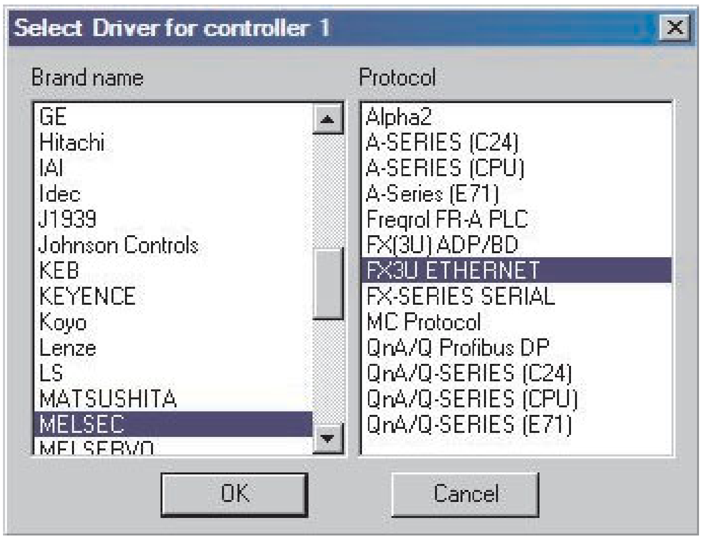
Operator Terminal

E910
E1012
E1022
E1032
E1041 Landscape
E1041 Portrait
E1043 Landscape
E1043 Portrait
E1060
E1061 Landscape
E1061 Portrait
E1062
E1063 Landscape
E1063 Portrait
E1070
E1071 Landscape
E1071 Portrait

Version

1.0x
1.1x
1.2x
1.3x
1.5x

④ Select the PLC



Select Driver for controller 1

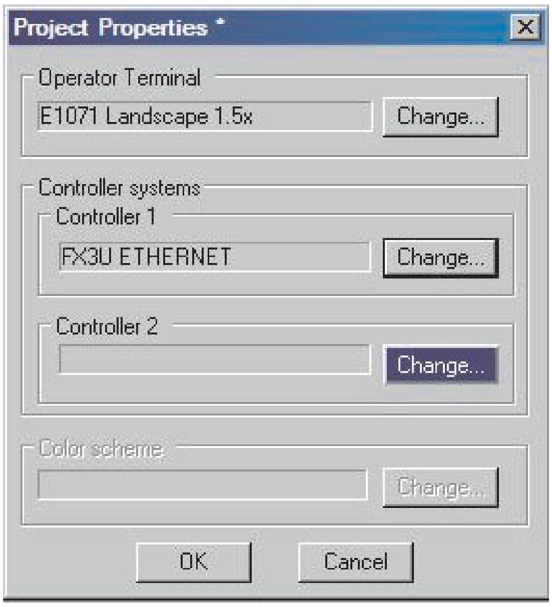
Brand name

GE
Hitachi
IAI
Idec
J1939
Johnson Controls
KEB
KEYENCE
Koyo
Lenze
LS
MATSUSHITA
MELSEC
MELSERVO

Protocol

Alpha2
A-SERIES (C24)
A-SERIES (CPU)
A-Series (E71)
Freqrol FR-A PLC
FX(3U) ADP/BD
FX3U ETHERNET
FX-SERIES SERIAL
MC Protocol
QnA/Q Profibus DP
QnA/Q-SERIES (C24)
QnA/Q-SERIES (CPU)
QnA/Q-SERIES (E71)

⑤ Click ***OK*** to confirm the selection.



Project Properties *

Operator Terminal: E1071 Landscape 1.5x

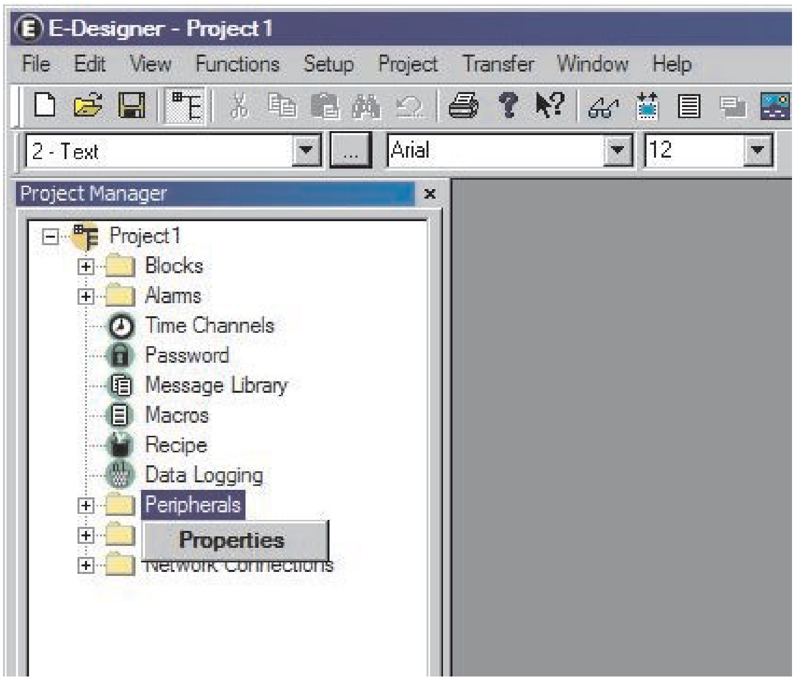
Controller systems

Controller 1: FX3U ETHERNET

Controller 2:

Color scheme:

⑥ Open the properties of the peripherals. (Right click on ***peripherals***, than click on ***properties***.)



E-Designer - Project1

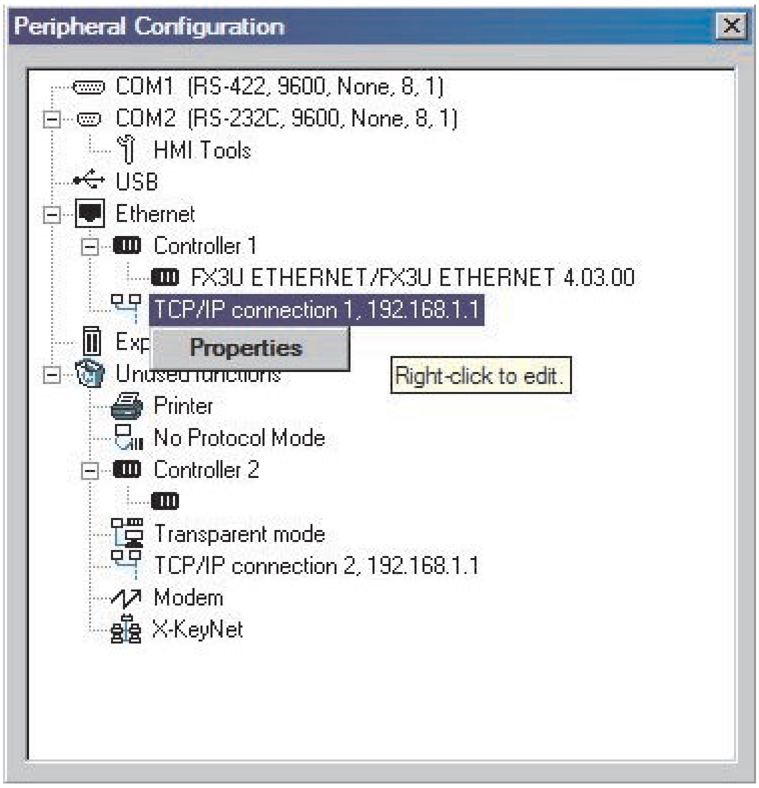
File Edit View Functions Setup Project Transfer Window Help

2 - Text | Arial | 12

Project Manager

- Project1
  - Blocks
  - Alarms
  - Time Channels
  - Password
  - Message Library
  - Macros
  - Recipe
  - Data Logging
  - Peripherals
  - Properties
  - Network Connections

⑦ Open the properties of the TCP/IP connection.

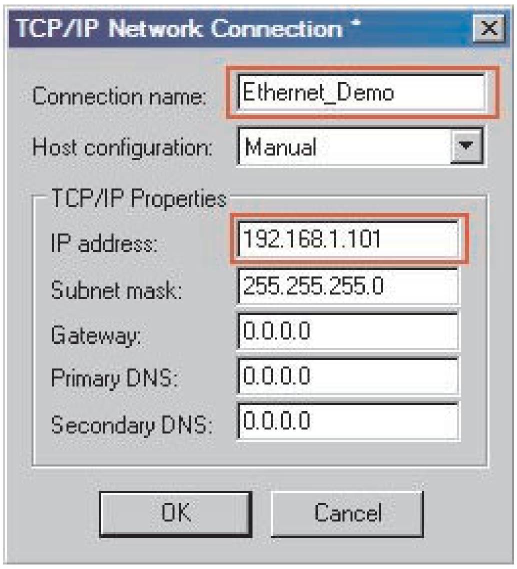


Peripheral Configuration

- COM1 (RS-422, 9600, None, 8, 1)
- COM2 (RS-232C, 9600, None, 8, 1)
  - HMI Tools
- USB
- Ethernet
  - Controller 1
    - FX3U ETHERNET/FX3U ETHERNET 4.03.00
  - TCP/IP connection 1, 192.168.1.1
- Exp
- Properties
- Unused functions
  - Printer
  - No Protocol Mode
  - Controller 2
  - Transparent mode
  - TCP/IP connection 2, 192.168.1.1
  - Modem
  - X-KeyNet

Right-click to edit.

⑧ Please enter a name for the connection and the IP address of the FX3U-ENET used for the connection.



