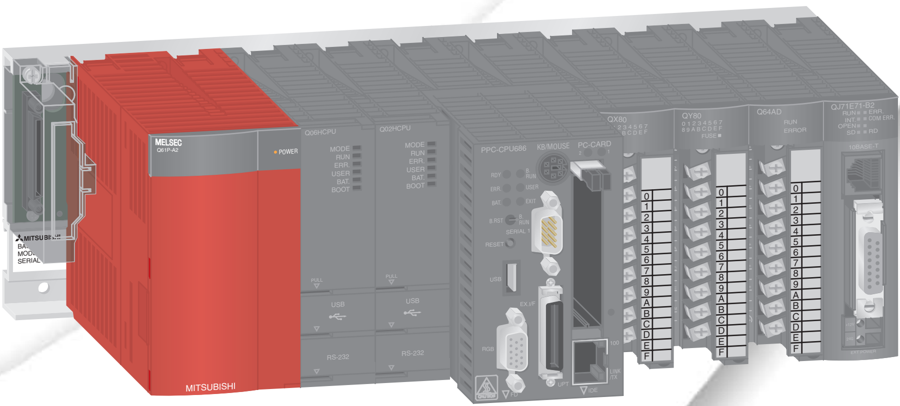
MITSUBISHI ELECTRIC
FACTORY AUTOMATION

MELSEC SYSTEM Q

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.

Training Manual
GX IEC Developer Programming Software Package
Art.-no.: 170295

Version			**Changes / Additions / Corrections**
A	03/2006	pdp	First edition
B	08/2006	pdp-dk	Correction on page 2-5 Changed illustration on page 2-9 Addition of modules in section 2.9.2 on page 2-37

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

This course has been specially produced as an introduction to Mitsubishi's Q-Series range of modular PLC's utilising the GX-IEC Developer Version 7 software package.

The course content has been selectively produced to provide an introduction into the functionality of the Mitsubishi range of Q-Series 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.

This material covered in this document complements and provides a lead-in to the 'follow-on' course; "Q-Series Advanced Training - Using GX-IEC Developer".

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 Mitsubishi Q-Series Training Rig. 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 (1–100 Hz and 0.1– 10 kHz): X7
- 6 Digital Output LED Indicators: Y0-Y5
- 4 Analogue Inputs: Q64AD Located at Head Address 30H
- 4 Analogue Outputs: Q64DA Located at Head Address 40H.

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 constantmaintenance

2.1.4 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** (covered in detail later in this document).

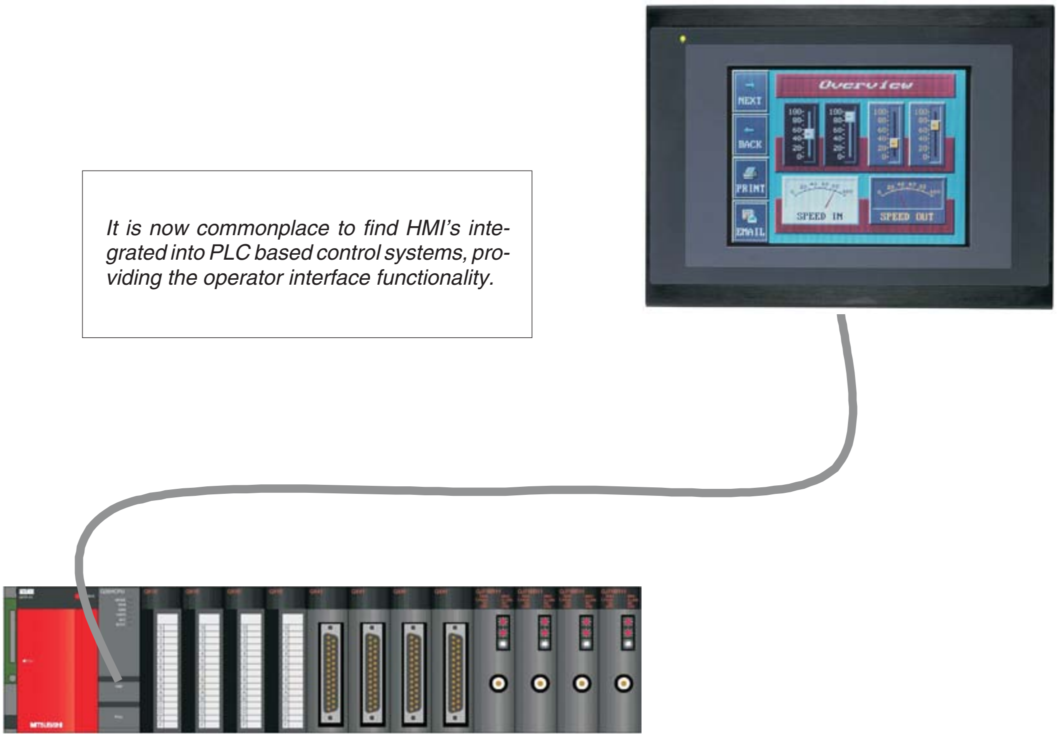
Until recently there has been no formal programming standard for PLC’s. The introduction of the **IEC 61131-3** Standard in 1998 provides a more formal approach to coding. Mitsubishi Electric has developed a programming package, “**GX-IEC Developer**”. This enables IEC compliant coding to be adopted.

2.1.5 SCADA and HMI

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 Hardware Configuration

This section deals with the design concepts and configuration of a Q-Series system.

2.2.1 Specifying a PLC System

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

External devices, Inputs and Outputs

- Input/Output Requirements
- System Signal Voltage: 24V DC, 110V/240VAC
- If 24V DC inputs then: NPN (Sink) or PNP (Source) devices
- Output Configuration: Transistor (Sink/Source), Triac, Relay or Volt Free Relay contact

Power supply requirements

- Supply voltage: 24VDC, 110V/240VAC

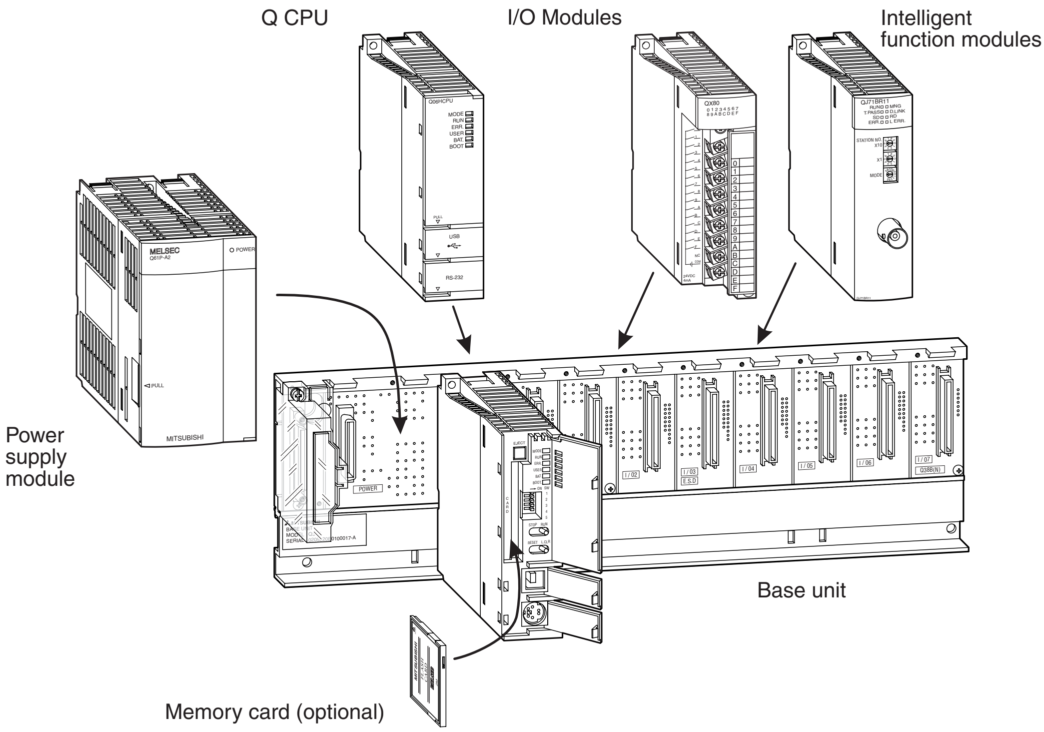
Intelligent Modules

- Number of modules in system
- External power supply requirements

2.3 Qn Series PLC Overview

The following information represents an overview of the configuration and format of the Qn PLC hardware. Data is also provided on the internal and operational specification of the Qn PLC systems.

2.3.1 System Configuration



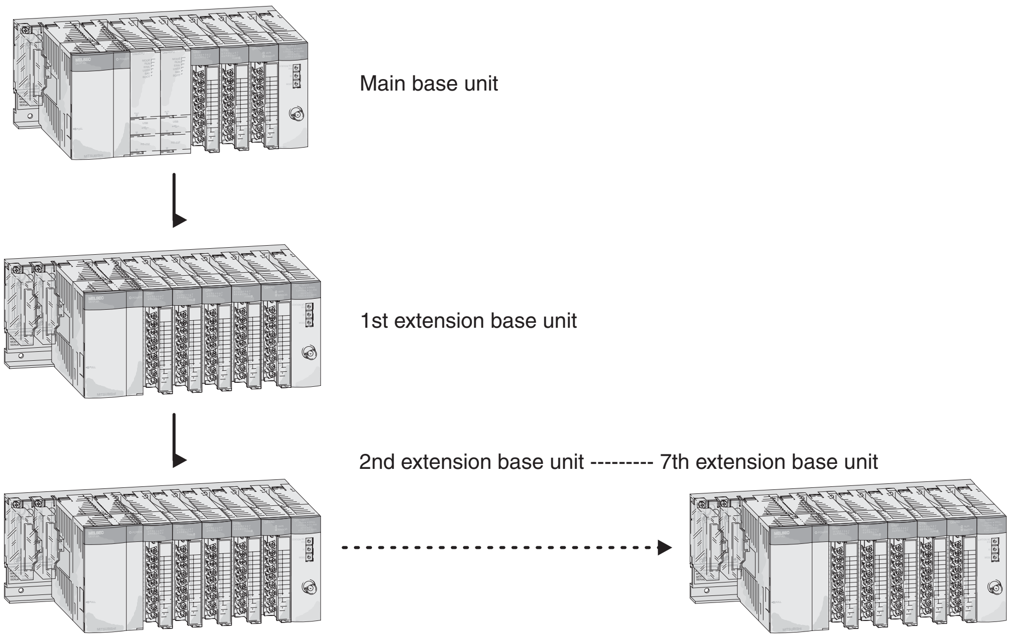
The CPU and modules are held in a base unit which has an internal bus connection for communication between the individual modules and the CPUs. The power supply module which supplies the voltage for the entire system is also installed on this base unit.

The base units are available in 4 different versions with 3 to 12 module slots. Each base unit can be supplemented by means of an extension unit providing additional slots.

If you wish to keep open the option of subsequent extension of your PLC or if you have free slots on your base unit, you can insert dummy modules here. They serve to protect the free slots from soiling or from mechanical effects but can also be used for reserving I/O points.

For cabling larger systems and machines - e.g. in a modular design - the use of remote I/O modules offers additional communications facilities.

Main Base and Extension Base Configuration



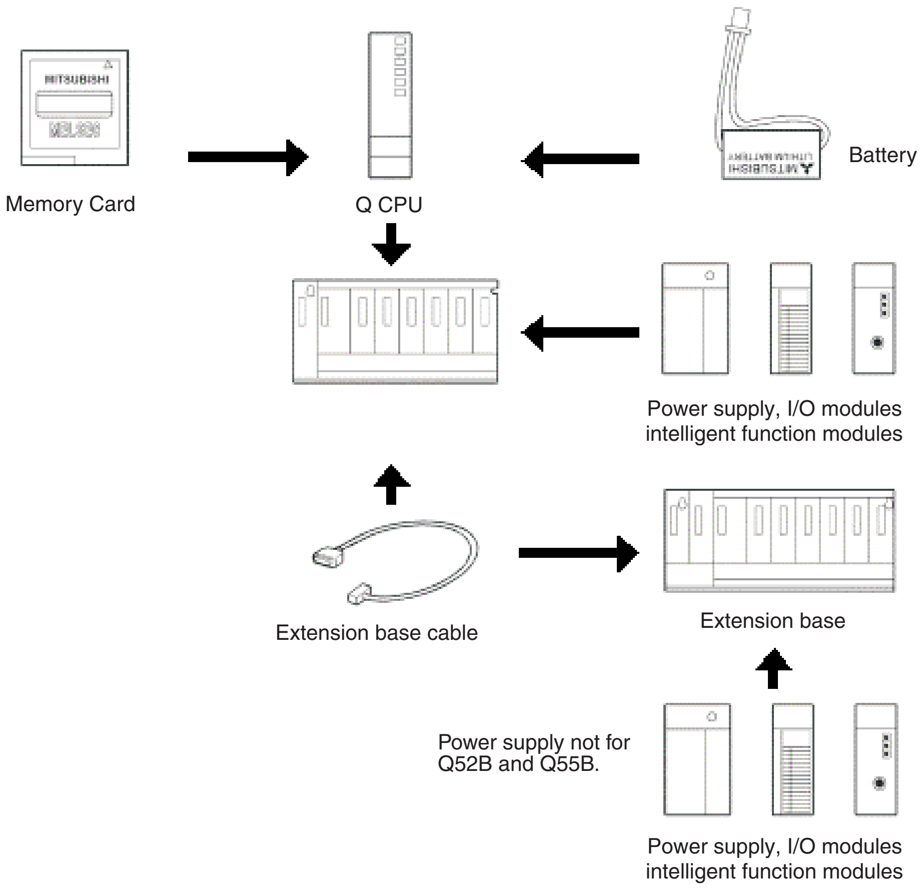
The base unit and extension units are simply connected to one another by extension cables. These connecting cables also supply the extension units with the operating voltage of 5 V DC.

Up to seven extension units with up to 64 modules can be connected to base units or extension base units. The maximum length of the extensions cables is 13.2 m.

When choosing the power supply module, the total power consumption of the I/O modules, of the special function modules and of the peripherals must be taken into account. If necessary, an extension unit with a further power supply module should be used.

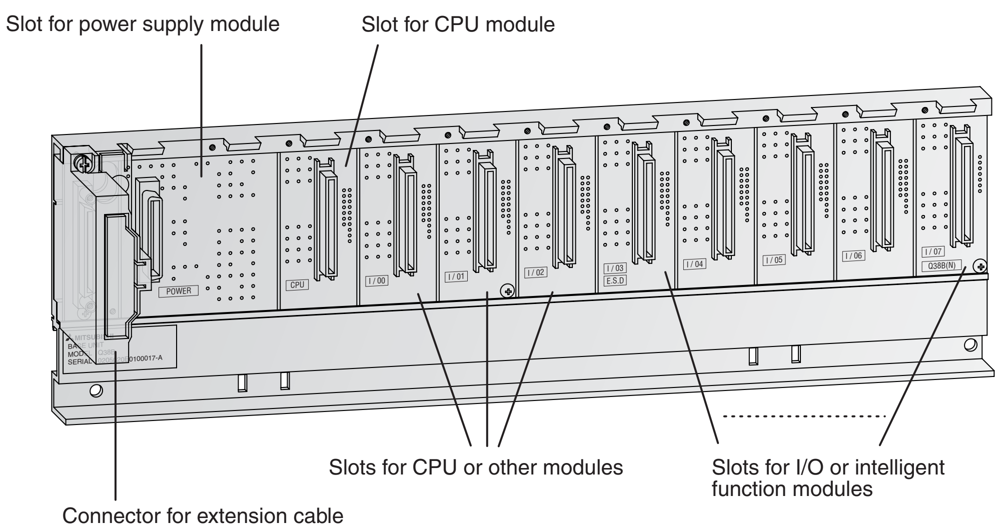
Number of extension base units

- Up to 4 extension base units can be connected to a main base unit in which a Q00CPU or Q01CPU is installed. The maximum number of loadable modules is 24.
- A system using Q02CPU, Q02HCPU, Q06HCPU, Q12HCPU or Q25HCPU can be extented by up to 7 extension base units.The total number of I/O and intelligent function modules in all base units is 64.



2.3.2 Base units

The main base units provide slots for a power supply module, up to four CPU modules,and I/O and intelligent function modules. I/O and intelligent function modules can also be mounted on the extension base units.The base units can be installed directly using screws or on a DIN rail using adapters.