Connection name: Ethernet_Demo

Host configuration: Manual

TCP/IP Properties

IP address: 192.168.1.101

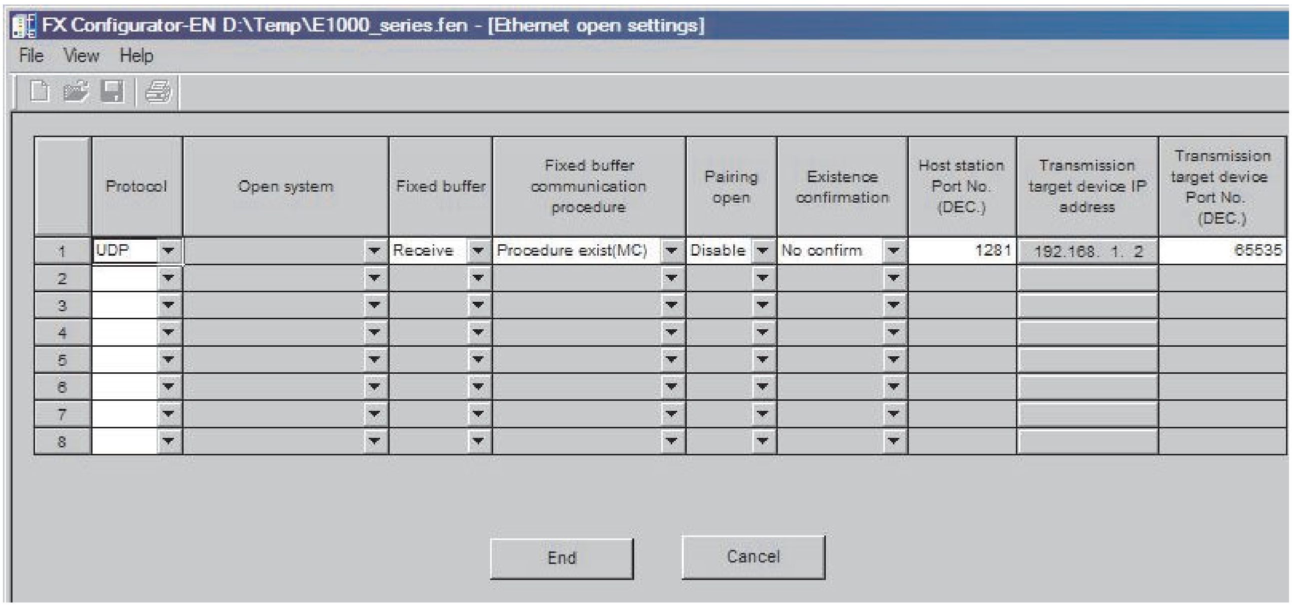
Subnet mask: 255.255.255.0

Gateway: 0.0.0.0

Primary DNS: 0.0.0.0

Secondary DNS: 0.0.0.0

When a HMI of the E1000 series is connected to the network, the following setting is required for the FX3U-ENET (refer to section 18.2, step ④):


FX Configurator-EN D:\Temp\E1000_series.fen - [Ethernet open settings]
File
View
Help
End
Cancel

	Protocol	Open system	Fixed buffer	Fixed buffer communication procedure	Pairing open	Existence confirmation	Host station Port No. (DEC.)	Transmission target device IP address	Transmission target device Port No. (DEC.)
1	UDP		Receive	Procedure exist(MC)	Disable	No confirm	1281	192.168. 1. 2	65535
2									
3									
4									
5									
6									
7									
8									

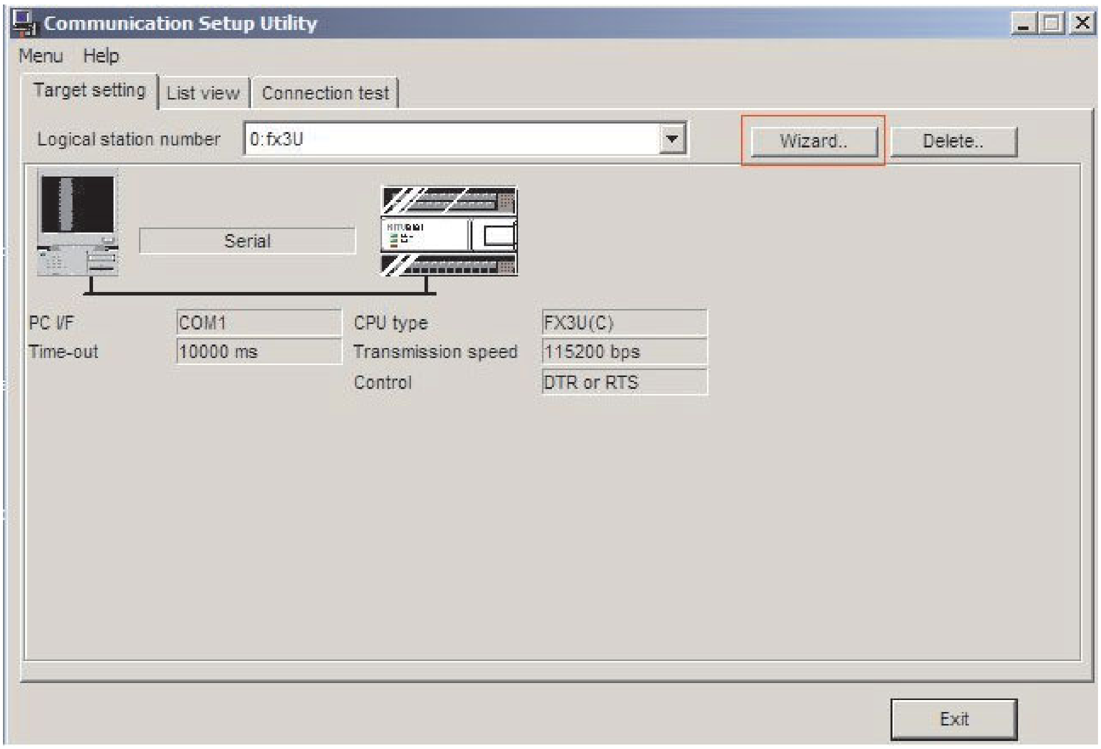
18.6 Communication via MX Component

MX Component is a tool designed to implement communication from a Personal Computer to the PLC without any knowledge of communication protocols and modules. MX Component is a powerful, user-friendly tools that make it very easy to connect your Mitsubishi PLC with the PC world.

MX Component supports serial CPU port connection, serial computer links (RS232C, RS422) and networks (Ethernet, CC-Link, MELSEC).

The following figures show the easy way for creating of communication between a PC and a PLC via MX Component.

① Start the **Communication Setting Utility** and select the **Wizard**.



Logical station number: 0:fx3U

PC I/F: COM1

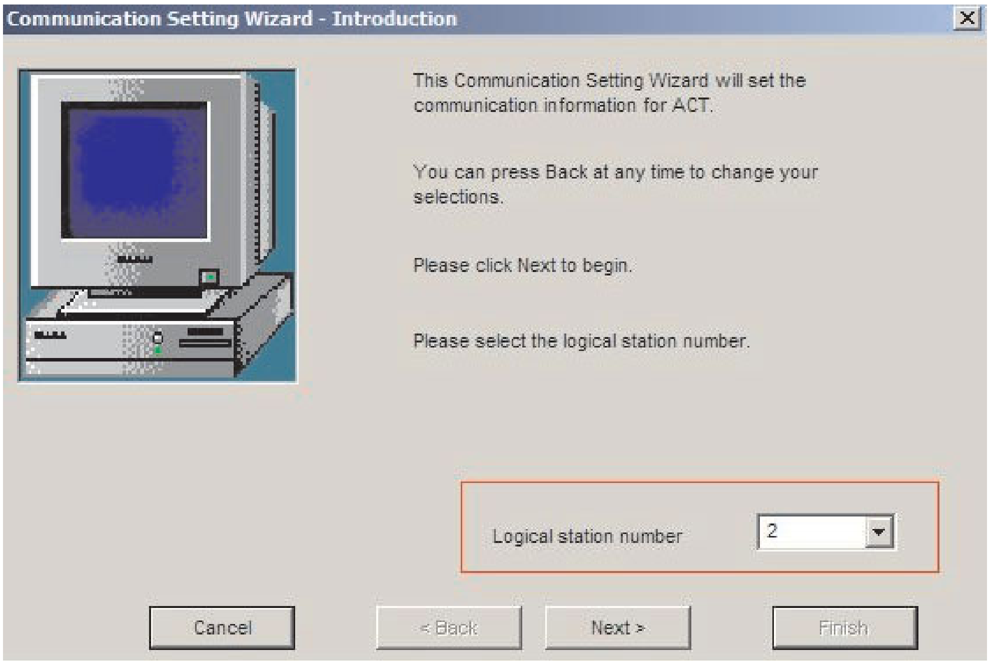
Time-out: 10000 ms

CPU type: FX3U(C)

Transmission speed: 115200 bps

Control: DTR or RTS

② First you must define the **Logical station number**.



Communication Setting Wizard - Introduction

This Communication Setting Wizard will set the communication information for ACT.

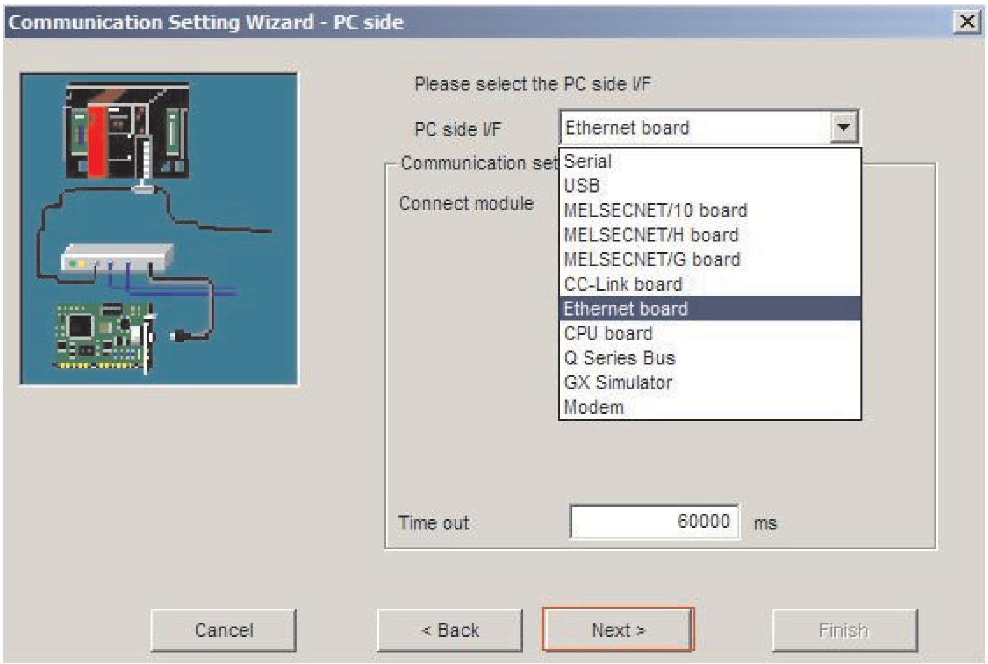
You can press Back at any time to change your selections.

Please click Next to begin.

Please select the logical station number.

Logical station number: 2

③ Next, configure the **Communication Settings** on the PC side. (Select the **Ethernet board**.)



Communication Setting Wizard - PC side

Please select the PC side I/F

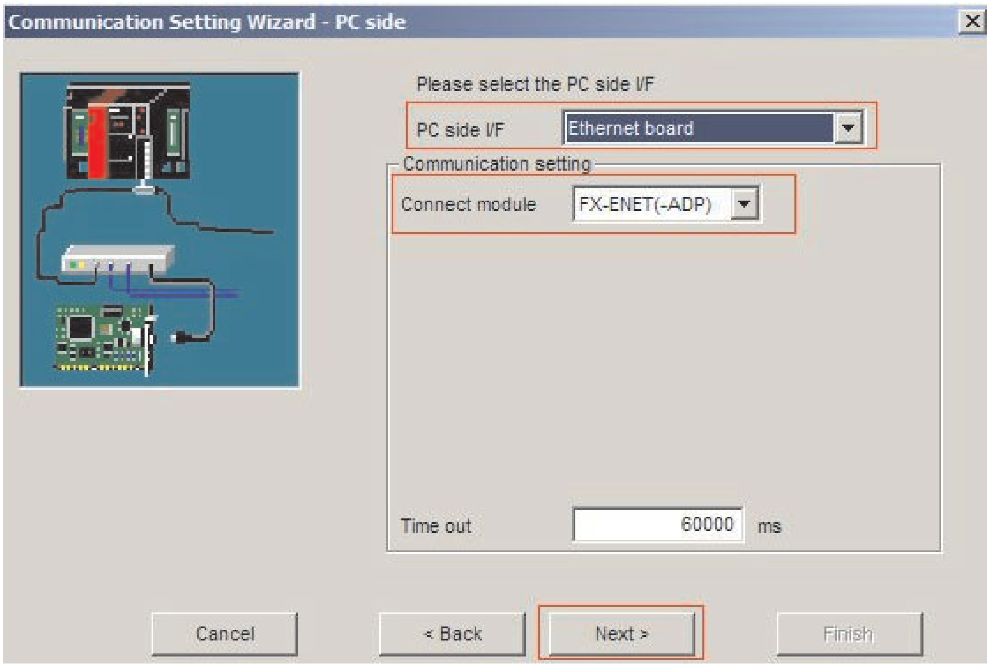
PC side I/F: Ethernet board

Communication set

Connect module

Time out: 60000 ms

④ Select the FX-ENET(-ADP).



Communication Setting Wizard - PC side

Please select the PC side I/F

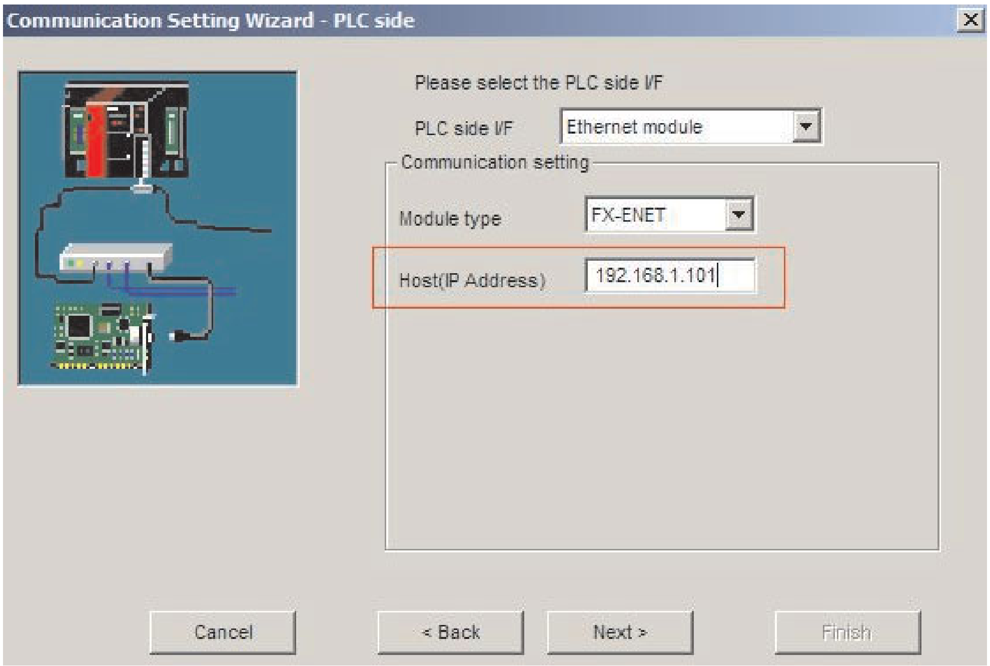
PC side I/F: Ethernet board

Communication setting

Connect module: FX-ENET(-ADP)

Time out: 60000 ms

⑤ Enter the IP address of the Ethernet interface module.



Communication Setting Wizard - PLC side

Please select the PLC side I/F

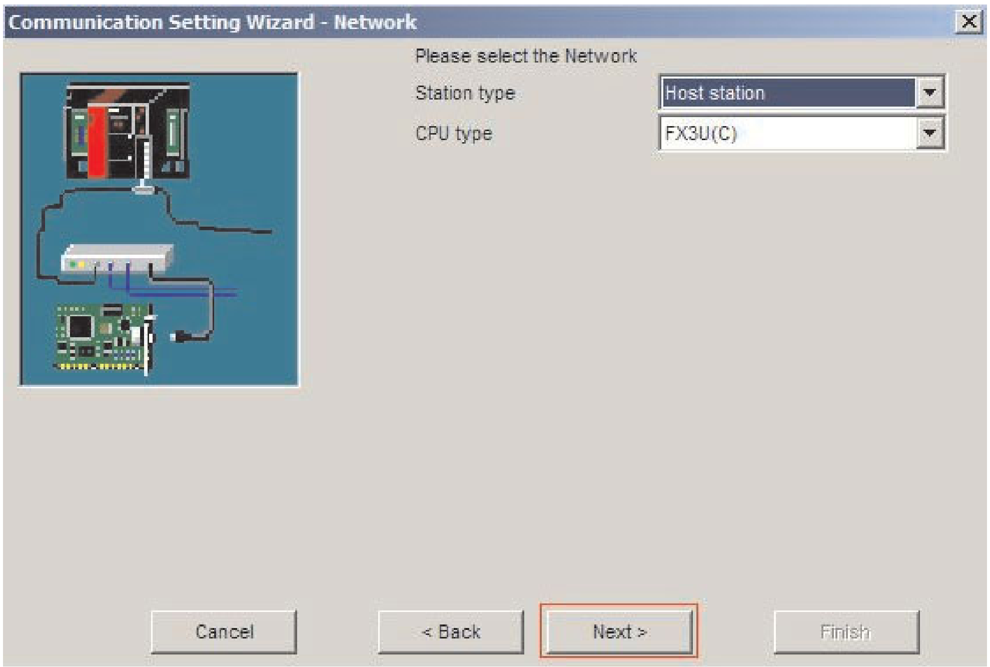
PLC side I/F: Ethernet module

Communication setting

Module type: FX-ENET

Host(IP Address): 192.168.1.101

⑥ Select the correct CPU type.



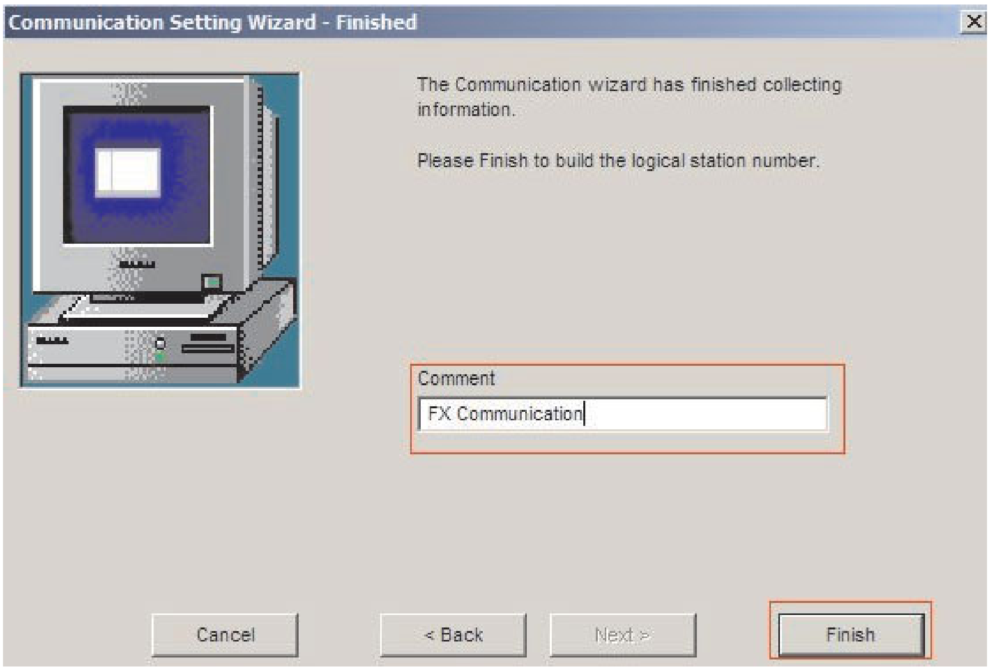
Communication Setting Wizard - Network

Please select the Network

Station type: Host station

CPU type: FX3U(C)

⑦ For the conclusion of the configuration define a name and press the ***Finish*** button.