The following table shows the available base units.

Item	Main base units				
	Q33B	Q35B	Q38B	Q38RB	Q312B
Loadable power supply modules	1	1	1	2*	1
Number of slots for I/O or intelligent funktion modules	3	5	8	8	12

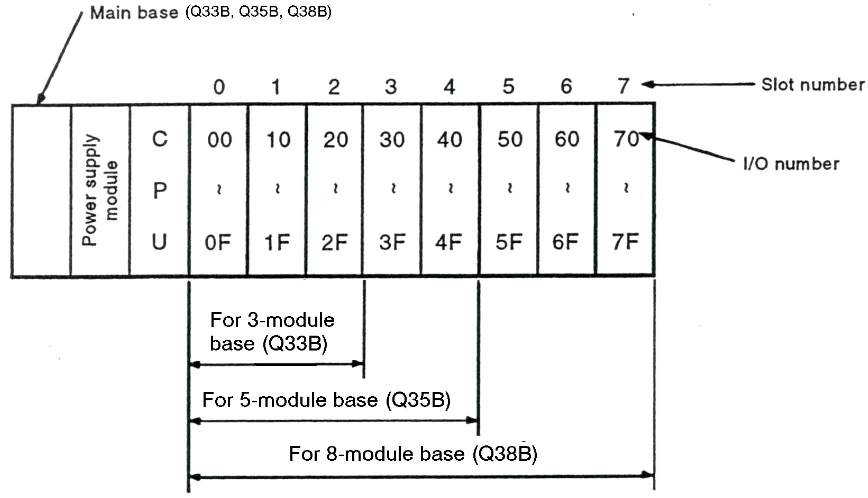
* In this main unit redundant power supply modules can be used.

Item	Extension base units						
	Q52B	Q55B	Q63B	Q65B	Q68B	Q68RB	Q612B
Loadable power supply modules	—	—	1	1	1	2*	1
Number of slots for I/O or intelligent funktion modules	2	5	3	5	8	8	12

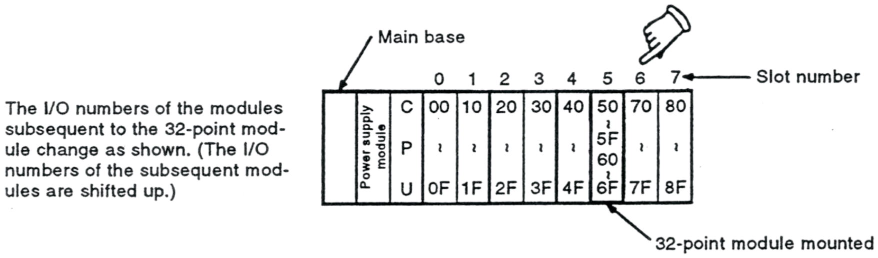
* In this extension base unit redundant power supply modules can be used.

2.3.3 Main base I/O numbering

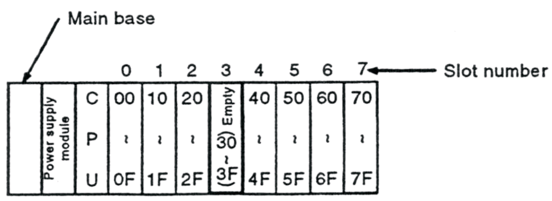
I/O numbers are assigned to the I/O modules mounted on the main base unit as described below. This also applies to special function modules.



- I/O numbers are assigned to one slot (one module) in increments of 16 points (0 to F^H) in ascending numerical order.
  This means that I/O numbering assumes that all slots have 16-point modules mounted in them.
  For example, when the fifth slot has a 32-point module mounted in it, I/O numbers are assigned as follows:



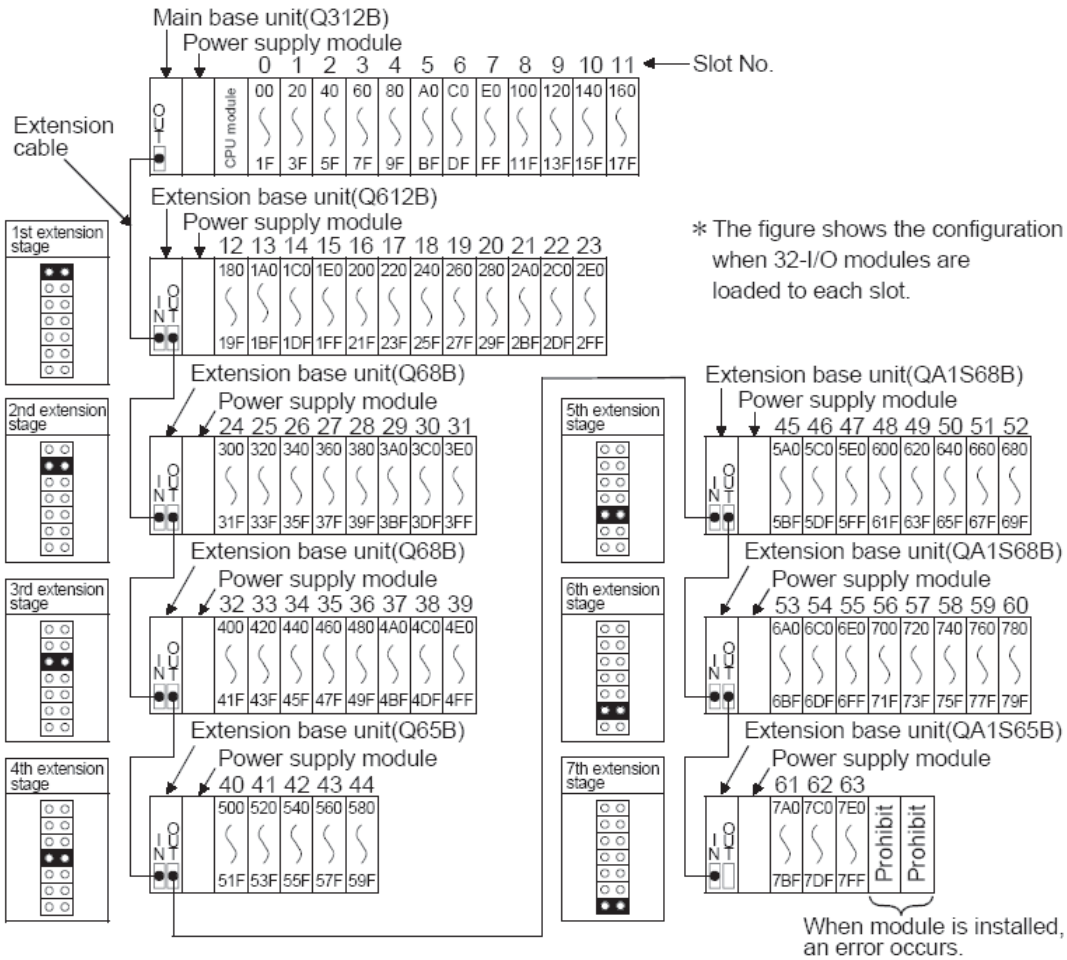
- 16 points (0 to F^H) are also assigned to an empty slot (without an I/O module).
  For example, when the third slot is empty, it is numbered as shown below:



2.3.4 Extension base I/O numbering

The slots of extension bases are also numbered in increments of 16 points in numerical order.

- The first slot of any extension base is numbered following the last number of the main base or preceding extension base.
- An extension base cannot be connected to a 2 slot main base.
- Connect extension bases when more slots are needed in addition to the main base unit. Their I/O numbers are assigned as follows:



2.4 Extensions Base Cables

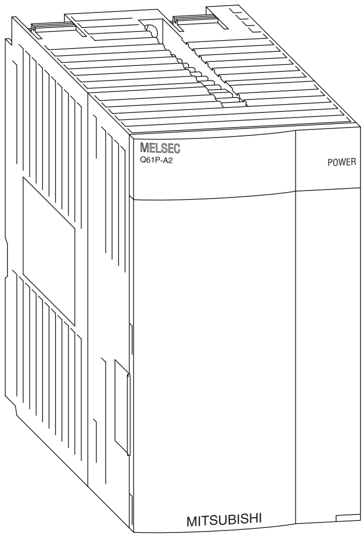
The extension base cables are used for connections between the base units.

Type	QC05B	QC06B	QC12B	QC30B	QC50B	QC100B
Cable length	0.45 m	0.50 m	1.2 m	3.0 m	5.0 m	10,0 m

The overall distance of all extension cables must not exceed 13.2 m.

For connection of the base units without an own power supply (Q52B, Q55B) the cable QC05B is recommended.

2.5 Power Supply Modules



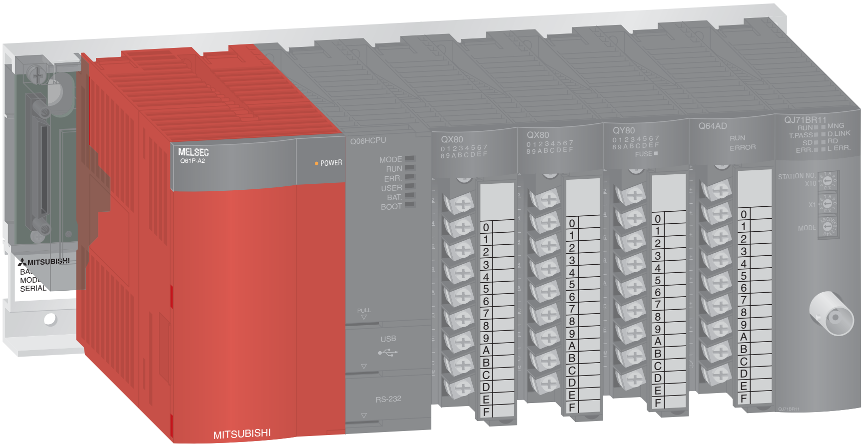
The power supply modules supply 5 V DC to each module on the base unit.
Power supply modules with input voltages of 24 V DC or 240 V AC are available.

Item	Q63P	Q61P-A1	Q61P-A2	Q62P	Q64P
Input voltage	24 V DC	100 – 120 V AC	200 – 220 V AC	100 – 240 V AC	100 – 120 V AC 200 – 240 V AC
Power consumption	45 W	105 VA	105 VA	105 VA	105 VA
Output voltage	5 V DC	5 V DC	5 V DC	5 V DC, 3 A 24 V DC, 0.6 A	5 V DC
Output current	6 A	6 A	6 A		8.5 A

2.5.1 Selection of an appropriate Power Supply

The total current consumption of the installed modules must be smaller than the rated output current of the power supply module. Reduce the number of modules on the base unit, if the current consumption is too high.

Example calculation of the total current consumption

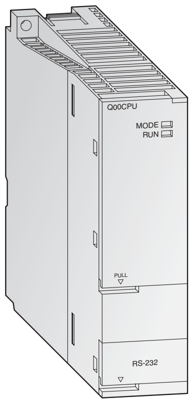


Module	Description	Current consumption
Q06HCPU	CPU module	0.64 A
QX80	Digital input module	0.16 A
QX80	Digital input module	0.16 A
QY80	Digital output module	0.008 A
Q64AD	A/D-converter module	0.63 A
QJ71BR11	MELSECNET/H module	0.75 A
Total current consumption		2.42 A

The total current consumption is 2.42 A. The installed power supply module is able to deliver a current of 6 A. This configuration will work without problems

2.6 CPU Modules

Basic PLC CPUs



The CPU modules of the MELSEC System Q are available as single and multi processor CPUs through which they achieve a wide application range. The performance of the controller here grows with the application by simply replacing the CPU (except Q00J).

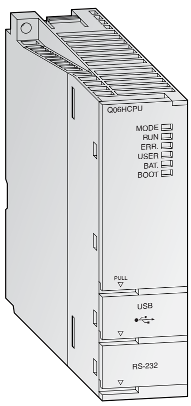
While Q00CPU and Q01CPU are classical separate CPUs, the Q00JCPU forms an inseparable unit consisting of CPU, power supply and base unit and thus enables a low-priced entry into the modular PLC technology.

The standard CPUs were developed especially for applications where a compact system configuration easily to be realized is to the fore.

Special features:

- Every CPU is equipped with an RS232C interface for easy programming and monitoring from a personal computer or operating panel.
- Integrated Flash ROMs for memory operation without additional memory cards.
- Processing the inputs and outputs with refresh mode

High performance CPUs



With the high-performance CPUs a high processing speed and expandability are to the fore. They provide a great variety of functions and an even optimized programming and debugging environment to ensure a flexible response to all systems.

The two process CPU models Q12PHCPU and Q25PHCPU have extended control functions with two degrees of freedom, PID cascading and autotuning. These processors also feature a set of 52 process instructions and support an unlimited number of PID loops

Special features:

- Every multi processor H-CPU is equipped with an USB interface for easy programming and monitoring from a personal computer.
- Processing the inputs and outputs with refresh mode
- Floating point arithmetic according to IEEE 754
- Special statements for processing PID control loops
- Mathematical functions, such as angle/exponential functions and logarithm
- Hot-swap module replacement in RUN mode (with process CPUs)
- Multi processor mode is possible with up to 4 CPU modules.

2.6.1 CPU Specification

Feature		Q00CPU	Q01CPU	Q02CPU	Q02HCPU	Q06HCPU	Q12HCPU	Q25HCPU
Control method		Repeated operation using stored program						
I/O control method		Refresh mode						
Programming language		IEC ladder, lögic symbolic language, list, structured text (ST), SFC						
Processing speed	LD	160 ns	100 ns	79 ns	34 ns			
	MOV	560 ns	350 ns	237 ns	102 ns			
	Mixed instructions per µs	2.0	2.7	4.4	10.3			
	Floating point addition	27 µs*		1.8 µs	0.78 µs			
Number of instrucions (without instructions for intelligent function modules)		249		363				
Processing of floating point numbers		Supported*		Supported				
Processing of character strings		$MOV is supported only		Supported				
PID control		Supported*		Supported				
Special functions (such as trigonometrical functions, extraction of root or logarithm)		Supported*		Supported				

* For Q00/Q01CPU function version B (First 5 digits of serial number are "04122" or later)

Feature		Q00CPU	Q01CPU	Q02CPU	Q02HCPU	Q06HCPU	Q12HCPU	Q25HCPU
Constant scan (program start at given time intervals)		1 to 2000 ms (can be specified in 1 ms increments)		0.5 to 2000 ms (can be specified in 0.5 ms increments)				
Program capacity (number of steps)		8 k	14 k	28 k		60 k	124 k	252 k
Memory capacity	Built-in program memory (drive 0)	94 kbytes		112 kbytes		240 kbytes	496 kbytes	1 MB
	RAM memory card (drive 1)	—		Capacity of loaded memory card (maximum 1 MB)				
	ROM memory card (drive 2)	—		Capacity of loaded memory card (maximum 4 MB for flash cards and 32 MB for ATA cards)				
	Built-in RAM (drive 3)	128 kbytes *		64 kbytes			256 kbytes	
	Built-in ROM (drive 4)	94 kbytes		112 kbytes		240 kbytes	496 kbytes	1 MB
	Common memory for multi processor mode	1 kbytes **		8 kbytes				
I/O points	Total (including remote I/O)	2048		8192				
	Local I/O	1024		4096				

* 64 k bytes for function version A

** For Q00/Q01CPU function version B (First 5 digits of serial number are "04122" or later)

Number of Devices

Device (Device symbol)	Q00CPU	Q01CPU	Q02CPU	Q02HCPU	Q06HCPU	Q12HCPU	Q25HCPU
Internal relay (M)	8192		8192				
Latch relay (L)	2048		8192				
Link relay (B)	2048		8195				
Timer (T)	512		2048				
Retentive Timer (ST)	0		0				
Counter (C)	512		1024				
Data register (D)	11136		12288				
Link register (W)	2048		8196				
Annunciator (F)	1024		2048				
Edge relay (V)	1024		2048				

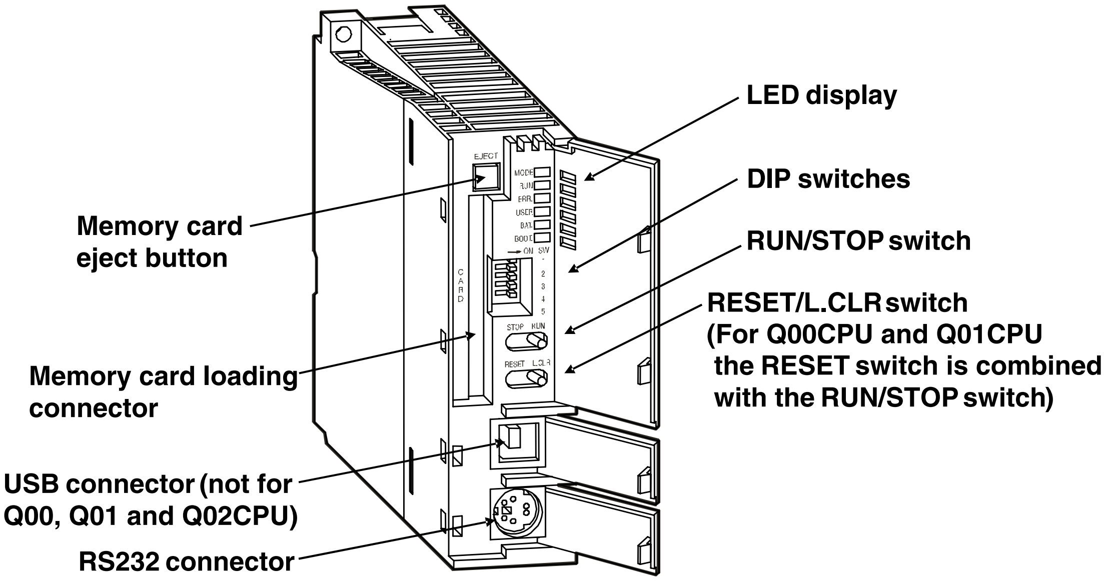
The table above indicates the default number of points. These can be changed in the parameter configuration.