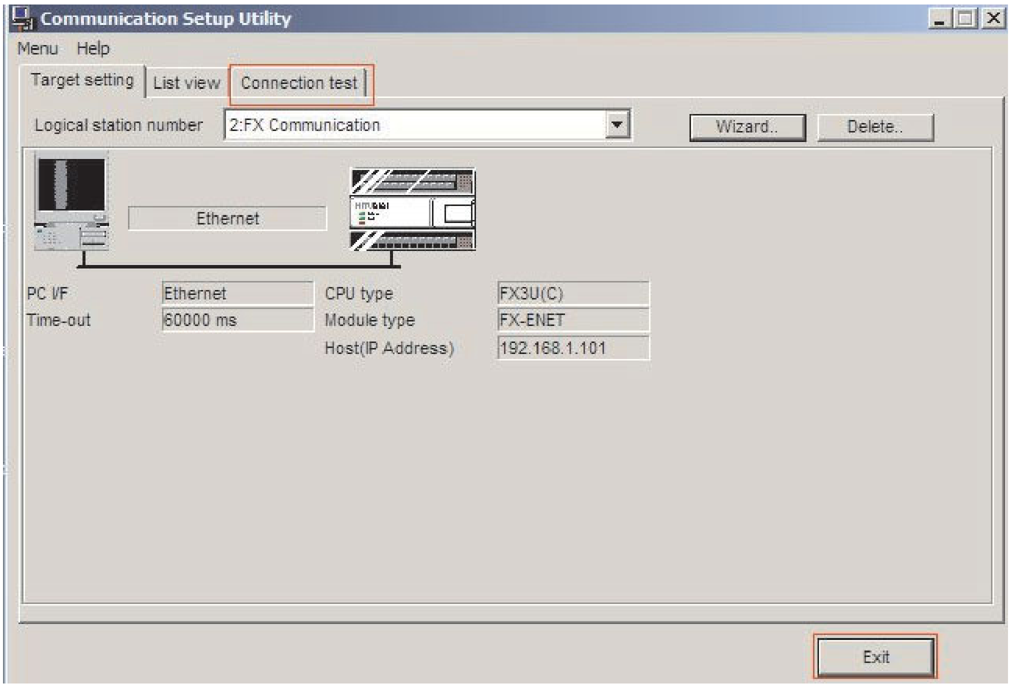
Communication Setting Wizard - Finished

The Communication wizard has finished collecting information.

Please Finish to build the logical station number.

Comment: FX Communication

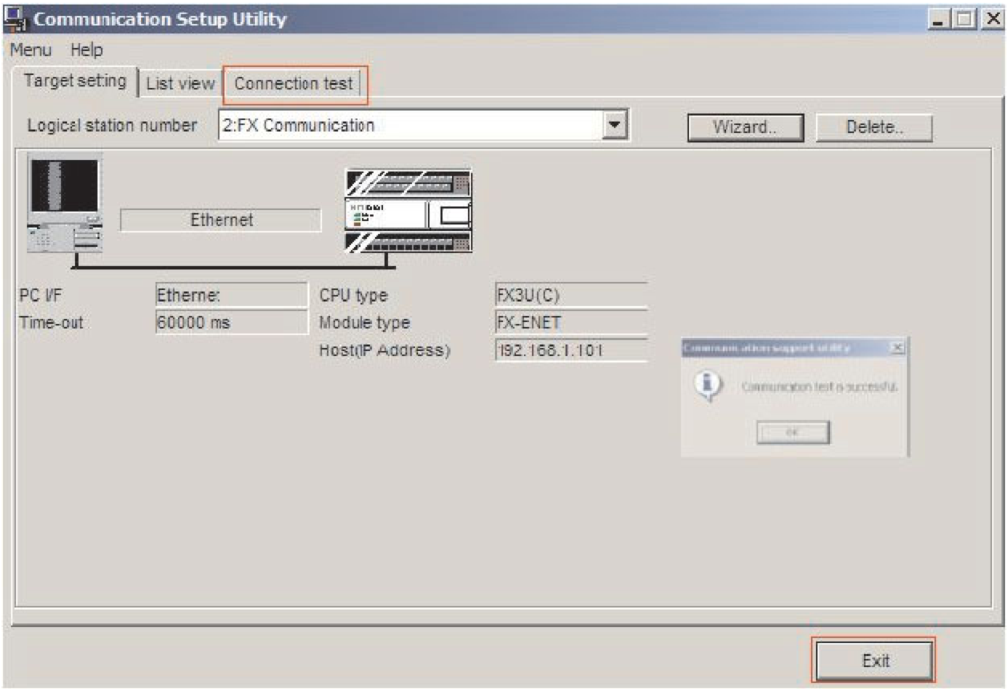
Now the definition of communication is finished. Under the folder **Connection test** the connection can be examined.



Logical station number: 2:FX Communication

PC I/F: Ethernet
Time-out: 60000 ms
CPU type: FX3U(C)
Module type: FX-ENET
Host(IP Address): 192.168.1.101

The message **Communication test is successful** indicates that your configuration is correct.



Logical station number: 2:FX Communication

PC I/F: Etherne:
Time-out: 60000 ms
CPU type: FX3U(C)
Module type: FX-ENET
Host(IP Address): 192.168.1.101

Communication test is successful.

After configuring the communication paths you can access all controller devices (read/write) with Microsoft programming languages like MS Visual Basic, MS C++ etc.

A Appendix

A.1 Special Relays

In addition to the relays that you can switch on and off with the PLC program there is also another class of relays known as special or diagnostic relays. These relays use the address range starting with M8000. Some contain information on system status and others can be used to influence program execution. Special relays cannot be used like other internal relays in a sequence program. However, some of them can be set ON or OFF in order to control the CPU. Represented here are some of the most commonly used devices.

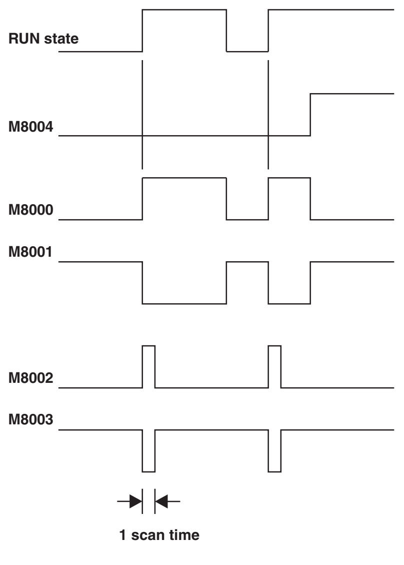
Special relays can be divided in two groups:

- Special relays whose signal state can only be read by the program (for instance using a LD or LDI instruction).
- Special relays whose signal state can be read and written (set or reset) by the program.

The following tables feature a "Read" and a "Write" column. If the symbol "●" is shown in one of these columns, the corresponding action is possible. The symbol "—" means that the corresponding action is not allowed.

There are also special registers for word information in a FX CPU. They are descriped in the next section.

A.1.1 PLC Status Diagnostic Information (M8000 to M8009)

Special Relay	Read	Write	CPU	Function	
M8000	●	—	FX1S FX1N FX2N FX2NC FX3G FX3U FX3UC	RUN monitor (NO contact)	 RUN state M8004 M8000 M8001 M8002 M8003 1 scan time
M8001	●	—		RUN monitor (NC contact)	
M8002	●	—		Initial pulse (NO contact)	
M8003	●	—		Initial pulse (NC contact)	
M8004	●	—		Error occurrence	
M8005	●	—	FX2N FX2NC FX3G FX3U FX3UC	Battery voltage low (ON when battery voltage is below the value set in D8006)	
M8006	●	—		Battery error latch (M8006 is set when battery voltage low is detected)	
M8007	●	—	FX2N FX2NC FX3U FX3UC	Momentary power failure	
M8008	●	—		Power failure detected	
M8009	●	—	FX2N FX2NC FX3G FX3U FX3UC	24V DC down (service power supply)	

A.1.2 Clock Devices and Real Time Clock (M8011 to M8019)

Special Relay	Read	Write	CPU	Function
M8010	—	—	—	Not used
M8011	●	—	FX1S FX1N FX2N FX2NC FX3G FX3U FX3UC	10 ms clock pulse ON and OFF in 10 ms cycle (ON: 5 ms, OFF: 5 ms)
M8012	●	—		100 ms clock pulse ON and OFF in 100 ms cycle (ON: 50 ms, OFF: 50 ms)
M8013	●	—		1 s clock pulse ON and OFF in 1 s cycle (ON: 500 ms, OFF: 500 ms)
M8014	●	—		1 min clock pulse ON and OFF in 1 min cycle (ON: 30 s, OFF: 30 s)
M8015	●	●		Clock stop and preset (For real time clock)
M8016	●	—		Time read display is stopped (For real time clock) The contents of D8013 to D8019 is frozen, but the clock is still running.
M8017	●	●		±30 seconds correction (For real time clock)
M8018	●	—		Real time clock installation detection (Always ON) For an FX2NC a memory card with integrated RTC must be installed.
M8019	●	—		Real time clock (RTC) setting error

A.1.3 PLC Operation Mode (M8030 to M8039)

Special relay	Read	Write	CPU	Function	
M8030	●	—	FX2N FX2NC FX3G FX3U FX3UC	Battery LED OFF When M8030 set to ON, LED on PLC is not lit even if battery voltage low is detected.	
M8031	●	●	FX1S FX1N FX2N FX2NC, FX3G FX3U FX3UC	Non-latch memory all clear	If this special auxiliary relays are activated, the ON/OFF image memory of Y, M, S, T, and C, and present values of T, C, D, special data registers and R are cleared to zero. However, file registers (D) in program memory, and extension file registers (ER) in the memory cassette are not cleared.
M8032	●	●		Latch memory all clear	
M8033	●	●		Memory hold STOP When PLC is switched from RUN to STOP, image memory and data memory are retained.	
M8034	●	●		All outputs disable All external output contacts of the PLC are turned OFF. The program however is still executed.	
M8035	●	●		Forced RUN mode	
M8036	●	●		Forced RUN signal	
M8037	●	●		Forced STOP signal	
M8038	—	●	FX1S FX1N FX2N (V2.0 or later) FX2NC FX3G FX3U FX3UC	Communication parameter setting flag (for N:N network setting)	
M8039	●	●	FX1S FX1N FX2N FX2NC FX3G, FX3U FX3UC	Constant scan mode When M8039 is ON, PLC waits until scan time specified in D8039 and then executes cyclic operation.	

A.1.4 Error Detection (M8060 to M8069)

Special relay	Read	Write	CPU	Function
M8060	●	—	FX2N FX2NC FX3G FX3U FX3UC	I/O configuration error
M8061	●	—	FX1S FX1N FX2N FX2NC FX3G FX3U FX3UC	PLC hardware error
M8062	●	—	FX2N FX2NC	PLC/Programming device communication error
			FX3G	Serial communication error [ch0]
M8063 [1]	●	—	FX1S FX1N FX2N FX2NC FX3G FX3U FX3UC	Serial communication error 1 [ch1]
M8064	●	—		Parameter error
M8065	●	—		Syntax error
M8066	●	—		Ladder error
M8067 [2]	●	—		Operation error
M8068	—	●		Operation error latch
M8069	—	●	FX2N FX2NC FX3G FX3U FX3UC	I/O bus check [3]

[1] The operation varies according to a PLC: Cleared in an FX1S, FX1N, FX2N, FX1NC, or FX2NC when PLC switches from STOP to RUN. Cleared in an FX3G, FX3U, and FX3UC PLC when the power supply is switched on Serial communication error 2 [ch2] in FX3G, FX3U, and FX3UC PLCs is detected by M8438.

[2] Cleared when PLC switches from STOP to RUN.

[3] When M8069 is ON, I/O bus check is executed.If an error is detected, the error code 6130 is written to special register D8069 and the special relay M8061 is set.

A.1.5 Extension Boards (Dedicated to FX1S and FX1N)

Special relay	Read	Write	CPU	Function
M8112	●	●	FX1S FX1N	Extension board FX1N-4EX-BD: Input BX0
				Extension board FX1N-2AD-BD: ch1 input mode change
				Extension board FX1N-1DA-BD: output mode change
M8113	●	●		Extension board FX1N-4EX-BD: Input BX1
				Extension board FX1N-2AD-BD: ch2 input mode change
M8114	●	●		Extension board FX1N-4EX-BD: Input BX2
M8115	●	●		Extension board FX1N-4EX-BD: Input BX3
M8116	●	●		Extension board FX1N-2EYT-BD: Output BY0
M8117	●	●		Extension board FX1N-2EYT-BD: Output BY1

A.1.6 Analog Special Adapter and Adapter Boards for FX3G

Special relay	Read	Write	CPU	Function
M8260 to M8269	●	—	FX3U FX3UC[4]	Special relays for the 1st analog special adapter [1]
	●	—	FX3G[5]	Special relays for the 1st analog adapter board [2]
M8270 to M8279	●	—	FX3U FX3UC[4]	Special relays for the 2nd analog special adapter [1]
	●	—	FX3G[5]	Special relays for the 2nd analog adapter board [3]
M8280 to M8289	●	—	FX3U FX3UC[4]	Special relays for the 3rd analog special adapter [1]
	●	—	FX3G	Special relays for the 1st analog special adapter
M8290 to M8299	●	—	FX3U, FX3UC[4]	Special relays for the 4th analog special adapter [1]
	●	—	FX3G	Special relays for the 2nd analog special adapter (FX3G-40M□/□ and FX3G-60M□/□) only

[1] The unit number of the analog special adapter is counted from the base units side.

[2] Mounted to the expansion board connector of the base units FX3G-14M□/□ or FX3G-24M□/□ or to the left expansion board connector (BD1) of the base units FX3G-40□/□ or FX3G-60M□/□.

[3] Mounted to the right expansion board connector (BD2) of the base units FX3G-40□/□ or FX3G-60M□/□.

[4] Available in Version 2.00 or later

[5] Available in Version 1.10 or later

A.2 Special Registers

Just like the special relays (section A.1) starting at address M8000 the FX controllers also have special or diagnostic registers, whose addresses start at D8000. Often there is also a direct connection between the special relays and special registers. For example, special relay M8005 shows that the voltage of the PLC’s battery is too low, and the corresponding voltage value is stored in special register D8005. The following tables shows a small selection of the available special registers as examples.

Special registers can be divided in two groups:

– Special registers whose value can only be read by the program

– Special relays whose value can be read and written by the program.

The following tables feature a "Read" and a "Write" column. If the symbol "●" is shown in a one of these columns, the corresponding action is possible. The symbol "—" means that the corresponding action is not allowed.

A.2.1 PLC Status Diagnostic Information (D8000 to D8009)

Special Register	Read	Write	CPU	Function
D8000	●	●	FX1S FX1N FX2N FX2NC FX3G FX3U FX3UC	Watchdog timer setting (in 1ms steps). (Writes from system ROM at power ON) Value overwritten by program is valid after END or WDT instruction execution. The setting must be larger than the maximum scan time (stored in D8012). Default value is 200 ms.
D8001	●	—		PLC type and system version FX1S: 22V$_{VV}$ FX1N/FX3G: 26V$_{VV}$ FX2N/FX2NC/FX3U/FX3UC: 24V$_{VV}$ (e. g. FX1N Version 1.00 → 26100)
D8002	●	—		Memory capacity 0002 → 2k steps (FX1S only) 0004 → 4k steps (FX2N/FX2NC only) 0008 → 8k steps or more (not for FX1S) If 16K steps or more "K8" is written to D8002 and "16", "32" or "64" is written to D8102.
D8003	●	—		Memory typ: 00H→ RAM (Memory cassette) 01H→ EPROM (Memory cassette) 02H→ EEPROM (Memory cassette or flash memory) 0AH→ EEPROM (Memory cassette or flash memory, write-protected) 10H→ Built-in memory in PLC
D8004	●	—		Error number (M) If D8004 contains e.g. the value 8060, special relay M8060 is set.
D8005	—	—	FX2N FX2NC FX3G FX3U FX3UC	Battery voltage (Example: "36" -> 3.6 V)
D8006	—	—		Low battery voltage detection level. Default settings: FX2N/FX2NC: 3.0 V ("30") FX3G/FX3U/FX3UC: 2.7 V ("27")
D8007	—	—	FX2N FX2NC FX3U FX3UC	Momentary power failure count Operation frequency of M8007 is stored. Cleared at power-off.
D8008	—	—	FX2N FX2NC FX3U FX3UC	Power failure detection Default settings: FX2N/FX3U: 10 ms (AC power supply) FX2NC/FX3UC: 5 ms (DC power supply)
D8009	—	—	FX2N FX2NC FX3G FX3U FX3UC	24V DC failed device Minimum input device number of extension units and extension power units in which 24V DC has failed.

A.2.2 Scan Information and Real Time Clock (D8010 to D8019)

Special Register	Read	Write	CPU	Function
D8010	●	—	FX1S FX1N FX2N FX2NC FX3G FX3U FX3UC	Present scan time (in units of 0.1 ms)
D8011	●	—		Minimum value of scan time (in units of 0.1 ms)
D8012	●	—		Maximum value of scan time (in units of 0.1 ms)
D8013	●	●	FX1S FX1N FX2N FX2NC* FX3G FX3U FX3UC	Real time clock: Seconds (0 to 59)
D8014	●	●		Real time clock: Minutes (0 to 59)
D8015	●	●		Real time clock: Hours (0 to 23)
D8016	●	●		Real time clock: Date (Day, 1 to 31)
D8017	●	●		Real time clock: Date (Month, 1 to 12)
D8018	●	●		Real time clock: Date (Year, 0 to 99)
D8019	●	●		Real time clock: Day of the week (0 (Sunday) to 6 (Saturday))

* For an FX2NC a memory card with integrated RTC must be installed.