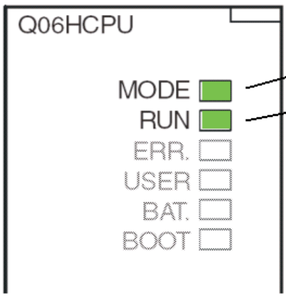
Device (Device symbol)	Q00CPU	Q01CPU	Q02CPU	Q02HCPU	Q06HCPU	Q12HCPU	Q25HCPU
File register (R)	32768		32768 (when the built-in memory is used)			131072 (built-in memory)	
Special link relay (SB)	1024		2048				
Special link register (SW)	1024		2048				
Step relay (S)	2048 (S0 to 127/block)		8192				
Index register (Z)	10		16				
Pointer (P)	300		4096				
Interrupt pointer (D)	128		256				
Special relay (SM)	1024		2048				
Special register (SD)	1024		2048				
Function input	16		16				
Function output	16		16				
Function register	5		5				

You can increase the number of file register for the Q02CPU, Q02HCPU, Q06HCPU, Q12HCPU, and Q25HCPU to up to 1 041 408 points by using a SRAM or flash card.

QnCPU – Operating Items

Feature	Q00CPU	Q01CPU	Q02CPU	Q02HCPU	Q06HCPU	Q12HCPU	Q25HCPU
Switch operation	RUN, STOP, RESET		RUN, STOP, RESET, L.CLR (Reset of the latched devices)				
External interfaces	RS232		RS232	RS232, USB			
Memory card	Not available		Available				
LED display	RUN, ERR.		MODE, RUN, ERR., USER, BAT., BOOT, POWER				
Current consumption @ 5 V DC	0.25 A	0.27 A	0.60 A	0.64 A			





LED Display

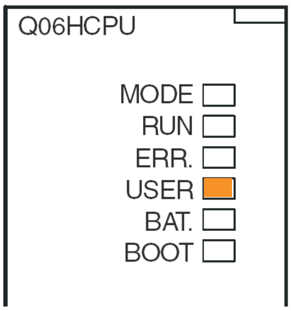
Green: Q mode	
ON:	During operation in “RUN“ mode
OFF:	During “STOP“ mode or after detection of an error occurrence that stopped the operation
Flicker:	RUN/STOP switch was switched from “STOP“ to “RUN“ after a program or a parameter was written during “STOP“. The CPU is not in “RUN“ mode.

Procedure to switch a Q CPU from “STOP“ to “RUN“ after the program or parameters have been changed during “STOP“:

1. Switch the RESET/L.CLR switch to the “RESET“ position.
2. Switch the RUN/STOP switch from “STOP“ to “RUN“.

However, when you want to set the CPU to “RUN“ without clearing the device information, switch the RUN/STOP switch from “STOP“ to “RUN“, then back to “STOP“ and finally to “RUN“ again.

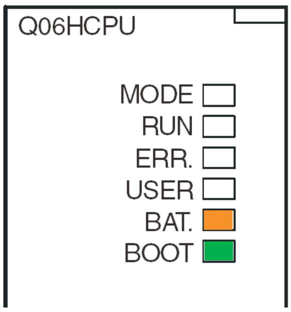
ERR and USER LED



ON:	**After the detection of an error during self-diagnostics. This error will not stop operation.**
OFF:	**Normal operation of the CPU**
Flicker:	**An error that stops the operation has been detected during self-diagnostic.**

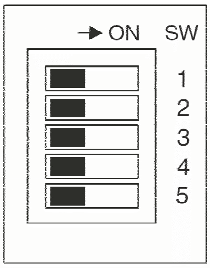
ON:	**An error has been detected by the CHK instruction or an annunciator (F) has been switched ON.**
OFF:	**Normal operation of the CPU**
Flicker:	**Execution of latch clear**

BAT and BOOT LED



ON:	**Voltage of either the battery for the CPU or the memory card is too low.**
OFF:	**Voltage is normal**

Q CPU DIP Switch Functions:



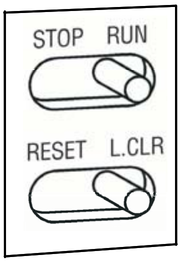
System protection	
OFF:	No protection
ON:	Protection

Used to specify where parameters are stored		
SW2	SW3	Parameter drive
OFF	OFF	Program memory (Drive 0)
ON	OFF	SRAM card (Drive 1)
OFF	ON	Flash card/ATA card (Drive 2)
ON	ON	Built-in ROM (Drive 4)

Parameters can not be stored in the built-in RAM (Drive 3).

All switches are shipped in the OFF position.

RUN/STOP and RESET/L.CLR Switches



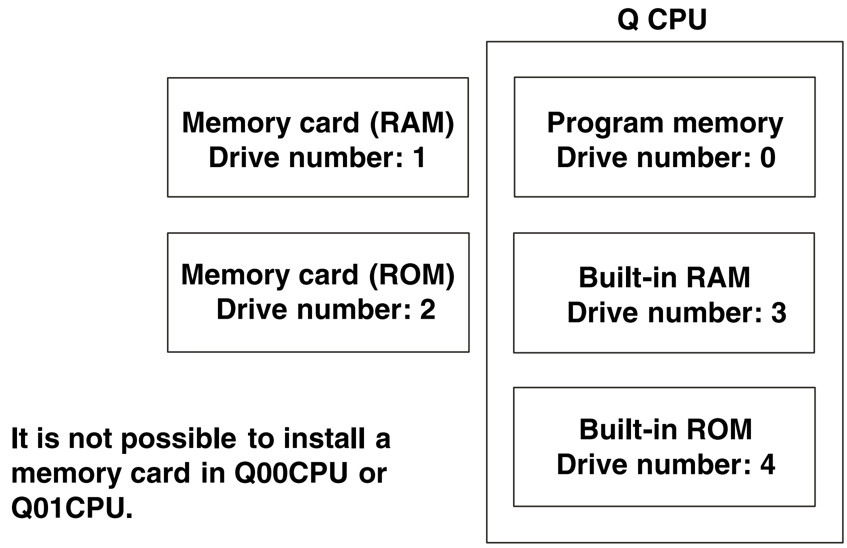
RUN: CPU executes sequence program

STOP: The execution of the sequence program is stopped

RESET: Used to perform hardware reset, operation fault reset, operation initialization etc.
After performing a reset, always return this switch to the neutral position.

L.CLR: Used to clear (turn either OFF or set to zero) all data in the parameter set latch area.
(Not available for Q00CPU and Q01CPU)

Memory Organisation



Organisation of Storage

Q00CPU and Q01CPU

Data	Built-in memory		
	Programm memory (Drive 0)	RAM (Drive 3)	ROM (Drive 4)
Program	●	○	●
Parameters	●	○	●
Intelligent function module parameters	●	○	●
Device comment	●	○	●
File register	○	●	○

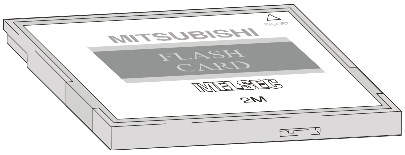
● = Storage is possible
○ = Storage is not possible

Q02CPU, Q02HCPU, Q06HCPU, Q12HCPU and Q25HCPU:

Data	Built-in memory			Memory cards		
	Programm memory (Drive 0)	RAM (Drive 3)	ROM (Drive 4)	RAM (Drive 1)	Flash ROM (Drive 2)	ATA ROM (Drive 2)
Program	●	○	●	●	●	●
Parameters	●	○	●	●	●	●
Intelligent function module parameters	●	○	●	●	●	●
Device comment	●	○	●	●	●	●
Device initial value	●	○	●	●	●	●
File register	○	●	○	●	●	○
Local devices	○	●	○	●	○	○
Debugging data	○	○	○	●	○	○
Failure history	○	○	○	●	○	○
Data file written by a FWRITE instruction	○	○	○	○	○	●

● = Storage is possible
○ = Storage is not possible

Memory Card Specifications

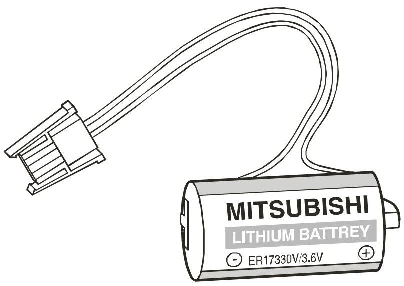


The write protect switch on the card will prevent any unintentional overwriting of stored data. A battery within the RAM memory card will hold the data during an interrupt of the power supply.

Available memory cards

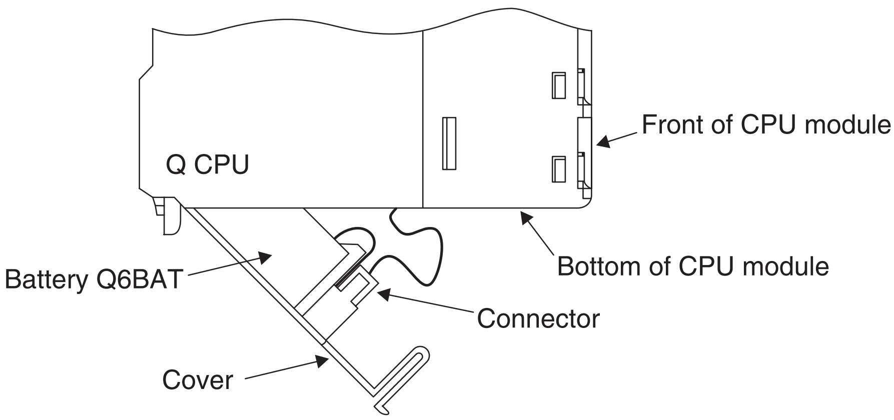
Designation	Type of memory	Memory capacity [Bytes]	Memory capacity [Number of files]	Number of writings
Q2MEM-1MBS	SRAM	1011 k	256	No limitation
Q2MEM-2MBS		2034 k	288	
Q2MEM-2MBF	Flash ROM	2035 k	288	100 000
Q2MEM-4MBF		4079 k		
Q2MEM-8MBA	ATA ROM	7940 k	512	1 000 000
Q2MEM-16MBA		15932 k		
Q2MEM-32MBA		31854 k		

Installation of the Battery for the CPU Module



The battery is installed at the bottom side of the Q CPU. During interruption of the power supply the battery can hold the data of the program memory , the built-in RAM and the clock for several thousand hours. However, this time depends on the type of CPU.

The CPU is shipped with its connector disconnected. Connect the battery before the CPU is used for the first time.



The battery should be changed every 10 years.

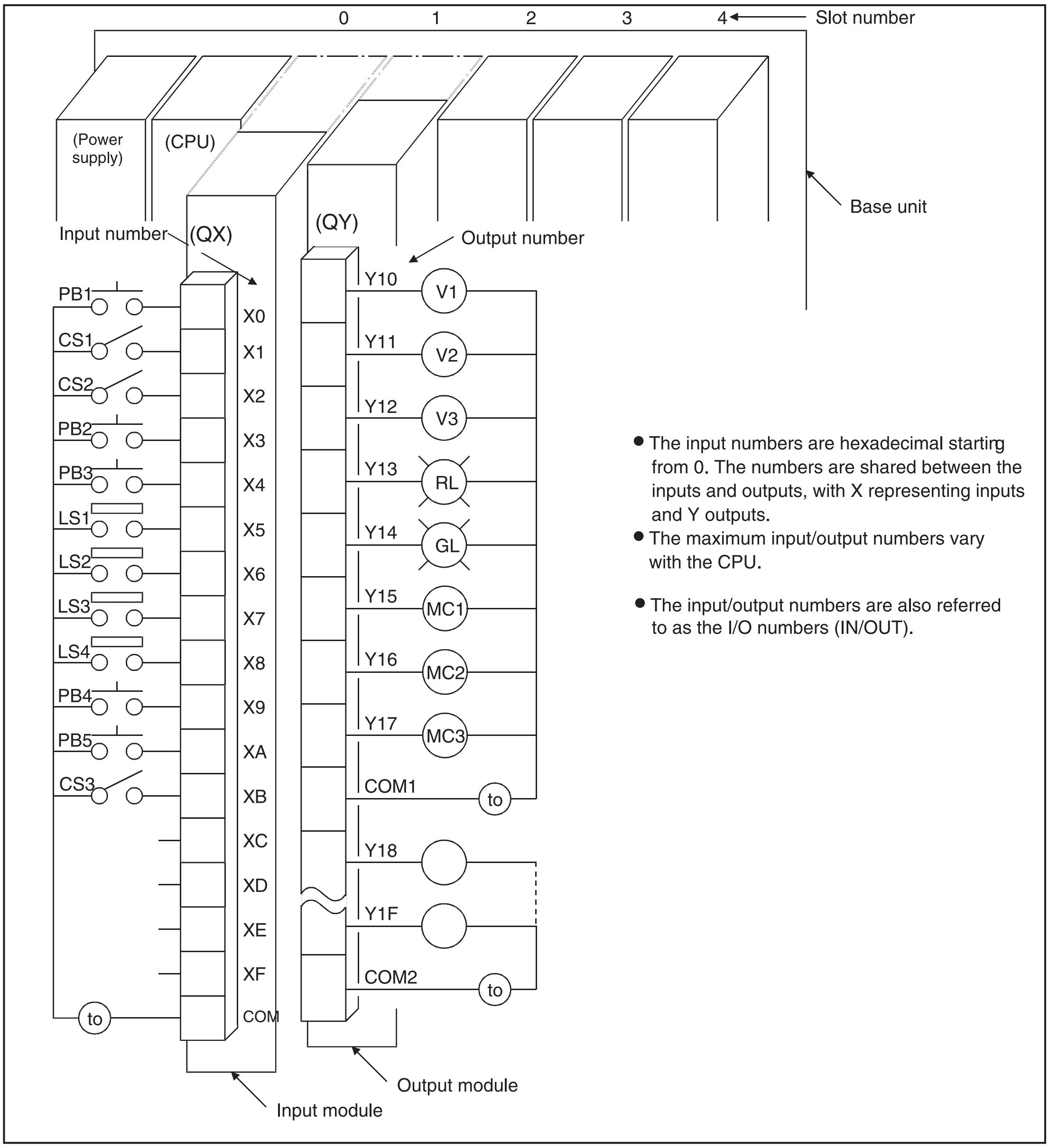
2.7 External I/O Signals and I/O Numbers

2.7.1 I/O device wiring

Signals from external input devices are replaced by input numbers, which are determined by the mounting position and terminal numbers of the input module connected and are handled in the program.

The outputs (coils) of the program operation results use output numbers which are determined by the mounting position and terminal numbers of the output module with which external output devices are connected.

As can be seen in the following examples, the I/O numbering system used is Hexadecimal. This is sensible as the PLC system is based on a 16 bit platform, it therefore follows that the addressing is also in this format.