A.2.3 PLC Operation Mode (D8030 to D8039)

Special Register	Read	Write	CPU	Function
D8030	●	—	FX1S FX1N FX3G	Value of analog volume VR1 (Integer from 0 to 255)
D8031	●	—		Value of analog volume VR2 (Integer from 0 to 255)
D8032 – D8038	—	—	—	Not used
D8039	—	●	FX1S FX1N FX2N FX2NC FX3G FX3U FX3UC	Constant scan duration Default: 0 ms (in 1 ms steps) (Writes from system ROM at power ON) Can be overwritten by program

A.2.4 Error Codes (D8060 to D8069)

Special Register	Read	Write	CPU	Function
D8060	●	—	FX2N FX2NC FX3G FX3U FX3UC	If the unit or block corresponding to a programmed I/O number is not actually loaded, M8060 is set to ON and the first device number of the erroneous block is written to D8060 Meaning of the four digit code: 1st digit: 0 = Output, 1 = Input 2nd to 4th digit: First device number of the erroneous block
D8061	●	—	FX1S FX1N FX2N FX2NC FX3G FX3U FX3UC	Error code for PLC hardware error
D8062	●	—	FX2N, FX2NC FX3G FX3U FX3UC	Error code for PLC/PP communication error
			FX3G	Error code for serial communication error [ch0]
D8063	●	—	FX1S FX1N FX2N FX2NC FX3G FX3U FX3UC	Error code for serial communication error 1 [ch1]
D8064	●	—		Error code for parameter error
D8065	●	—		Error code for syntax error
D8066	●	—		Error code for ladder error
D8067	●	—		Error code for operation error
D8068*	—	●		Operation error step number latched In case of 32K steps or more, the step number is stored in [D8313, D8312].
D8069*	●	—		Error step number of M8065 to M8067 In case of 32K steps or more, the step number is stored in [D8315, D8314].

* Cleared when PLC switches from STOP to RUN.

A.2.5 Extension Boards (Dedicated to FX1S and FX1N)

Special Register	Read	Write	CPU	Function
D8112	●	—	FX1S FX1N	Adapter FX1N-2AD-BD: Digital input value ch.1
D8113	●	—		Adapter FX1N-2AD-BD: Digital input value ch.2
D8114	●	●		Adapter FX1N-1DA-BD: Digital output value ch.1

A.2.6 Analog Special Adapter and Adapter Boards for FX3G

Special Register	Read	Write	CPU	Function
D8260 to D8269	●	—	FX3U FX3UC[4]	Special registers for the 1st analog special adapter [1]
	●	—	FX3G[5]	Special registers for the 1st analog adapter board [2]
D8270 to D8279	●	—	FX3U FX3UC[4]	Special registers for the 2nd analog special adapter [1]
	●	—	FX3G[5]	Special registers for the 2nd analog adapter board [3]
D8280 to D8289	●	—	FX3U FX3UC[4]	Special registers for the 3rd analog special adapter [1]
	●	—	FX3G	Special registers for the 1st analog special adapter
D8290 to D8299	●	—	FX3U, FX3UC[4]	Special registers for the 4th analog special adapter [1]
	●	—	FX3G	Special registers for the 2nd analog special adapter (FX3G-40M□/□ and FX3G-60M□/□) only

[1] The unit number of the analog special adapter is counted from the base units side.

[2] Mounted to the expansion board connector of the base units FX3G-14M□/□ or FX3G-24M□/□ or to the left expansion board connector (BD1) of the base units FX3G-40□/□ or FX3G-60M□/□.

[3] Mounted to the right expansion board connector (BD2) of the base units FX3G-40□/□ or FX3G-60M□/□.

[4] Available in Version 2.00 or later

[5] Available in Version 1.10 or later

A.3 Error Code List

When an error has been detected in the PLC, the error code is stored in special registers D8060 to D8067 and D8438. The following actions should be followed for diagnostic errors.

Represented here are some of the most common error codes.

A.3.1 Error codes 6101 to 6409

Error	Special Register	Error Code	Description	Corrective Action
PLC hardware error	D8061	0000	No error	—
		6101	RAM error	
		6102	Operation circuit error	
		6103	I/O bus error (M8069 = ON)	Check for the correct connection of extension cables.
		6104	Powered extension unit 24 V failure (M8069 = ON)	
		6105	Watchdog timer error	Check user program. The scan time exceeds the value stored in D8000.
		6106	I/O table creation error (CPU error) When turning the power ON to the baseunit, a 24V power failure occurs in a powered extension unit. (The error occurs if the 24V power is not supplied for 10 seconds or more after main power turn ON.)	Check the power supply for the powered extension units.
		6107	System configuration error	Check the number of the connected special function units/blocks. A few special function units/blocks are limited the number to connect.
Communication error between PLC and programming device (FX2N and FX2NC only)	D8062	0000	No error	—
		6201	Parity, overrun or framing error	Check the cable connection between the programming device and the PLC. This error may occur when a cable is disconnected and reconnected during PLC monitoring.
		6202	Communication character error	
		6203	Communication data sum check error	
		6204	Data format error	
		6205	Command error	
Serial communication error	D8063	0000	No error	—
		6301	Parity, overrun or framing error	● Inverter communication, computer link and programming: Ensure the communication parameters are correctly set according to their applications. ● N:N network, parallel link, etc.: Check programs according to applications. ● Remote maintenance: Ensure modem power is ON and check the settings of the AT commands. ● Wiring: Check the communication cables for correct wiring.
		6302	Communication character error	
		6303	Communication data sum check error	
		6304	Communication data format error	
		6305	Command error	
		6306	Communication time-out detected	
		6307	Modem initialization error	
		6308	N:N network parameter error	
		6312	Parallel link character error	
		6313	Parallel link sum error	
		6314	Parallel link format error	
		6320	Inverter communication error	

Error	Special Register	Error Code	Description	Corrective Action
Parameter error	D8064	0000	No error	—
		6401	Program sum check error	STOP the PLC, and correctly set the parameters.
		6402	Memory capacity setting error	
		6403	Latched device area setting error	
		6404	Comment area setting error	
		6405	File register area setting error	
		6406	Special unit (BFM) initial value setting, positioning instruction setting sum check error	
		6407	Special unit (BFM) initial value setting, positioning instruction setting error	
		6409	Other setting error	

A.3.2 Error codes 6501 to 6510

Error	Special Register	Error Code	Description	Corrective Action
Syntax error	D8065	0000	Kein Fehler	During programming, each instruction is checked. If a syntax error is detected, modify the instruction correctly.
		6501	Incorrect combination of instruction, device symbol and device number	
		6502	No OUT T or OUT C before setting value	
		6503	– No OUT T or OUT C before setting value – Insufficient number of operands for an applied instruction	
		6504	– Same label number is used more than once. – Same interrupt input or high speed counter input is used more than once.	
		6505	Device number is out of allowable range.	
		6506	Invalid instruction	
		6507	Invalid label number [P]	
		6508	Invalid interrupt input [I]	
		6509	Other error	
		6510	MC nesting number error	

A.3.3 Error codes 6610 to 6632

Error	Special Register	Error Code	Description	Corrective Action
Circuit error	D8066	0000	No error	—
		6610	LD, LDI is continuously used 9 times or more.	This error occurs when a combination of instructions is incorrect in the entire circuit block or when the relationship between a pair of instructions is incorrect. Modify the instructions in the program mode so that their mutual relationship becomes correct.
		6611	More ANB/ORB instructions than LD/LDI instructions	
		6612	Less ANB/ORB instructions than LD/LDI instructions	
		6613	MPS is continuously used 12 times or more.	
		6614	No MPS instruction	
		6615	No MPP instruction	
		6616	No coil between MPS, MRD and MPP, or incorrect combination	
		6617	Instruction below is not connected to bus line: STL, RET, MCR, P, I, DI, EI, FOR, NEXT, SRET, IRET, FEND or END	
		6618	STL, MC or MCR can be used only in main program, but it is used elsewhere (e.g. in interrupt routine or subroutine).	
		6619	Invalid instruction is used in FOR-NEXT loop: STL, RET, MC, MCR, I (interrupt pointer) or IRET.	
		6620	FOR-NEXT instruction nesting level exceeded	
		6621	Numbers of FOR and NEXT instructions do not match.	
		6622	No NEXT instruction	
		6623	No MC instruction	
		6624	No MCR instruction	
		6625	STL instruction is continuously used 9 times or more.	
		6626	Invalid instruction is programmed within STL-RET loop: MC, MCR, I (interrupt pointer), SRET or IRET.	
		6627	No RET instruction	
		6628	Invalid instruction is used in main program: I (interrupt pointer), SRET or IRET	
		6629	No P or I (interrupt pointer)	
		6630	No SRET or IRET instruction	
		6631	SRET programmed in invalid location	
		6632	FEND programmed in invalid location	

A.3.4 Error codes 6701 to 6710

Error	Special Register	Error Code	Description	Corrective Action
Operation error	D8067	0000	No error	—
		6701	– No jump destination (pointer) for CJ or CALL instruction – Label is undefined or out of P0 to P4095 due to indexing – Label P63 is executed in CALL instruction; cannot be used in CALL instruction as P63 is for jumping to END instruction.	This error occurs in the execution of operation. Review the program, or check the contents of the operands used in the applied instructions.*
		6702	CALL instruction nesting level is 6 or more	
		6703	Interrupt nesting level is 3 or more	
		6704	FOR-NEXT instruction nesting level is 6 or more.	
		6705	Operand of applied instruction is inapplicable device.	
		6706	Device number range or data value for operand of applied instruction exceeds limit.	
		6707	File register is accessed without parameter setting of file register.	
		6708	FROM/TO instruction error	This error occurs in the execution of operation. Review the program, or check the contents of the operands used in the applied instructions. Check whether the specified buffer memories exist in the equipment. Check whether the extension cables are correctly connected.
		6709	Other (e.g. improper branching)	This error occurs in the execution of operation. Review the program, or check the contents of the operands used in the applied instructions.*
		6710	Mismatch among parameters	This error occurs when the same device is used within the source and destination in a shift instruction, etc.

* Even if the syntax or circuit design is correct, an operation error may still occur. For example: "T200Z" itself is not an error. But if Z had a value of 400, the timer T600 would be attempted to be accessed. This would cause an operation error since there is no T600 device available.

A.4 Number of Occupied Input/Output Points and Current Consumption

The following tables show how many input/output points are occupied in a base unit by a certain unit, along with the power supply type and current consumption values needed for selecting a product.

The current consumption is determined differently in the following cases.

5V DC and internal 24V DC are supplied to the products through an extension cable, and the current consumption must be calculated

Subtract the current consumption at the internal 24V DC as follows.

– For the AC power type base unit, subtract the current consumption at the internal 24V DC from the 24V DC service power supply.

– For the DC power type base unit, subtract the current consumption at the internal 24V DC from the power supply for the internal 24V DC.

– Some special function modules need "external 24 V DC". Include this current in the calculation of current consumption when the current is supplied by the 24V DC service power supply. When the current is supplied by an external power supply, the current is not included in the calculation of current consumption.

A.4.1 Interface Adapter Boards and Communication Adapter Boards

Type	Number of occupied I/O points	Current consumption [mA]		
		5 V DC	24 V DC (internal)	24 V DC (external)
FX1N-232-BD	—	20	—	—
FX2N-232-BD	—			
FX3G-232-BD	—	—		
FX3U-232-BD	—	20		
FX1N-422-BD	—	60*	—	—
FX2N-422-BD	—			
FX3G-422-BD	—	—		
FX3U-422-BD	—	20*		
FX1N-485-BD	—	60	—	—
FX2N-485-BD	—			
FX3G-485-BD	—	—		
FX3U-485-BD	—	40		
FX3U-USB-BD	—	15	—	—
FX1N-CNV-BD	—	—	—	—
FX2N-CNV-BD				
FX3G-CNV-BD				
FX3U-CNV-BD				
FX3G-2AD-BD	—	—	—	—
FX3G-1DA-BD				
FX3G-8AV-BD				

* When a programming tool or GOT is connected, add the current consumed by this unit (see next page)

Programming Tool, Interface Converter, Display Module and GOT

Type	Number of occupied I/O points	Current consumption [mA]		
		5 V DC	24 V DC (internal)	24 V DC (external)
FX-20P(-E)	—	150	—	—
FX-232AWC-H	—	120	—	—
FX-USB-AW	—	15	—	—
FX3U-7DM		20		
FX10DM-E	—	220	—	—
F920GOT-BBD5-K-E	—	220	—	—

A.4.2 Special Adapters

Type	Number of occupied I/O points	Current consumption [mA]			
		5 V DC	24 V DC (internal)	24 V DC (external)	At start up
FX3U-4HSX-ADP	—	30	30	0	30*
FX3U-2HSY-ADP	—	30	60	0	120*
FX3U-4AD-ADP	—	15	0	40	—
FX3U-4DA-ADP	—	15	0	150	—
FX3U-4AD-PNK-ADP	—	15	0	50	—
FX3U-4AD-PT-ADP	—	15	0	50	—
FX3U-4AD-PTW-ADP	—	15	0	50	—
FX3U-4AD-TC-ADP	—	15	0	45	—
FX3U-3A-ADP	—	20	0	90	—
FX2NC-232ADP	—	100	0	0	—
FX3U-232ADP	—	30	0	0	—
FX3U-485ADP	—	20	0	0	—

* The current consumption at start up must be considered when connected to a DC powered base unit.

A.4.3 Extension Blocks

Type	Number of occupied I/O points	Current consumption [mA]		
		5 V DC	24 V DC (internal)	24 V DC (external)
FX2N-8ER-ES/UL	16	–	125	0
FX2N-8EX-ES/UL	8	—	50	0
FX2N-16EX-ES/UL	16	—	100	0
FX2N-8EYR-ES/UL	8	—	75	0
FX2N-8EYT-ESS/UL	8	—	75	0
FX2N-16EYR-ES/UL	16	—	150	0
FX2N-16EYT-ESS/UL	16	—	150	0

A.4.4 Special Function Modules

Type	Number of occupied I/O points	Current consumption [mA]			
		5 V DC	24 V DC (internal)	24 V DC (external)	At start up
FX3U-2HC	8	245	0	0	—
FX3U-4AD	8	110	0	90	—
FX3U-4DA	8	120	0	160	—
FX3U-4LC	8	160	0	50	—
FX3U-20SSC-H	8	100	0	220	—
FX2N-2AD	8	20	50 [1]	0	170
FX2N-2DA	8	30	85 [1]	0	190
FX2N-4AD	8	30	0	55	—
FX2N-4DA	8	30	0	200	—
FX2N-4AD-TC	8	30	0	50	—
FX2N-4AD-PT	8	30	0	50	—
FX2N-8AD	8	50	0	80	—
FX2N-5A	8	70	0	90	—
FX2N-2LC	8	70	0	55	—
FX2N-1HC	8	90	0	0	—
FX2N-1PG-E	8	55	0	40	—
FX2N-10PG	8	120	0	70 [2]	—
FX2N-232IF	8	40	0	80	—
FX2N-16CCL-M	8 [3]	0	0	150	—
FX2N-32CCL-M	8	130	0	50	—
FX2N-32ASI-M	8 [4]	150	0	70	—
FX0N-3A	8	30	90 [1]	0	165
FX2N-10GM	8	—	—	5	—
FX2N-20GM	8	—	—	10	—

[1] When analog special function blocks (FX0N-3A, FX2N-2AD and FX2N-2DA) are connected to an input/ output powered extension unit (FX2N-32E□ or FX2N-48E□), the following limitation must be taken into consideration. (When the blocks are connected to the base unit, this limitation is not applied.)

The total current consumption of the analog special function blocks (FX0N-3A, FX2N-2AD and FX2N-2DA) should be less than the following current values.
- When connected to FX2N-32E□: 190 mA or less
- When connected to FX2N-48E□: 300 mA or less.

[2] When the voltage of the external DC power supply is 5 V DC, the current is 100 mA.

[3] A FX2N-16CCL-M cannot be used together with a FX2N-32ASI-M. The following number of points is added according to the products connected to the network: (Number of remote I/O stations) x 32 points.

[4] A FX2N-32ASI-M cannot be used together with a FX2N-16CCL-M. Only one unit can be added to the whole system. The following number of points is added according to the products connected to the network:
(Number of active slaves) x 8 points.

NOTE

When applying a DC power type base unit, calculate the current consumption at startup.

A.5 PLC Components Glossary

The following table describes the meaning and functionality of the single components und parts of a Mitsubishi PLC.

Component	Description
Connection for expansion adapter boards	Optional expansion adapter boards can be connected to this interface. A variety of different adapters are available for all FX lines (except the FX2NC). These adapters extend the capabilities of the controllers with additional functions or communications interfaces. The adapter boards are plugged directly into the slot.
Connection for programming units	This connection can be used for connecting the FX-20P-E hand-held programming unit or an external PC or notebook with a programming software package (e.g. GX Developer).
EEPROM	Read/write memory in which the PLC program can be stored and read with the programming software. This solid-state memory retains its contents without power, even in the event of a power failure, and does not need a battery.
Memory cassette slot	Slot for optional memory cassettes. Inserting a memory cassette disables the controller’s internal memory – the controller will then only execute the program stored in the cassette.
Extension bus	Both additional I/O expansion modules and special function modules that add additional capabilities to the PLC system can be connected here. See Chapter 6 for an overview of the available modules.
Analog potentiometers	The analog potentiometers are used for setting analog setpoint values. The setting can be polled by the PLC program and used for timers, pulse outputs and other functions.
Service power supply	The service power supply (not for FX2NC ans FX3UC) provides a regulated 24V DC power supply source for the input signals and the sensors. The capacity of this power supply depends on the controller model (e.g. FX1S, FX1N and FX3G: 400mA; FX2N-16M□-□□ through FX2N-32M□-□□: 250 mA, FX2N-48M□-□□ through FX2N-64M□-□□: 460 mA)
Digital inputs	The digital inputs are used for inputting control signals from the connected switches, buttons or sensors. These inputs can read the values ON (power signal on) and OFF (no power signal).
Digital outputs	You can connect a variety of different actuators and other devices to these outputs, depending on the nature of your application and the output type.
LEDs for indicating the input status	These LEDs show which inputs are currently connected to a power signal, i.e. a defined voltage. When a signal is applied to an input the corresponding LED lights up, indicating that the state of the input is ON.
LEDs for indicating the output status	These LEDs show the current ON/OFF states of the digital outputs. These outputs can switch a variety of different voltages and currents depending on the model and output type.
LEDs for indicating the operating status	The LEDs RUN, POWER and ERROR show the current status of the controller. POWER shows that the power is switched on, RUN lights up when the PLC program is being executed and ERROR lights up when an error or malfunction is registered.
Memory battery	The battery protects the contents of the MELSEC PLC’s volatile RAM memory in the event of a power failure (FX2N, FX2NC and FX3U only). It protects the latched ranges for timers, counters and relays. In addition to this it also provides power for the integrated real-time clock when the PLC’s power supply is switched off.
RUN/STOP switch	MELSEC PLCs have two operating modes, RUN and STOP. The RUN/STOP switch allows you to switch between these two modes manually. In RUN mode the PLC executes the program stored in its memory. In STOP mode program execution is stopped and it is possible to program the controller.