- The input numbers are hexadecimal starting from 0. The numbers are shared between the inputs and outputs, with X representing inputs and Y outputs.
- The maximum input/output numbers vary with the CPU.
- The input/output numbers are also referred to as the I/O numbers (IN/OUT).

Inputs & Outputs

The Q-Series range of controllers can be considered to be made up of three parts:

- CPU (Central Processing Unit)
- Input circuit
- Output circuit

The input circuitry provides the PLC CPU with information from a wide variety of input signals.

Typical Input Devices

The Input signals can come from a wide variety of devices i.e.

- Push buttons.
- Rotary switches.
- Key switches.
- Limit switches.
- Level sensors.
- Flow rate sensors
- Photo-electric detectors.
- Proximity detectors (Inductive or Capacitive).

  Proximity detectors usually provide a transistor output which can be either an NPN (Sink) or PNP (Source) transistor.

2.8 Digital Input and Output Modules

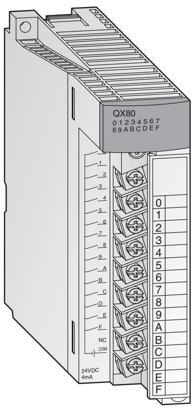
Overview of Digital I/O module types

Type		Number of inputs/outputs			
		8	16	32	64
Input modules	120 V AC	○	●	○	○
	240 V AC	●	○	○	○
	24 V DC	○	●	●	●
	24 V DC (High speed)	●	○	○	○
	5 V DC / 12 V DC	○	●	●	●
Output modules	Relay	●	●	○	○
	Individual relay	●	○	○	○
	Triac output	○	●	○	○
	Transistor output (sink)	●	●	●	●
	Transistor output (source)	○	●	●	○
Combined input/output modules		●	○	●	○

● = Module is available
○ = Module is not available

2.8.1 Digital Input Modules

Input modules are available for various input voltages:



Input voltage	Number of input points			
	8	16	32	64
5 – 12 V DC		QX70	QX71	QX72
24 V DC		QX80	QX81	QX82
24 V DC (Interrupt module)		QI60		
100 – 120 V AC		QX10		
100 – 240 V AC	QX28			

Modules with 8 or 16 connection points provide removable screw terminal blocks. The modules with 32 or 64 connection points are connected via a plug.

General PLC Input - Considerations

All inputs are isolated by Opto-couplers. This prevents the sensitive CPU electronics in the PLC from being affected by electrical noise spikes induced by external equipment.

Another common problem is contact bounce generated by electromechanical switches.

To avoid the PLC from being affected by these parasitic effects, the inputs are filtered so that the On/Off status will register an ‘On’ state only if the signal is stable for a period exceeding the filter coefficient (see note below).

This filter response time should be taken into account when programming as it will have a direct effect on the way the program will operate.

For the PLC to register a logical change in input condition, it will have to draw a minimum of 3mA; anything less than this will result in the Input not turning on.

The input will accept up to a 7mA signal, anything in excess of this could result in the input being damaged.

If higher speed input functionality is utilised where the input filter coefficient is reduced, care should be taken when using these inputs for digital signalling. Cables should be shielded and run separately to other potential sources of electrical noise!

If very high speed operation is required within the system then use of special modules such as or Interrupt of High Speed Counter should be adopted.

NOTE

A-Series: Standard Input Modules are preset to 10 ms Filter Coefficient.
Q-Series: The Filter Coefficient of the standard Input Modules is preset to 10 ms but may be individually adjusted in the range of 1 ms to 70 ms from within the Parameter setup of the CPU (See individual module specifications).

Source / Sink Inputs

This subject often causes confusion due to differing interpretations of the definition of Sink and Source by different manufacturer's each side of the Atlantic.

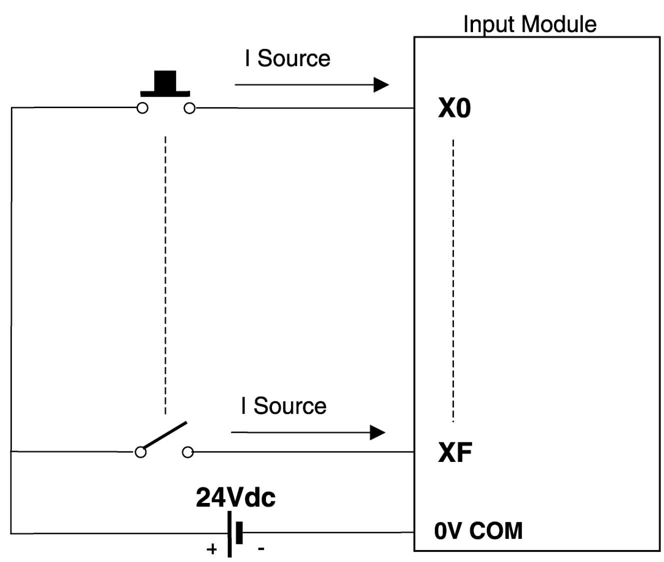
The term Source /Sink refers to the direction of current flow into or out of the input terminals of the PLC.

The following descriptions describe Mitsubishi's interpretation of the subject, which is shared by most other European and Far Eastern PLC manufacturers!

Source Input

When the PLC is connected for Source inputs, then the input signal current flows into the X inputs.

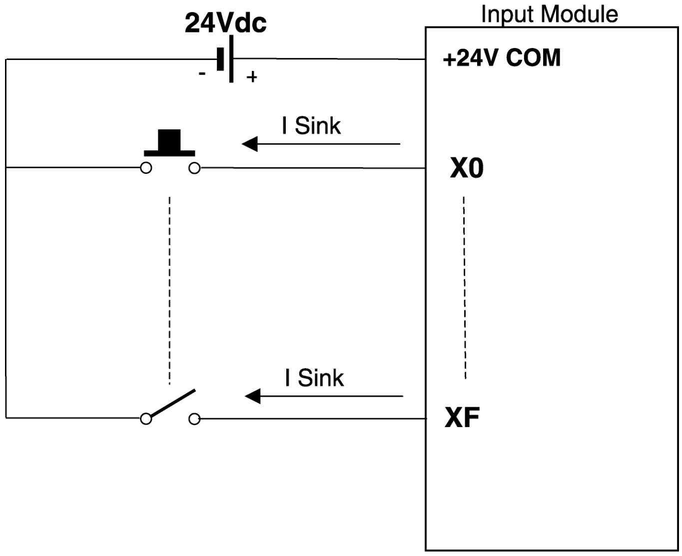
Source Input Configuration



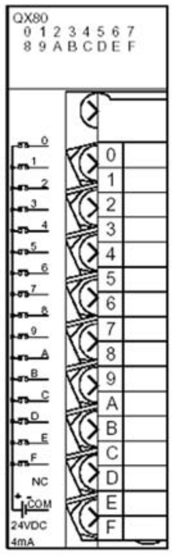
Sink Input

When the PLC is connected for Sink inputs, then the input signal current flows out of the X Inputs.

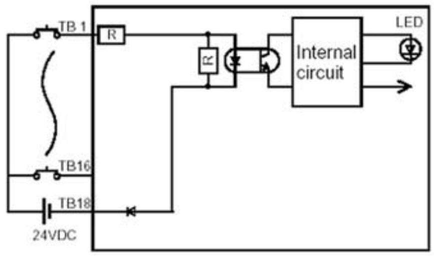
Sink Input Configuration



Source Inputs (Negative Common)- Module Details

Specifications \ Type		DC Input Module (Negative Common Type) QX80	Appearance
Number of input points		16 points	
Isolation method		Photocoupler	
Rated input voltage		24VDC (+20/-15%, ripple ratio within 5%)	
Rated input current		Approx. 4mA	
Input derating		No	
ON voltage/ON current		19V or higher/3mA or higher	
OFF voltage/OFF current		11V or lower/1.7mA or lower	
Input impedance		Approx. 5.6kΩ	
Response time	OFF to ON	1ms/5ms/10ms/20ms/70ms or less (CPU parameter setting) * Initial setting is 10ms.	
	ON to OFF	1ms/5ms/10ms/20ms/70ms or less (CPU parameter setting) * Initial setting is 10ms.	
Dielectric withstand voltage		560VAC rms/3 cycles (altitude 2000m (6557.38ft.))	
Insulation resistance		10MΩ or more by insulation resistance tester	
Noise immunity		By noise simulator of 500Vp-p noise voltage, 1μs noise width and 25 to 60Hz noise frequency	
		First transient noise IEC61000-4-4: 1kV	
Protection of degree		IP2X	
Common terminal arrangement		16 points/common (common terminal: TB18)	
Number of I/O points		16 (I/O allocation is set as a 16-points input module)	
Operation indicator		ON indication (LED)	
External connections		18-point terminal block (M3×6 screws)	
Applicable wire size		0.3 to 0.75mm^2 core (2.8mm (0.11in.) OD max.)	
Applicable crimping terminal		R1.25-3 (sleeved crimping terminals cannot be used.)	
5VDC internal current consumption		50mA (TYP. all points ON)	
Weight		0.16kg	

Input Circuit Detail

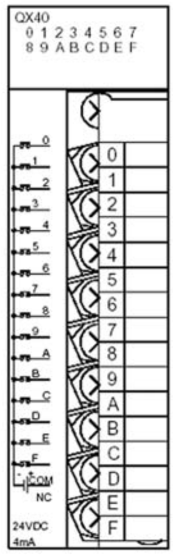
External Connections	Terminal Block Number	Signal Name
	TB1	X00
	TB2	X01
	TB3	X02
	TB4	X03
	TB5	X04
	TB6	X05
	TB7	X06
	TB8	X07
	TB9	X08
	TB10	X09
	TB11	X0A
	TB12	X0B
	TB13	X0C
	TB14	X0D
	TB15	X0E
	TB16	X0F
	TB17	Vacant
	TB18	COM

Direction of Source Current Flow

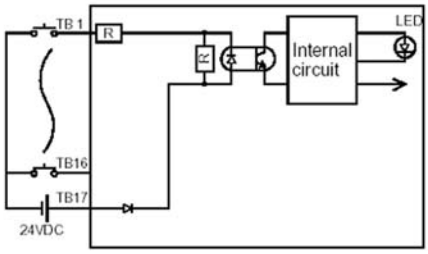
Referring to the preceding circuit diagram, when the push button is closed, the direction of current flow will be as follows:

- From the +24 Volt terminal of the external power supply, through the push button and into the TB1 (X0) input terminal i.e. Source Current.
- Through the input resistor network circuit and then through the LED.
- When current flows through the LED it will emit light, which in turn will cause the Photo-Transistor to turn ON.
- The function of the Opto-Isolator is to isolate the plant side 24 Volt input circuit from the sensitive 5 Volt PLC processor logic circuitry. This also provides noise immunity from the input.
- With the Photo-Transistor turning ON, this will cause a signal to be sent to the Input Image Table, to store the information that the input X0 is ON.
- The Input current now flows out of (TB18) COM terminal and then back to the terminal of the External power supply.

Sink Inputs (Positive Common)- Module Details

Specifications / Type		DC Input Module (Positive Common Type) QX40	Appearance
Number of input points		16 points	
Isolation method		Photocoupler	
Rated input voltage		24VDC (+20/-15%, ripple ratio within 5%)	
Rated input current		Approx. 4mA	
Input derating		No	
ON voltage/ON current		19V or higher/3mA or higher	
OFF voltage/OFF current		11V or lower/1.7mA or lower	
Input impedance		Approx. 5.6kΩ	
Response time	OFF to ON	1ms/5ms/10ms/20ms/70ms or less (CPU parameter setting) * Initial setting is 10ms.	
	ON to OFF	1ms/5ms/10ms/20ms/70ms or less (CPU parameter setting) * Initial setting is 10ms.	
Dielectric withstand voltage		560VAC rms/3 cycles (altitude 2000m (6557.38ft.))	
Insulation resistance		10MΩ or more by insulation resistance tester	
Noise immunity		By noise simulator of 500Vp-p noise voltage, 1μs noise width and 25 to 60Hz noise frequency	
		First transient noise IEC61000-4-4: 1kV	
Protection of degree		IP2X	
Common terminal arrangement		16 points/common (common terminal: TB17)	
Number of I/O points		16 (I/O allocation is set as a 16-points input module)	
Operation indicator		ON indication (LED)	
External connections		18-point terminal block (M3×6 screws)	
Applicable wire size		0.3 to 0.75mm^2 core (2.8mm (0.11in.) OD max.)	
Applicable crimping terminal		R1.25-3 (sleeved crimping terminals cannot be used.)	
5VDC internal current consumption		50mA (TYP. all points ON)	
Weight		0.16kg	

Input Circuit Detail

External Connections	Terminal Block Number	Signal Name
	TB1	X00
	TB2	X01
	TB3	X02
	TB4	X03
	TB5	X04
	TB6	X05
	TB7	X06
	TB8	X07
	TB9	X08
	TB10	X09
	TB11	X0A
	TB12	X0B
	TB13	X0C
	TB14	X0D
	TB15	X0E
	TB16	X0F
	TB17	COM
	TB18	Vacant

Direction of Sink Current Flow

In the preceding diagram, when the push button is closed, the direction of current flow will be as follows:

- From the +24 Volt terminal of the external power supply to the Common terminal (TB17) .
- Through the 1st LED and then through the input resistor network circuit to the TB1 (X0) input terminal.
- When current flows through the LED, it will then emit light which in turn will cause the Photo-Transistor to turn ON.
- The Photo-Transistor turning ON causes a signal to be sent to the Input Image Table, to store the information that the input X0 is ON.
- The Input current now flows out of the X0 input terminal i.e. ‘Sink Current’.
- It then flows through the push button and then back to the negative (0V) terminal of the external power supply.

Sensors: Proximity and Optical

There are 2 types of proximity sensor; Inductive and Capacitive. There are also many varieties of optical sensors that may be found in Industrial application. The supply voltages to these sensors are commonly 24V DC.

Most Opto and Proximity sensors utilise semiconductor outputs and these are available in two polarities, which are:

- PNP - (SOURCE)
- NPN - (SINK)

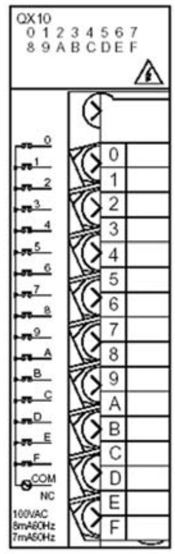
NOTE

When connecting devices to the physical PLC I/O, think of current flow rather than voltage levels. For example: Input Activated = current flowing. Input Deactivated = No current flowing.

AC Input - Module Details

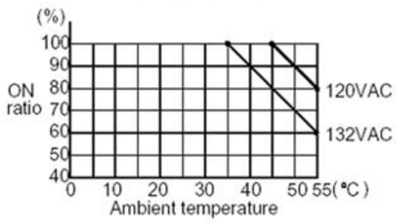
Specifications / Type		AC Input Module QX10
Number of input points		16 points
Isolation method		Photocoupler
Rated input voltage, frequency		100-120VAC (+10/-15%) 50/60Hz (±3Hz) (distortion factor within 5%)
Rated input current		Approx. 8mA (100VAC, 60Hz), approx. 7mA (100VAC, 50Hz)
Input derating		Refer to the derating chart.
Inrush current		Max. 200mA within 1ms (at 132VAC)
ON voltage/ON current		80VAC or higher/5mA or higher (50Hz, 60Hz)
OFF voltage/OFF current		30VAC or lower/1.7mA or lower (50Hz, 60Hz)
Input impedance		Approx. 12kΩ (60Hz), approx. 15kΩ (50Hz)
Response time	OFF to ON	15ms or less (100VAC 50Hz, 60Hz)
	ON to OFF	20ms or less (100VAC 50Hz, 60Hz)
Dielectric withstand voltage		1780VAC rms/3 cycles (altitude 2000m (6557.38ft.))
Insulation resistance		10MΩ or more by insulation resistance tester
Noise immunity		By noise simulator of 1500Vp-p noise voltage, 1μs noise width and 25 to 60Hz noise frequency
		First transient noise IEC61000-4-4: 1kV
Protection of degree		IP1X
Common terminal arrangement		16 points/common (common terminal: TB17)
Number of I/O points		16 (I/O allocation is set as a 16-points input module)
Operation indicator		ON indication (LED)
External connections		18-point terminal block (M3×6 screws)
Applicable wire size		0.3 to 0.75mm² core (2.8mm (0.11in.) OD max.)
Applicable crimping terminal		R1.25-3 (sleeved crimping terminals cannot be used.)
5VDC internal current consumption		50mA (TYP. all points ON)
Weight		0.17kg

Appearance

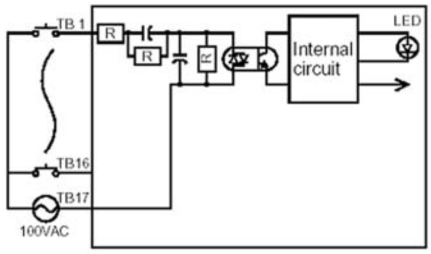


Input Circuit Detail

Derating Chart



External Connections

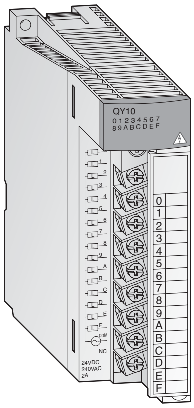


Terminal Block Number	Signal Name
TB1	X00
TB2	X01
TB3	X02
TB4	X03
TB5	X04
TB6	X05
TB7	X06
TB8	X07
TB9	X08
TB10	X09
TB11	X0A
TB12	X0B
TB13	X0C
TB14	X0D
TB15	X0E
TB16	X0F
TB17	COM
TB18	Vacant

With AC Input type modules, it is recommended that the same supply voltage to the PLC is used as for the inputs i.e. (100 - 120VAC). This minimises the possibility of an incorrect voltage being connected to the Inputs.

2.8.2 Digital Output Modules

The output modules of the Q-Series provide different switching elements for adaption to many control tasks:



Output type	Rated output voltage	Number of output points		
		8	16	32
Relay	24 V DC / 240 V AC	QY18A	QY10	
Triac	100 – 240 V AC		QY22	
Transistor	5 / 12 V DC		QY70	QY71
	12 / 24 V DC		QY80	QY81P
	5 – 24 V DC	QY68A		

Modules with 8 or 16 connection points are equipped with removable screw terminal blocks. The modules with 32 or 64 connection points are connected via a plug.

Output Types

Q-Series standard PLC outputs are available in four configurations:

- Relay
- Triac (SSR)
- Transistor (Source Type)
- Transistor (Sink Type)

Type	Advantages	Disadvantages
Relay	● Mixed voltage switching ● Volt-free operation possible ● High current switching capability	● Slow (max. 1 Hz) ● Finite reliability (electromechanical) ● Contact burn ● Noisy (electrical)
Triac	● High reliability ● Higher speed switching ● Suited to high duty switching applications	● AC operation only ● Current limited to 0.6 A /point ● Requires 10 ms to turn ON/OFF at AC 50 Hz
Transistor	● Very high reliability ● Very high speed switching ● Well suited to high duty switching applications	● Low voltage DC operation only ● Current limited to 0.1 A /point

Relay

This interface is more commonly used in the UK.

Electrical Isolation from the internal and external circuitry is achieved by the coils and the contacts of the output relays.

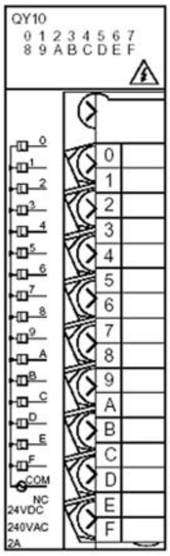
Modules are available as multiple outputs with isolated grouped commons or individually isolated ‘Volt Free’ outputs.

The operation of the output contact is driven by the internal CPU program.

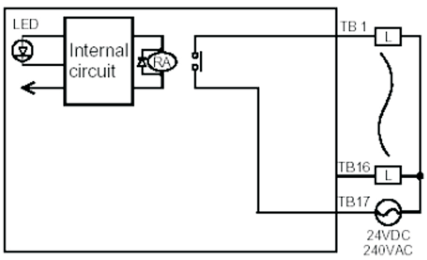
When the "END" instruction is triggered the PLC will REFRESH (update) the outputs from the Output Latch memory, an LED will light and the output contact will close.

The response for the operation of the relay is approximately 10 ms.

Relay Output Circuit Configuration

Specifications		Contact Output Module QY10	Appearance
Number of output points		16 points	
Isolation method		Relay	
Rated switching voltage, current		24VDC 2A (resistive load) 240VAC 2A (cos Φ =1) /point, 8A/common	
Minimum switching load		5VDC 1mA	
Maximum switching load		264VAC 125VDC	
Response time	OFF to ON	10ms or less	
	ON to OFF	12ms or less	
Life	Mechanical	20 million times or more	
	Electrical	Rated switching voltage/current load More than 100 thousand times or more	
		200VAC 1.5A, 240VAC 1A (COS Φ =0.7) 100 thousand times or more 200VAC 0.4A, 240VAC 0.3A (COS Φ =0.7) 300 thousand times or more	
		200VAC 1A, 240VAC 0.5A (COS Φ =0.35) 100 thousand times or more 200VAC 0.3A, 240VAC 0.15A (COS Φ =0.35) 300 thousand times or more	
		24VDC 1A, 100VDC 0.1A (L/R=7ms) 100 thousand times or more 24VDC 0.3A, 100VDC 0.03A (L/R=7ms) 300 thousand times or more	
Maximum switching frequency		3600 times/hour	
Surge suppressor		No	
Fuse		No	
Dielectric withstand voltage		2830VAC rms/3 cycles (altitude 2000m (6557.38ft.))	
Insulation resistance		10MΩ or more by insulation resistance tester	
Noise immunity		By noise simulator of 1500Vp-p noise voltage, 1 μs noise width and 25 to 60Hz noise frequency	
		First transient noise IEC61000-4-4: 1kV	
Protection of degree		IP1X	
Common terminal arrangement		16 points/common (common terminal: TB17)	
Number of I/O points		16 (I/O allocation is set as a 16-points output module)	
Operation indicator		ON indication (LED)	
External connections		18-point terminal block (M3×6 screws)	
Applicable wire size		0.3 to 0.75mm^2 core (2.8mm (0.11in.) OD max.)	
Applicable crimping terminal		R1.25-3 (sleeved crimping terminals cannot be used.)	
5VDC internal current consumption		430mA (TYP. all points ON)	
Weight		0.22kg	

Output Circuit Detail

External Connections	Terminal Block Number	Signal Name
	TB1	Y00
	TB2	Y01
	TB3	Y02
	TB4	Y03
	TB5	Y04
	TB6	Y05
	TB7	Y06
	TB8	Y07
	TB9	Y08
	TB10	Y09
	TB11	Y0A
	TB12	Y0B
	TB13	Y0C
	TB14	Y0D
	TB15	Y0E
	TB16	Y0F
	TB17	COM
	TB18	Vacant

Triac

Voltages of 240 V AC or 110 V AC can be used on separately commoned blocks.

As with all other output configurations the physical output is isolated by photocoupler.

The response of the Triac is obviously faster than the relay with a response time of 1msec to turn ON and 10 ms to turn OFF again.

Care should be taken when configuring your system so as not to overload the output circuitry. Referral to the relevant hardware module manual will give the correct loading.

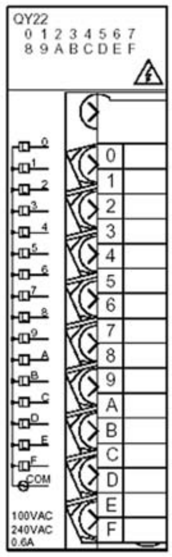
Because the leakage current in a Triac output circuit is greater than that of a relay circuit, care must be taken as this current is enough to cause indicators to illuminated and some miniature relays to hold their operation.
In fact, this is one of the most frequent causes of electric shock when working in cabinets controlled by PLC’s.

Special care must be taken when working in live environments with output circuits controlled by Triac devices, even if the outputs are apparently turned off!

Triac Output Circuit Configuration

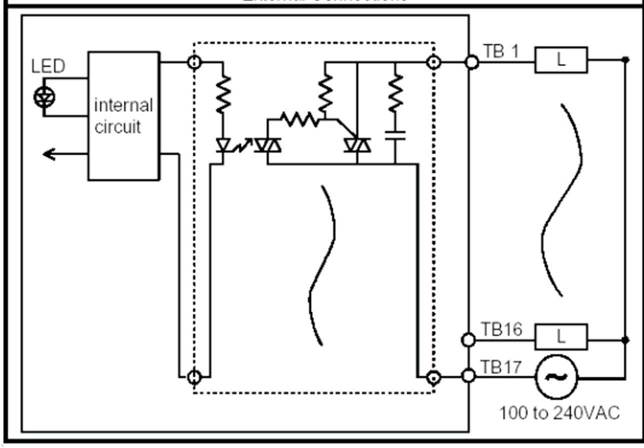
Specifications \ Type		TRIAC Output Module QY22
Number of output points		16 points
Isolation method		Photocoupler
Rated load voltage		100-240VDC (+20/-15%)
Maximum load current		0.6A/point, 4.8A/common
Minimum load voltage/current		24VAC 100mA, 100VAC 25mA, 240VAC 25mA
Maximum rush current		20A/cycle or less
Leakage current at OFF		3mA or lower (for 240V, 60Hz), 1.5mA or lower (for 120V, 60Hz)
Maximum voltage drop at ON		1.5V or lower
Response time	OFF to ON	1ms + 0.5Hz or less
	ON to OFF	1ms + 0.5Hz or less (rated load, resistance load)
Surge killer		CR absorber
Fuse		None (Attaching a fuse to external wiring is recommended. Refer to Section 1.2 (14))
Dielectric maximum voltage		2830VAC rms/3 cycles (altitude 2000m)
Insulation resistance		10MΩ or higher by insulation resistance meter
Noise immunity		By noise simulator of 1.5kVp-p noise voltage, 1μs noise width and 25 to 60Hz noise frequency
		First transient noise IEC61000-4-4: 1kV
Protection of degree		IP1X
Common terminal arrangement		16 points/common (common terminal: TB18)
Number of I/O points		16 (I/O allocation is set as a 16-points output module)
Operation indicator		ON indication (LED)
External connections		18-point terminal block (M3×6 screws)
Applicable wire size		Core cable: 0.3 to 0.75mm² (Outside diameter: 2.8mm or smaller)
Applicable connector terminal		R1.25-3 (Terminals with sleeve cannot be used)
5VDC internal current consumption		250mA (Max., all points ON)
Weight		0.40kg

Appearance



Output Circuit Detail

External Connections



Terminal Block Number	Signal Name
TB1	Y00
TB2	Y01
TB3	Y02
TB4	Y03
TB5	Y04
TB6	Y05
TB7	Y06
TB8	Y07
TB9	Y08
TB10	Y09
TB11	Y0A
TB12	Y0B
TB13	Y0C
TB14	Y0D
TB15	Y0E
TB16	Y0F
TB17	COM
TB18	Vacant

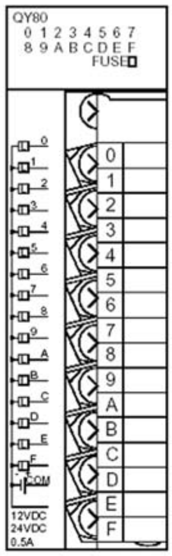
Transistor

As with all other output configurations the physical output is isolated by photocoupler.

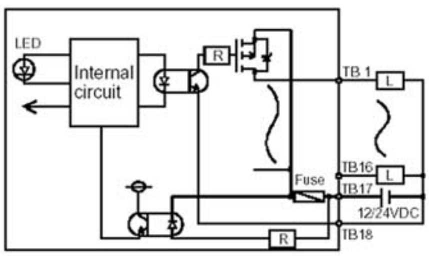
Response of the transistor in either direction is 1 ms at 24 V DC, 200 mA. The exact current handling capacity of each output is specified in the relevant hardware manual.

The Sink and Source Configurations are shown in the following module technical details.

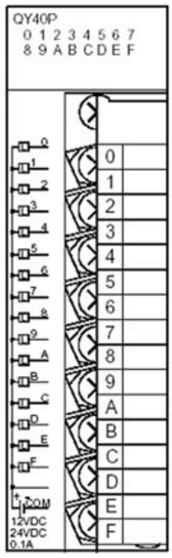
Source Transistor Output Circuit Configuration

Specifications \ Type		Transistor Output Module (Source Type) QY80	Appearance
Number of output points		16 points	
Isolation method		Photocoupler	
Rated load voltage		12-24VDC (+20/-15%)	
Maximum load current		0.5A/point, 4A/common	
Maximum inrush current		4A, 10ms or less	
Leakage current at OFF		0.1mA or less	
Maximum voltage drop at ON		0.2VDC (TYP.) 0.5A, 0.3VDC (MAX.) 0.5A	
Response time	OFF to ON	1ms or less	
	ON to OFF	1ms or less (rated load, resistive load)	
Surge suppressor		Zener diode	
Fuse		6.7A (unchangeable) (fuse blow capacity: 50A)	
Fuse blow indication		Yes (When fuse blows, LED indicates it and signal is output to CPU)	
External supply power	Voltage	12-24VDC (+20/-15%) (ripple ratio within 5%)	
	Current	20mA (at 24VDC)	
Dielectric withstand voltage		560VAC rms/3 cycles (altitude 2000m (6557.38ft.))	
Insulation resistance		10MΩ or more by insulation resistance tester	
Noise immunity		By noise simulator of 500Vp-p noise voltage, 1µs noise width and 25 to 60Hz noise frequency	
		First transient noise IEC61000-4-4: 1kV	
Protection of degree		IP2X	
Common terminal arrangement		16 points/common (common terminal: TB17)	
Number of I/O points		16 (I/O allocation is set as a 16-points output module)	
Operation indicator		ON indication (LED)	
External connections		18-point terminal block (M3×6 screws)	
Applicable wire size		0.3 to 0.75mm^2 core (2.8mm (0.11in.) OD max.)	
Applicable crimping terminal		R1.25-3 (sleeved crimping terminals cannot be used.)	
5VDC internal current consumption		80mA (TYP. all points ON)	
Weight		0.17kg	

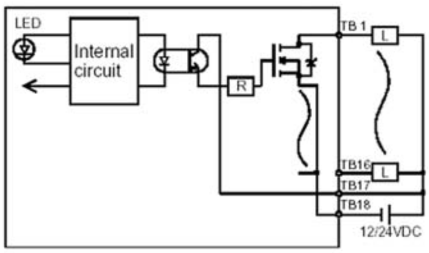
Output Circuit Detail

External Connections	Terminal Block Number	Signal Name
	TB1	Y00
	TB2	Y01
	TB3	Y02
	TB4	Y03
	TB5	Y04
	TB6	Y05
	TB7	Y06
	TB8	Y07
	TB9	Y08
	TB10	Y09
	TB11	Y0A
	TB12	Y0B
	TB13	Y0C
	TB14	Y0D
	TB15	Y0E
	TB16	Y0F
	TB17	COM
	TB18	0V

Sink Transistor Output Circuit Configuration

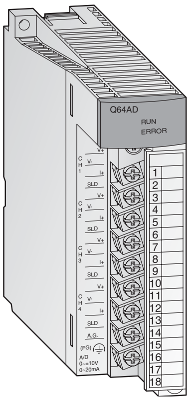
Specifications \ Type		Transistor Output Module (Sink Type)	
		QY40P	Appearance
Number of output points		16 points	
Isolation method		Photocoupler	
Rated load voltage		12-24VDC (+20/-15%)	
Maximum load current		0.1A/point, 1.6A/common	
Maximum inrush current		0.7A, 10ms or less	
Leakage current at OFF		0.1mA or less	
Maximum voltage drop at ON		0.1VDC (TYP.) 0.1A, 0.2VDC (MAX.) 0.1A	
Response time	OFF to ON	1ms or less	
	ON to OFF	1ms or less (rated load, resistive load)	
Surge suppressor		Zener diode	
Fuse		No	
External supply power	Voltage	12-24VDC (+20/-15%) (ripple ratio within 5%)	
	Current	10mA (at 24VDC) (Max. all points ON)	
Dielectric withstand voltage		560VAC rms/3 cycles (altitude 2000m (6557.38ft.))	
Insulation resistance		10MΩ or more by insulation resistance tester	
Noise immunity		By noise simulator of 500Vp-p noise voltage, 1μs noise width and 25 to 60Hz noise frequency	
		First transient noise IEC61000-4-4: 1kV	
Protection of degree		IP2X	
Common terminal arrangement		16 points/common (common terminal: TB18)	
Protection function		Yes (thermal protection, short circuit protection) • Thermal protection is activated in increments of 1 point. • Short circuit protection is activated in increments of 1 point.	
Operation indicator		ON indication (LED)	
External connections		18-point terminal block (M3×6 screws)	
Number of I/O points		16 (I/O allocation is set as a 16-points output module)	
Applicable wire size		0.3 to 0.75mm² core (2.8mm (0.11in.) OD max.)	
Applicable crimping terminal		R1.25-3 (sleeved crimping terminals cannot be used.)	
5VDC internal current consumption		65mA (TYP. all points ON)	
Weight		0.16kg	

Output Circuit Detail

External Connections	Terminal Block Number	Signal Name
	TB1	Y00
	TB2	Y01
	TB3	Y02
	TB4	Y03
	TB5	Y04
	TB6	Y05
	TB7	Y06
	TB8	Y07
	TB9	Y08
	TB10	Y09
	TB11	Y0A
	TB12	Y0B
	TB13	Y0C
	TB14	Y0D
	TB15	Y0E
	TB16	Y0F
	TB17	12/24VDC
	TB18	COM

2.9 Special Function Modules

2.9.1 Analog Input Modules

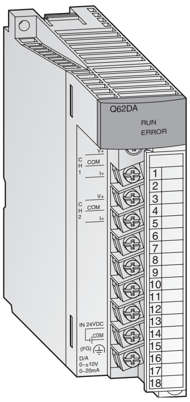


The analog input modules convert analog process signals into digital values which are further processed by the Q CPU. The A/D converter modules combine a high resolution (0.333 mV / 1.33 µA) with a high conversion speed (80 µs per channel).