Index

MITSUBISHI ELECTRIC

HEADQUARTERS

MITSUBISHI ELECTRIC EUROPE B.V. **EUROPE**
German Branch
Gothaer Straße 8
D-40880 Ratingen
Phone: +49 (0)2102 / 486-0
Fax: +49 (0)2102 / 486-1120

MITSUBISHI ELECTRIC EUROPE B.V. **CZECH REPUBLIC**
Czech Branch
Avenir Business Park, Radlická 714/113a
CZ-158 00 Praha 5
Phone: +420 - 251 551 470
Fax: +420 - 251-551-471

MITSUBISHI ELECTRIC EUROPE B.V. **FRANCE**
French Branch
25, Boulevard des Bouvets
F-92741 Nanterre Cedex
Phone: +33 (0)1 / 55 68 55 68
Fax: +33 (0)1 / 55 68 57 57

MITSUBISHI ELECTRIC EUROPE B.V. **IRELAND**
Irish Branch
Westgate Business Park, Ballymount
IRL-Dublin 24
Phone: +353 (0)1 4198800
Fax: +353 (0)1 4198890

MITSUBISHI ELECTRIC EUROPE B.V. **ITALY**
Italian Branch
Viale Colleoni 7
I-20041 Agrate Brianza (MB)
Phone: +39 039 / 60 53 1
Fax: +39 039 / 60 53 312

MITSUBISHI ELECTRIC EUROPE B.V. **POLAND**
Poland Branch
Krakowska 50
PL-32-083 Balice
Phone: +48 (0)12 / 630 47 00
Fax: +48 (0)12 / 630 47 01

MITSUBISHI ELECTRIC EUROPE B.V. **SPAIN**
Spanish Branch
Carretera de Rubí 76-80
E-08190 Sant Cugat del Vallés (Barcelona)
Phone: 902 131121 // +34 935653131
Fax: +34 935891579

MITSUBISHI ELECTRIC EUROPE B.V. **UK**
UK Branch
Travellers Lane
UK-Hatfield, Herts. AL10 8XB
Phone: +44 (0)1707 / 27 61 00
Fax: +44 (0)1707 / 27 86 95

MITSUBISHI ELECTRIC CORPORATION **JAPAN**
Office Tower "Z" 14 F
8-12,1 chome, Harumi Chuo-Ku
Tokyo 104-6212
Phone: +81 3 622 160 60
Fax: +81 3 622 160 75

MITSUBISHI ELECTRIC AUTOMATION, Inc. **USA**
500 Corporate Woods Parkway
Vernon Hills, IL 60061
Phone: +1 847 478 21 00
Fax: +1 847 478 22 53

EUROPEAN REPRESENTATIVES

GEVA **AUSTRIA**
Wiener Straße 89
AT-2500 Baden
Phone: +43 (0)2252 / 85 55 20
Fax: +43 (0)2252 / 488 60

TEHNIKON **BELARUS**
Oktyabrskaya 16/5, Off. 703-711
BY-220030 Minsk
Phone: +375 (0)17 / 210 46 26
Fax: +375 (0)17 / 210 46 26

ESCO DRIVES & AUTOMATION **BELGIUM**
Culliganlaan 3
BE-1831 Diegem
Phone: +32 (0)2 / 717 64 30
Fax: +32 (0)2 / 717 64 31

Koning & Hartman b.v. **BELGIUM**
Woluwelaan 31
BE-1800 Vilvoorde
Phone: +32 (0)2 / 257 02 40
Fax: +32 (0)2 / 257 02 49

INEA BH d.o.o. **BOSNIA AND HERZEGOVINA**
Aleja Lipa 56
BA-71000 Sarajevo
Phone: +387 (0)33 / 921 164
Fax: +387 (0)33/ 524 539

AKHNATON **BULGARIA**
4 Andrej Ljapchev Blvd. Pb 21
BG-1756 Sofia
Phone: +359 (0)2 / 817 6004
Fax: +359 (0)2 / 97 44 06 1

INEA CR d.o.o. **CROATIA**
Losinjska 4 a
HR-10000 Zagreb
Phone: +385 (0)1 / 36 940 - 01/ -02/ -03
Fax: +385 (0)1 / 36 940 - 03

AutoCont C.S. s.r.o. **CZECH REPUBLIC**
Technologická 374/6
CZ-708 00 Ostrava-Pustkovec
Phone: +420 595 691 150
Fax: +420 595 691 199

B:ELECTRIC, s.r.o. **CZECH REPUBLIC**
Mladoboleslavská 812
CZ-197 00 Praha 19 - Kbely
Phone: +420 286 850 848, +420 724 317 975
Fax: +420 286 850 850

Beijer Electronics A/S **DENMARK**
Lykkegårdsvej 17, 1.
DK-4000 Roskilde
Phone: +45 (0)46/ 75 76 66
Fax: +45 (0)46 / 75 56 26

Beijer Electronics Eesti OÜ **ESTONIA**
Pärnu mnt.160i
EE-11317 Tallinn
Phone: +372 (0)6 / 51 81 40
Fax: +372 (0)6 / 51 81 49

Beijer Electronics OY **FINLAND**
Jaakonkatu 2
FIN-01620 Vantaa
Phone: +358 (0)207 / 463 500
Fax: +358 (0)207 / 463 501

UTECO A.B.E.E. **GREECE**
5, Mavrogenous Str.
GR-18542 Piraeus
Phone: +30 211 / 1206 900
Fax: +30 211 / 1206 999

MELTRADE Ltd. **HUNGARY**
Fertő utca 14.
HU-1107 Budapest
Phone: +36 (0)1 / 431-9726
Fax: +36 (0)1 / 431-9727

Beijer Electronics SIA **LATVIA**
Vestienas iela 2
LV-1035 Riga
Phone: +371 (0)784 / 2280
Fax: +371 (0)784 / 2281

Beijer Electronics UAB **LITHUANIA**
Savanoriu Pr. 187
LT-02300 Vilnius
Phone: +370 (0)5 / 232 3101
Fax: +370 (0)5 / 232 2980

EUROPEAN REPRESENTATIVES

ALFATRADE Ltd. **MALTA**
99, Paola Hill
Malta- Paola PLA 1702
Phone: +356 (0)21 / 697 816
Fax: +356 (0)21 / 697 817

INTEHSIS srl **MOLDOVA**
bld. Traian 23/1
MD-2060 Kishinev
Phone: +373 (0)22 / 66 4242
Fax: +373 (0)22 / 66 4280

HIFLEX AUTOM.TECHNIEK B.V. **NETHERLANDS**
Wolweverstraat 22
NL-2984 CD Ridderkerk
Phone: +31 (0)180 – 46 60 04
Fax: +31 (0)180 – 44 23 55

Koning & Hartman b.v. **NETHERLANDS**
Haarlerbergweg 21-23
NL-1101 CH Amsterdam
Phone: +31 (0)20 / 587 76 00
Fax: +31 (0)20 / 587 76 05

Beijer Electronics AS **NORWAY**
Postboks 487
NO-3002 Drammen
Phone: +47 (0)32 / 24 30 00
Fax: +47 (0)32 / 84 85 77

Sirius Trading & Services srl **ROMANIA**
Aleea Lacul Morii Nr. 3
RO-060841 Bucuresti, Sector 6
Phone: +40 (0)21 / 430 40 06
Fax: +40 (0)21 / 430 40 02

Craft Con. & Engineering d.o.o. **SERBIA**
Bulevar Svetog Cara Konstantina 80-86
SER-18106 Nis
Phone: +381 (0)18 / 292-24-4/5
Fax: +381 (0)18 / 292-24-4/5

INEA SR d.o.o. **SERBIA**
Izletnicka 10
SER-113000 Smederevo
Phone: +381 (0)26 / 617 163
Fax: +381 (0)26 / 617 163

AutoCont Control s.r.o. **SLOVAKIA**
Radlinského 47
SK-02601 Dolny Kubin
Phone: +421 (0)43 / 5868210
Fax: +421 (0)43 / 5868210

CS MTrade Slovensko, s.r.o. **SLOVAKIA**
Vajanskeho 58
SK-92101 Piestany
Phone: +421 (0)33 / 7742 760
Fax: +421 (0)33 / 7735 144

INEA d.o.o. **SLOVENIA**
Stegne 11
SI-1000 Ljubljana
Phone: +386 (0)1 / 513 8100
Fax: +386 (0)1 / 513 8170

Beijer Electronics AB **SWEDEN**
Box 426
SE-20124 Malmö
Phone: +46 (0)40 / 35 86 00
Fax: +46 (0)40 / 35 86 02

Omni Ray AG **SWITZERLAND**
Im Schörli 5
CH-8600 Dübendorf
Phone: +41 (0)44 / 802 28 80
Fax: +41 (0)44 / 802 28 28

GTS **TURKEY**
Bayraktar Bulvari Nutuk Sok. No:5
TR-34775 Yukari Dudullu-Umraniye-ISTANBUL
Phone: +90 (0)216 526 39 90
Fax: +90 (0)216 526 3995

CSC Automation Ltd. **UKRAINE**
4-B, M. Raskovoyi St.
UA-02660 Kiev
Phone: +380 (0)44 / 494 33 55
Fax: +380 (0)44 / 494-33-66

EURASIAN REPRESENTATIVES

Kazpromautomatics Ltd. **KAZAKHSTAN**
Mustafina Str. 7/2
KAZ-470046 Karaganda
Phone: +7 7212 / 50 11 50
Fax: +7 7212 / 50 11 50

MIDDLE EAST REPRESENTATIVES

TEXEL ELECTRONICS Ltd. **ISRAEL**
2 Ha´umanut, P.O.B. 6272
IL-42160 Netanya
Phone: +972 (0)9 / 863 39 80
Fax: +972 (0)9 / 885 24 30

CEG INTERNATIONAL **LEBANON**
Cebaco Center/Block A Autostrade DORA
Lebanon - Beirut
Phone: +961 (0)1 / 240 430
Fax: +961 (0)1 / 240 438

AFRICAN REPRESENTATIVE

CBI Ltd. **SOUTH AFRICA**
Private Bag 2016
ZA-1600 Isando
Phone: + 27 (0)11 / 977 0770
Fax: + 27 (0)11 / 977 0761

MITSUBISHI ELECTRIC
FACTORY AUTOMATION

Mitsubishi Electric Europe B.V. /// FA - European Business Group /// Gothaer Straße 8 /// D-40880 Ratingen /// Germany
Tel.: +49(0)2102-4860 /// Fax: +49(0)2102-4861120 /// info@mitsubishi-automation.com /// www.mitsubishi-automation.com