All modules provide removable screw terminal blocks.

Analog input	Analog input range	Selectable input ranges	Input channels	
			4	8
Voltage	-10 to +10 V	1 to 5 V 0 to 5 V 0 to 10 V -10 to +10 V		Q68ADV
Current	0 to 20 mA	0 to 20 mA 4 to 20 mA		Q68ADI
Voltage or current (can be selected for each channel)	-10 to +10 V 0 to 20 mA	As for Q68ADV and Q68ADI	Q64AD	

2.9.2 Analog Output Modules



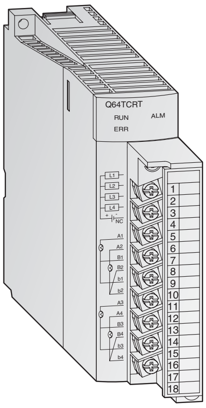
The analog output modules convert digital values into analog current or voltage signals. The resolution of 0.333 mV respectively 0.83 µA and the extremly short conversion time of 80 µs per output channel are only two of the many features of this modules. Isolation between process and control by means of optocouplers is also a standard feature.

All modules provide removable screw terminal blocks.

Analog output	Analog output range	Selectable output ranges	Output channels		
			2	4	8
Voltage or current (can be selected for each channel)	-10 to +10 V 0 to 20 mA	1 to 5 V -10 to +10 V 0 to 20 mA 4 to 20 mA	Q62DA	Q64DA	
Voltage	-10 to +10 V	-10 to +10 V			Q68DAV
Current	0 to 20 mA	0 to 20 mA 4 to 20 mA			Q68DAI

2.9.3 Temperature Control Modules with PID Algorithm

These modules enable PID algorithm temperatue control without placing any load on the Q CPU for the temperature control tasks.

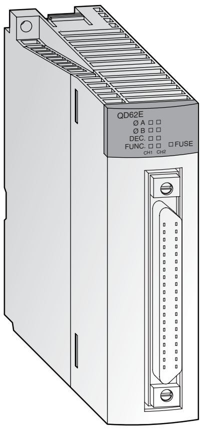


Special features

- 4 temperature input channels and 4 PID control circuits per module
- Input sensor types are either Pt100 temperature-measuring resistors (Q64TCRT and Q64TCRTBW) or thermocouples (Q64TCTT and Q64TCTTBW)
- The modules 64TCRTBW and Q64TCTTBW can detect the disconnection of a heater
- Auto tuning function for the PID control circuits
- Transistor output to drive the actuator in the control circuit

2.9.4 High -Speed Counter Modules

The modules QD62E, QD62, and QD62D detect signals at a frequency too high for normal input modules.



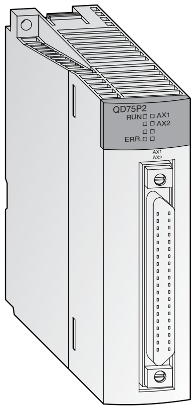
Special features

- Maximum counting frequency up to 500 kHz
- Input for incremental shaft encoder with automatic forward and backward detection
- Preset and selection of counter function via external digital inputs
- 32-bit counting range(-2 147 483 648 to +2 147 483 647)
- Can be used as up, down or ring counter
- All modules offer two counter inputs
- Two digital outputs which are set according to the counter value per counter input

All modules are connected via a plug.

2.9.5 Positioning Modules

In combination with stepper motors or servo amplifiers the modules QD75P1, QD75P2, and QD75P4 can be used for speed or position control.

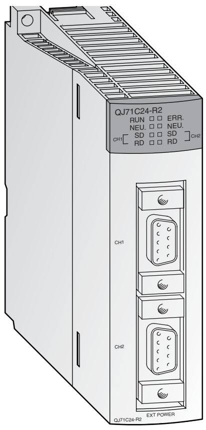


Special features:

- Control of up to four axes with linear interpolation(QD75P4) or two axes with circular interpolation (QD75P2 and QD75P4)
- Storage of up to 600 positional data sets in flash ROM
- Units of travel can be definedin pulses, µm, inches or degrees.
- Configuration and presetting of positional data is carried out by means of the PLC program or with the aid of the Microsoft Windows [TM] software GX Configurator QP.

2.9.6 Serial Communication Modules

The modules QJ71C24 and QJ71C24-R2 enable communications with peripheral devices via a standard serial interface.

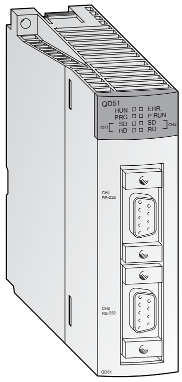


Special features:

- Two RS232C interfaces (QJ71C24-R2) or one RS422/485 and one RS232C interface (QJ71C24)
- Transmission speed up to 115200 bit/s
- Enables PCs connected to the PLC to access the full data set of the Q CPU
- Options for connection of a printer
- Integrated flash ROM memory for logging quality, productivity, or alarm data that can be transmitted when required.
- Support for plain ASCII data exchange. A user frame can be defined
- PLC programming and monitoring through the serial communication line is supported.

2.9.7 Intelligent Communication Modules

The modules QD51S-R24 and QD51 work through their own program(written in BASIC) independently of the Q CPU. Thus, data can be processed and communications can be performed with peripheral devices without imposing an additional load on the PLC CPU.



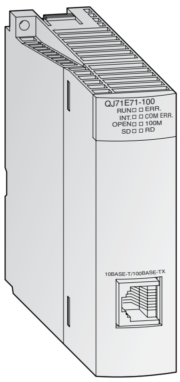
Special features:

- Either two RS232 interfaces (QD51) or one RS422/485 and one RS232 interface (QD51S-R24)
- Transmission speed of up to 38400 bit/s
- Access to devices in the Q CPU and tothe buffer memory of intelligent function modules is supported
- Remote RUN/STOP is supported via the serial communication line

2.9.8 ETHERNET Interface Modules

the modules QJ71E71/E71-100 and QD71E71-B2 are used on the PLC side to connect a host system, e.g. a PC or work station and the System Q via ETHERNET.

Besides the data transfer via TCP/IP or UDP/IP communications the reading and changing of PLC data as well as the monitoring of CPU module operation and control status is supported.

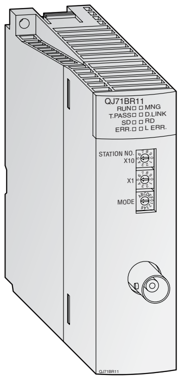


Special features

- Network types: 10BASE5, 10BASE2 or 10BASE-T
- Transfer rate of 10/100Mbit/s
- FTP-server functionality
- The communication function using fixed send and receive buffers is available.
- Up to 16 communication lines can be opened for concurrent data communication.
- PLC programming and monitoring can be performed fromGX Developer or GX IEC Developer on a personal computer via ETHERNET.

2.9.9 MELSECNET Modules

The modules QJ71BR11 and QJ71LP21 are used to connect the System Q to a MELSECNET/10 or MELSECNET/H network. This enables fast and effective communications between PLCs of the Q, QnA and QnAS series.

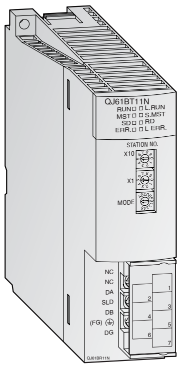


Special features

- Two different topologies are featured: Coaxial bus (QJ71BR11) or redundant optical loop (QJ71LP21).
- High data transfer rates: 10 Mbit/s with coaxial bussystems and optional 10 or 20 Mbit/s with optical loop systems
- Communications with other PLCs, PCs, or remote I/O
- The network system supports data communications between any two stations, no matter how many networks lie between them
- Station separating functionin coaxial bus system and loop back function in optical duplex loop systems in case of a station malfunction
- Control station shifting function and automatic return function

2.9.10 Master/Local Module for CC-Link

The QJ61BT11 is applicable as a master or local station in a CC-Link system and manages the connection of remote inputs and outputs.

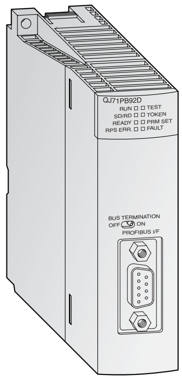


Special features

- The parameters of all modules across the network are set directly via the master module.
- The communications between the remote modules and the master module is performed automatically. The refresh time for 2048 I/O points is 3.3 ms only.
- Transmission speed of up to 10 Mbit/s
- With one master module a system can be extended to up to 2048 remote I/O points.
- An additional stand-by master establishes a duplex system. When an error occurs in the master station the datalink will be continued.
- Automatic CC-Link start without parameter setting
- Interrupt program start via network data command

2.9.11 PROFIBUS-DP Interface Module

The QJ71PB92D PROFIBUS-DP master module and the QJ71PB93D PROFIBUS-DP slave module enables PLCs of the System Q to communicate with other PROFIBUS devices.

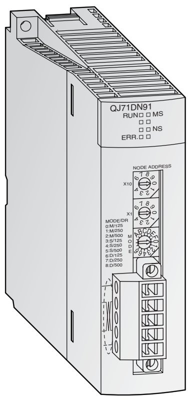


Special features

- The master station can communicate with up to 60 slave units.
- Up to 244 input bytes and 244 output bytes can be processed at a time per slave station.
- Supported functions include SYNC, FREEZE, and specialized diagnostic messages for the specific slave types used.
- Data excange with automatic refresh is supported . Batch transfer can be chosen as an option.

2.9.12 DeviceNet Module

The QJ71DN91 connects a Q series PLC with the DeviceNet. DeviceNet represents a cost-effective solution for the network integration of low-level terminal equipment.

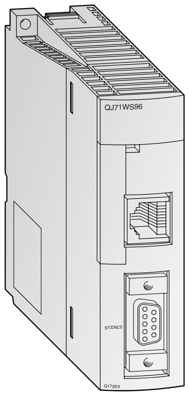


Special features

- The positions of master and slave stations are user selectable.
- Transfer rates of 125, 250 and 500 kBaud
- Transmission distances of up to 500 m
- Communication methods
  – Polling
  – Bit strobe
  – Change of state
  – Cyclic

2.9.13 Web Server Module

The web server module QJ71WS96 enables the remote control monitoring of a Q series PLC.



Special features

- Access to the PLC via the Internet
- Very easy setting functions integrated
- User needs only a Web browser for setting and monitoring
- RS232 interface for modem connection
- Various connections for data exchange are possible: ADSL, modem, LAN, etc.
- Sending and receiving data via mail or FTP
- Integration of a self-designed web site and Java applets is possible
- Standard connection via ETHERNET to exchange data between other PLCs or PCs
- Events and CPU data logging functions

2.10 Operation of a PLC

2.10.1 Programming Software

To be able to design a PLC program using a computer, it is essential for the software to have the following facilities:

- Programs can be designed using recognised and understandable conventions i.e. Relay Ladder diagrams and Instruction List formats.
- The functional integrity of programs may be tested prior to use on the chosen PLC.
- Programs can be permanently saved either on a computer's hard disk, or on removable media.
- Programs can be re- loaded from either the hard disk or the removable media.
- Ladder diagrams may be fully annotated.
- Hard copy print-outs can be obtained.
- The program can be transferred to and from the PLC, via a serial link.
- The Program operation can be monitored in 'real time'.
- Modifications can take place, whilst the PLC is On-line.
- Operational Parameters may be altered.
- Data memory areas may be saved and retrieved.
- Programs may be simulated on a PLC software emulator.

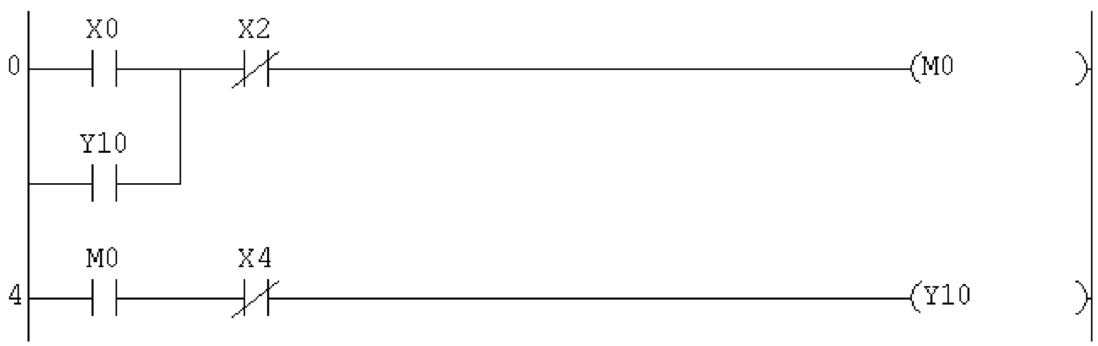
To name but a few!

2.10.2 Basic Operation of the Q-Series PLC System

Devices

PLC's like all computer systems, posses an internal structure. This could be described as a map of locations within the system. Every device in the system has a unique location called an Address. In the Mitsubishi Q-Series range of PLC's this is divided into numerous 'Device Names'.

To explain the basic operation of a PLC system, consider the following 2 networks of Ladder program.



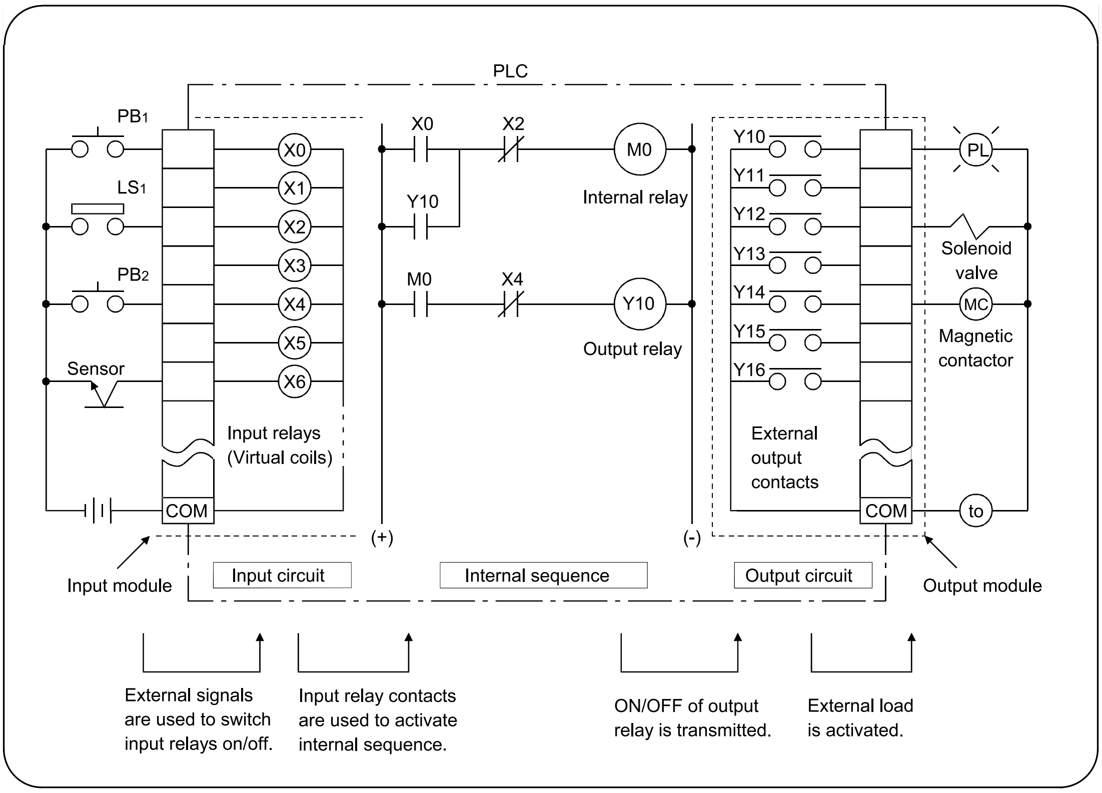
- Network1

  When Input X0 closes, providing X2 has not operated, this drives Internal Memory Coil M0. Y10 is in parallel with input X0 (MELSEC IL – “LD X0 OR Y10”). The condition of Y10 is dependant on the output of Y10 which is driven from Network 2 as described below:

- Network 2

  When the normally open contact of M0 closes and the normally closed contact of X4 has not operated, output Y10 becomes energised. Hence the circuit latches depending on the conditions set in Network1.

Based on the above circuit example, the following diagram helps to illustrate the operation of an Input/Output refresh cycle of a PLC system:



Principle of Operation

As can be seen from this illustration, the I/O PLC refresh cycle can be divided into three primary processes: **Input Processing, Program Processing and Output Processing.**

- Input Processing

  The Programmable Controller (PLC) initially reads the ON/OFF condition of all of the Inputs used in the program. These conditions are then stored into the Input Image Memory.

- Program Processing

  – The PLC then starts at the beginning of the PLC program and for each element of the program; it READS the actual logic state of that element, which is stored in either the Input Image Memory or the Output Image Memory.

  – If the required logic state is correct i.e. X0 is ON and X2 is OFF, the PLC will move on to the next element in the rung, i.e. M0.

- If M0 is ON, then logic 1 will be WRITTEN into the Output Image Memory in the location reserved for M0.
- If X0 is OFF, then logic 0 is WRITTEN into the M0 memory location.
- After an output instruction has been processed, the first element on the next line is executed, which in this example is a normally open contact of M0.
- Hence the logic state of the M0 memory location is this time READ from and if its logic state is at logic 1 indicating that the M0 coil is energised, this effectively means all M0 normally open contacts will now close. When the contact of M0 is closed and X4 open, a Logic 1 will be WRITTEN to the Image Memory Location reserved for the Output Y10.
- However if the contents of the M0 memory location are at logic 0, i.e. M0 is not energised, then a Logic 0 is WRITTEN to the Y10 Memory Location

● Output Processing

Upon completion of the execution of all instructions, the contents of the Y memory locations within the Output Image Memory are now transferred to the Output Latch Memory and the Output Terminals.

Hence any output, which is designated to be ON, i.e. Y10, will become energised.

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.

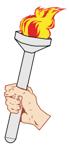
This system enables structured programs to be created using a high degree of modularisation. This provides increased efficiency, where tested programs and routines may be reused with a reduction of the number of programming errors.

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

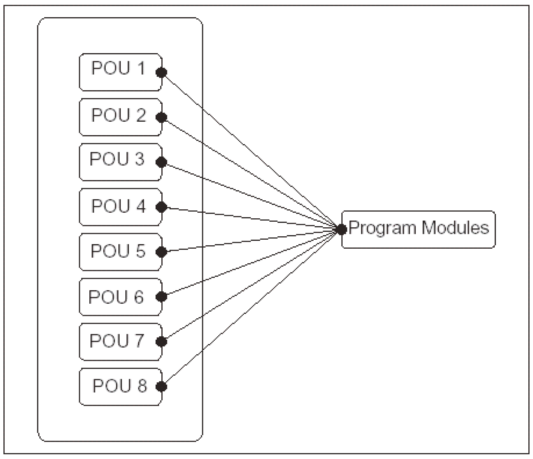
3.2.1 Definition of Terms in IEC61131-3

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:

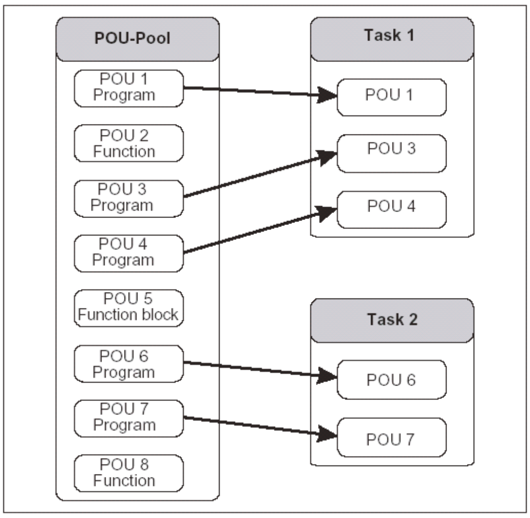
- Programs

- Functions
- Function Blocks

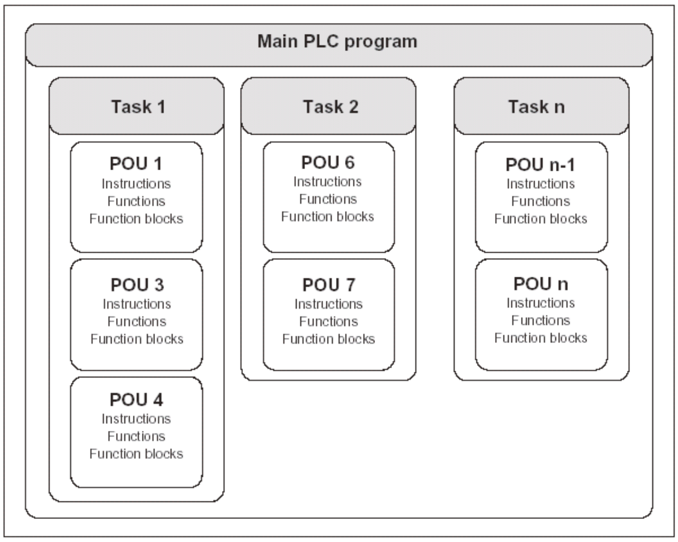
POU's declared as Function Blocks can be considered as **programming instructions in their own right** and they can be used as such in every module of your programs.

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

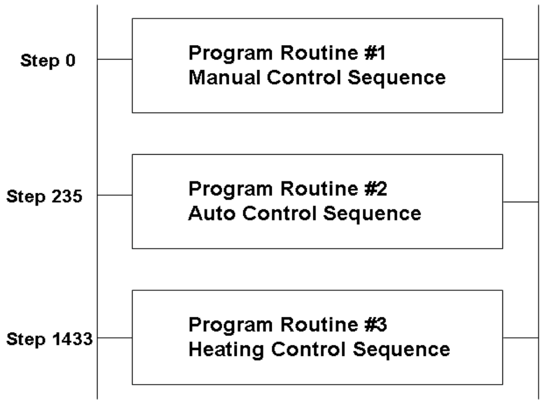


The Program POU's are grouped together in tasks



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

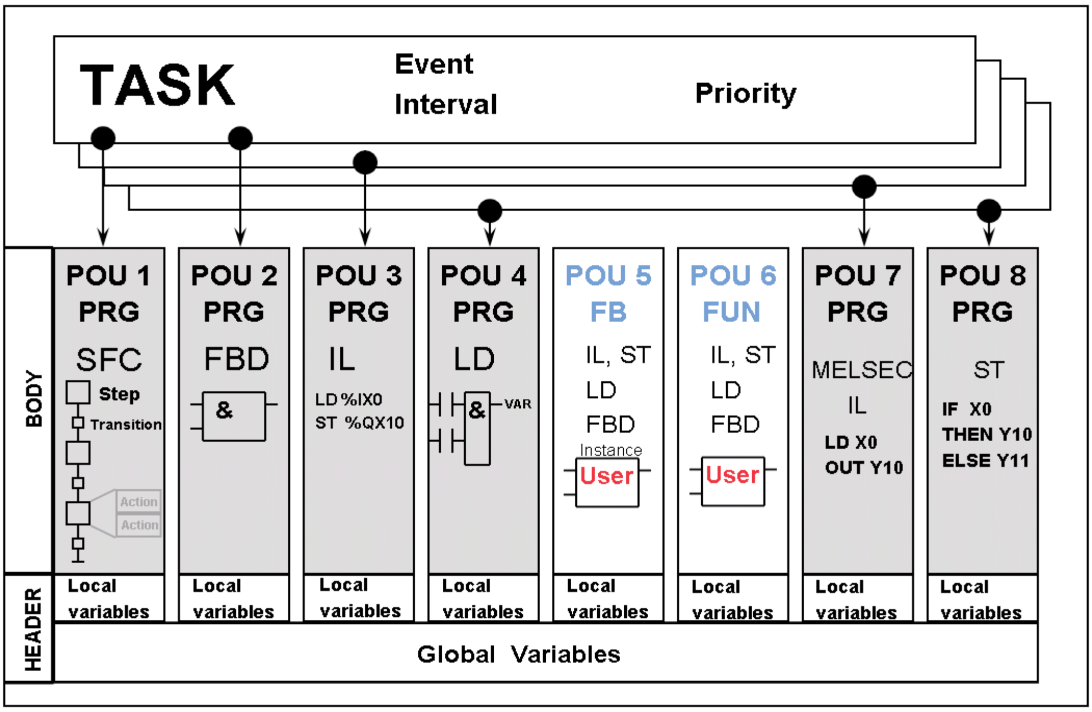
Most PLC programs consist of areas of code which perform specific tasks.They may form part of one large program, or be written in sub-routines, with program control instructions to select the current routine i. e. CALL, CJ etc.



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:



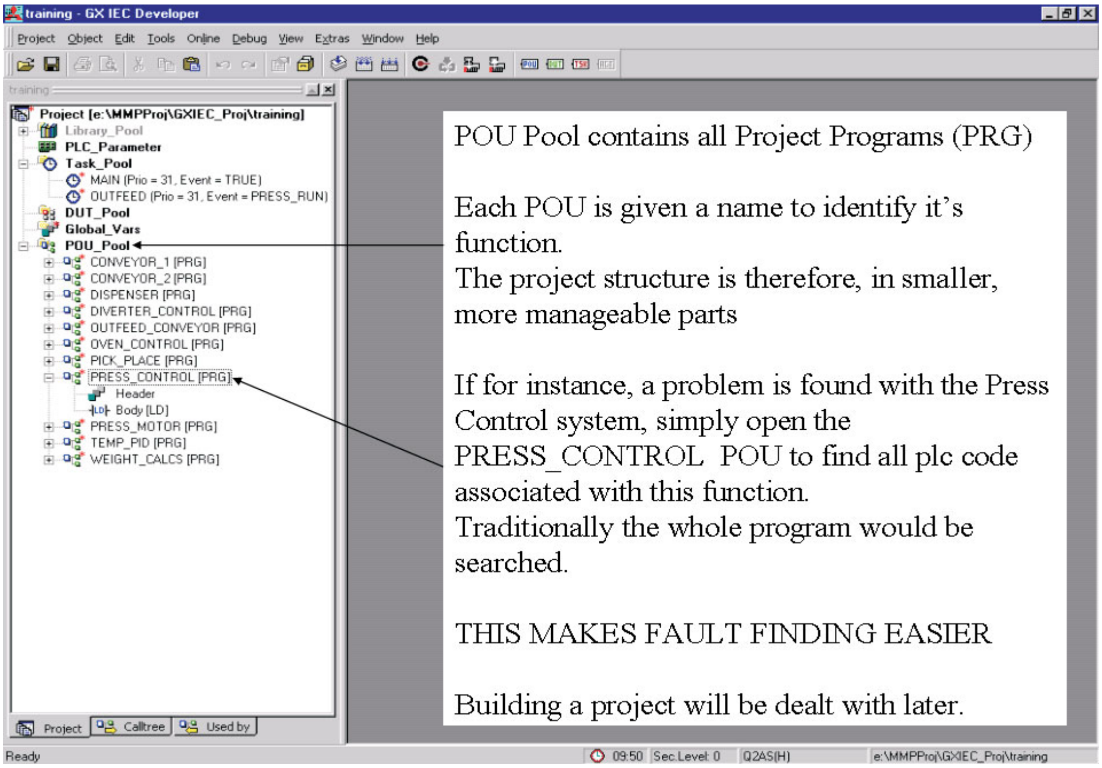
Overall Project Configuration illustrating POU integration using SFC, FBD, IL, LD and MELSEC IL and ST format programs.

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.

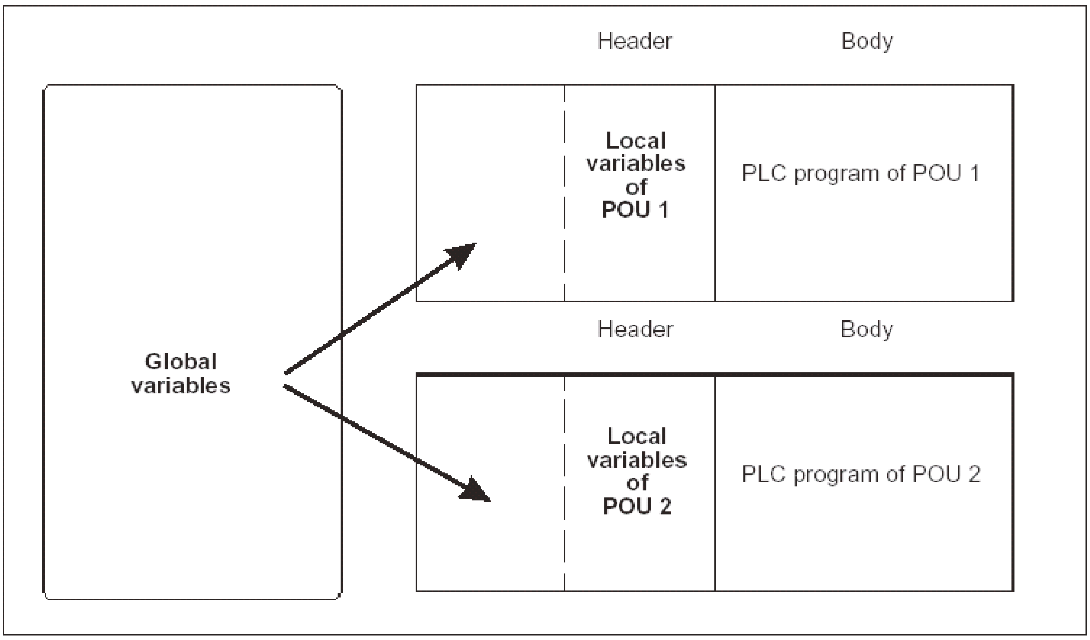
In this way, perhaps complicated interlocking routines, could be written in a ladder POU, whilst complex calculations or algorithms, might be better suited to one of the textual, or FDB editors.

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



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.

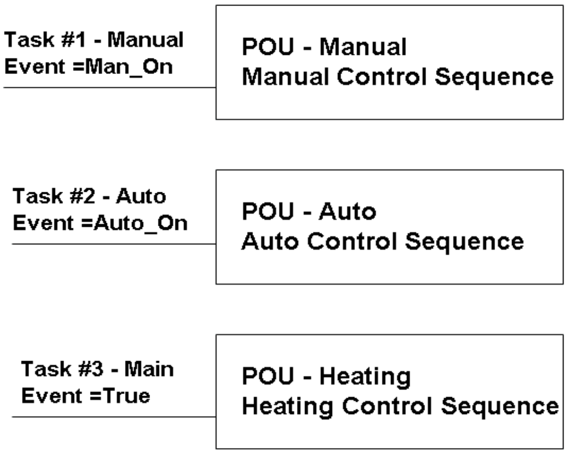
Task Pool and Task Manager

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.

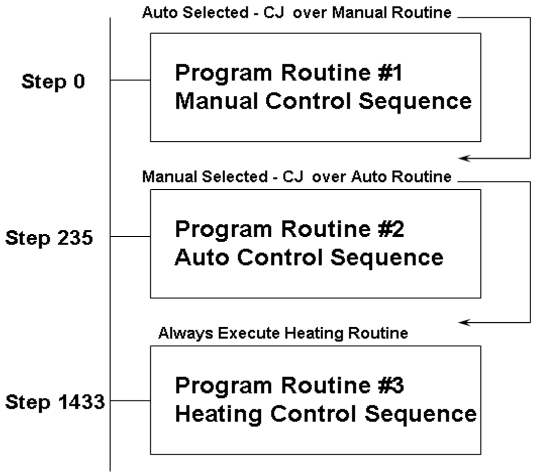
Each Task can have different operating conditions, or events.

- 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)

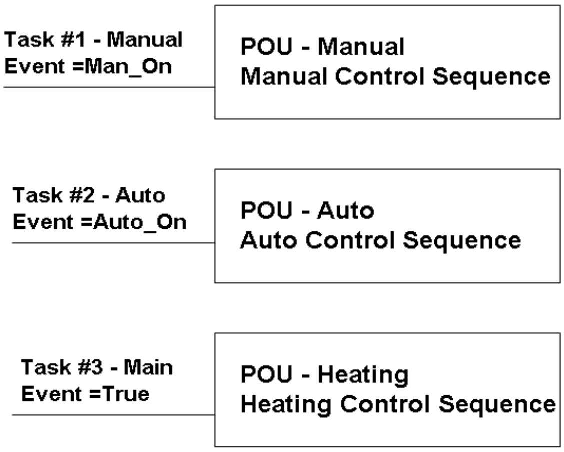
These tag names would be declared as Global Variables and assigned to PLC bit devices (they could be addresses i.e. X0).



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.



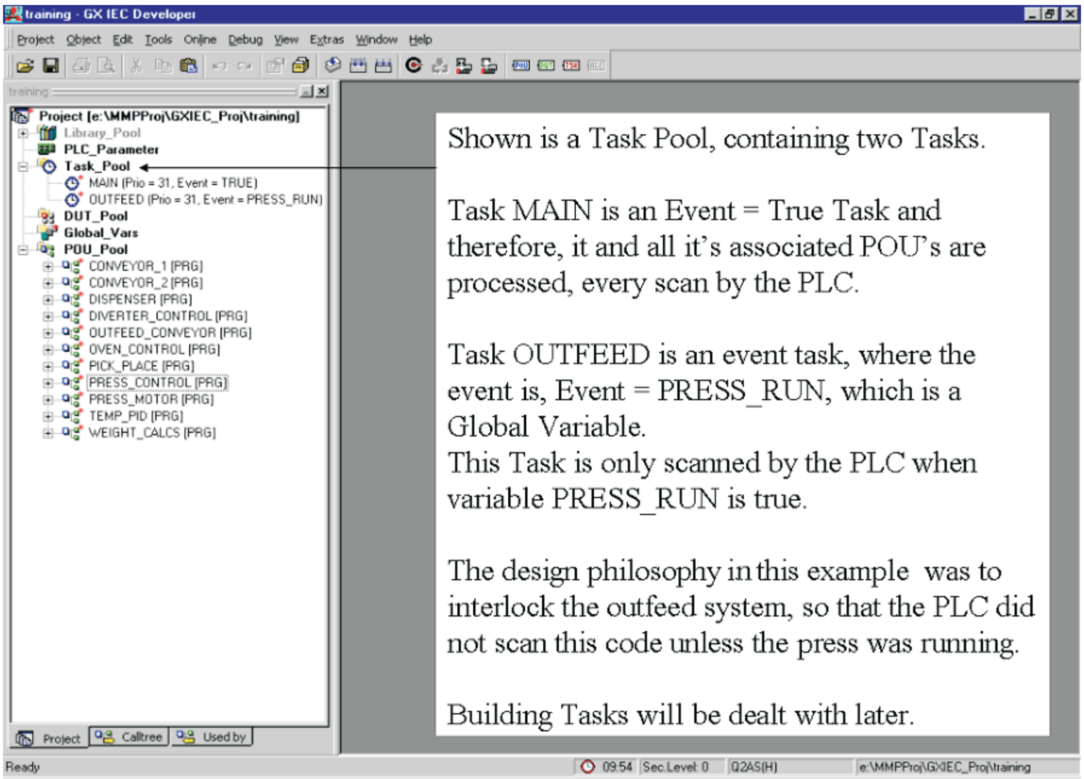
When GX IEC Developer compiles the project, it automatically inserts, program branching instructions, into the program, in line with event driven tasks.

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.



The **Task Manager** 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.

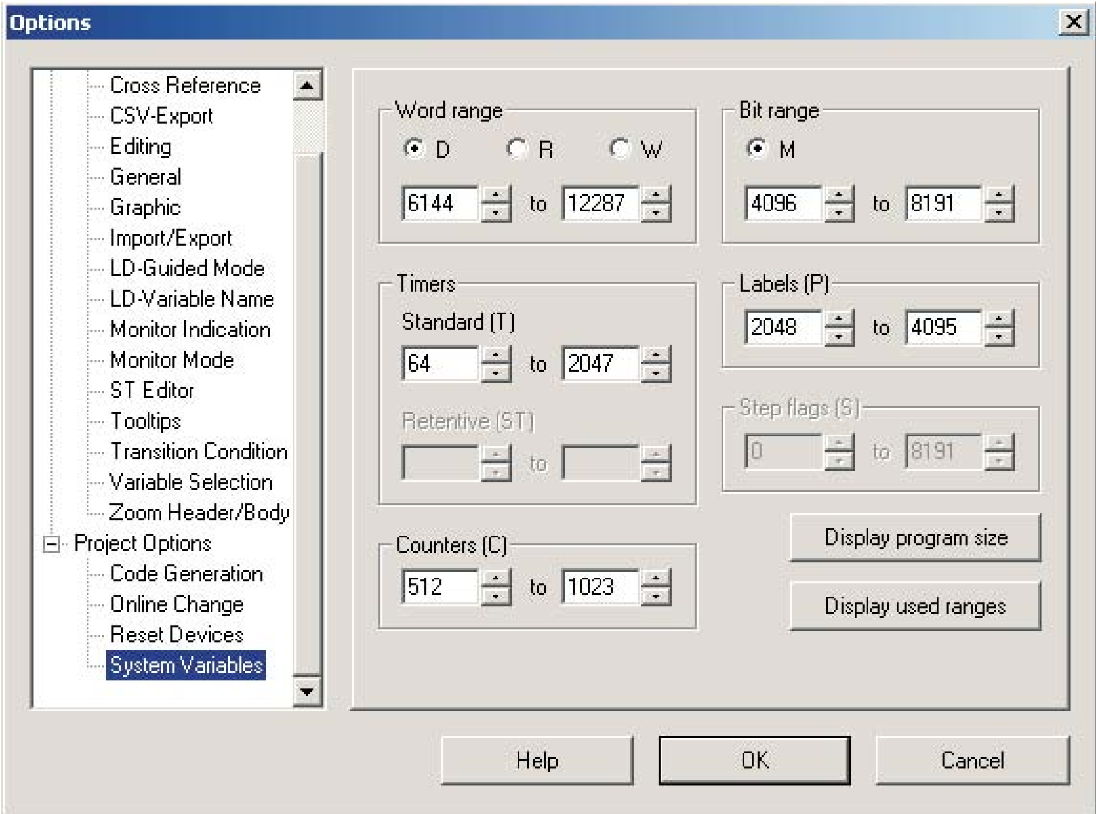
The software engineer need only be concerned about the program content, not whether the branch instructions are correct and obey the rules.

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:



Systen variable ranges for the actual project. Available if an Q/QnA project is open.

- 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 dialog box. If the program is not compiled the dialog 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 a separate dialog box.

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.)

Used System Devices

To read GX IEC Developer’s device allocation to system variables usage, the ***Display used ranges*** button should be clicked and the following notification will be displayed:

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 16.

- 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 CJ LD POU	X0 P_20 X1 Y0
P_20 MELSEC	LD OUT	X2 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*)

Y20:=X10;
Y21:=X11 AND X12 OR X13;
M0:=(M1 AND (M2 OR M3)) OR X14;
```

Example for Structured Text

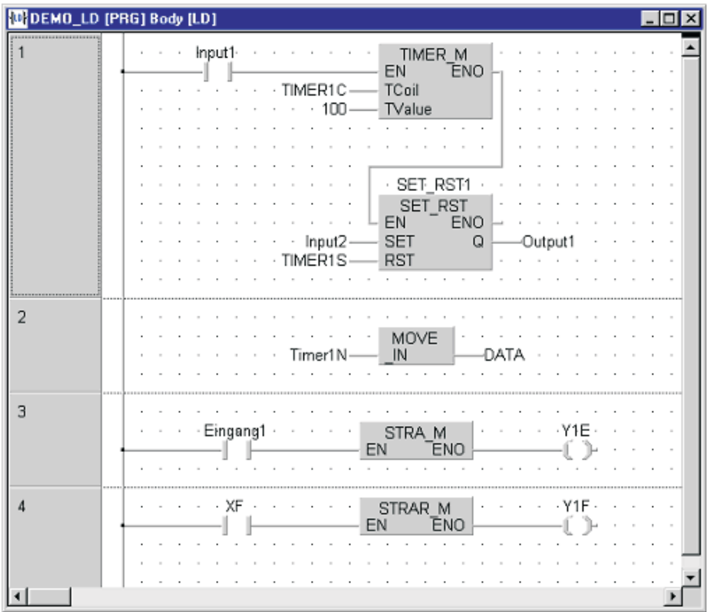
An example of Structured Text programming is given in chapter 17.

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.



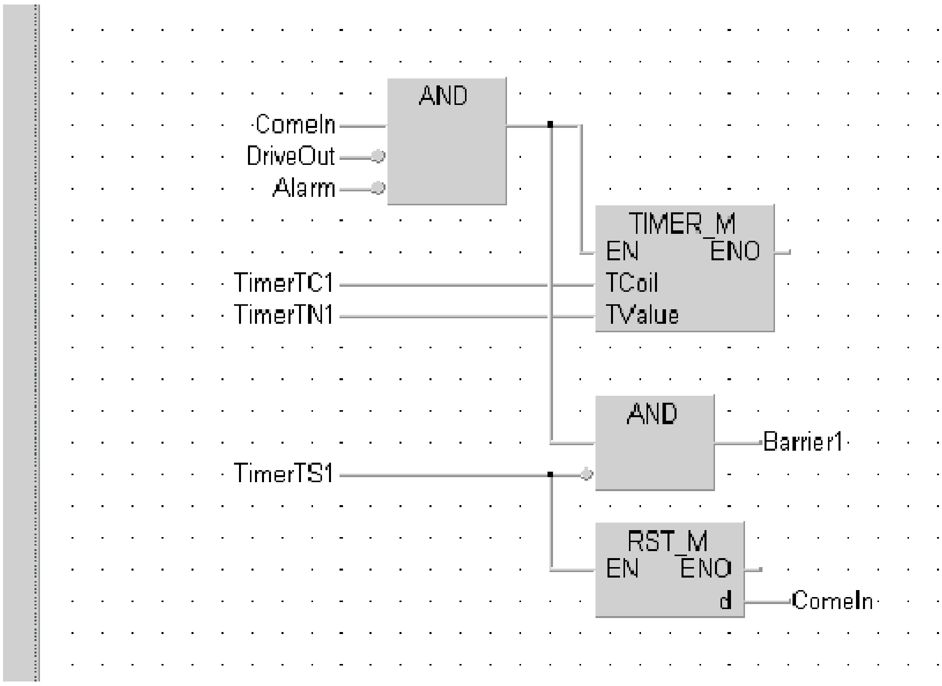
Example for Ladder diagram

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:

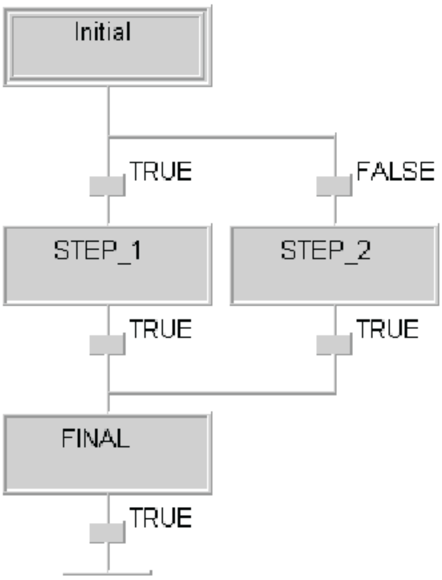


Sequential Function Chart

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

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

* Note: Excluding some early version Q00J CPU’s

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

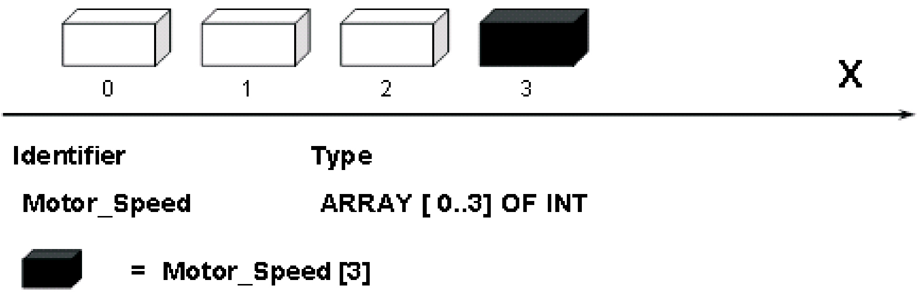
In software, program elements can use: Motor_Volts[1] and Motor_Volts[2], as declarations, which in this example mean that D1 and D2 are addressed.

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.

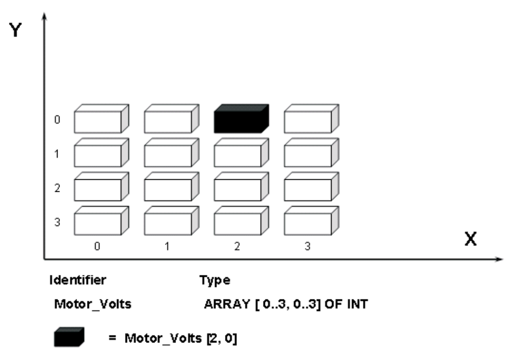
Arrays can provide a convenient way of 'indexing' tag names, i.e. one declaration in the Local or Global Variable Table can access many elements.

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

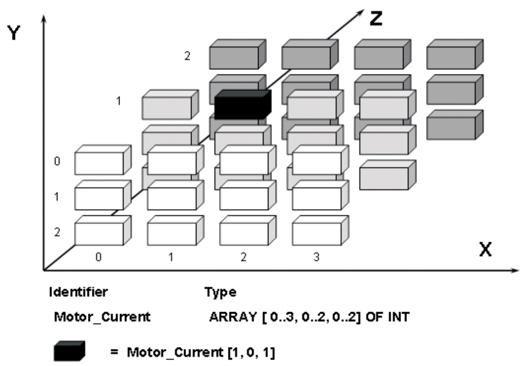
Single Dimensional Array



Two Dimensional Array



Three Dimensional Array



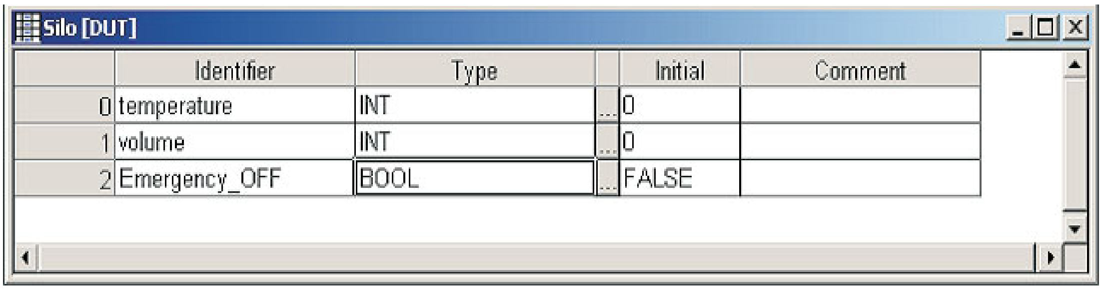
Data Unit Types (DUT)

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.

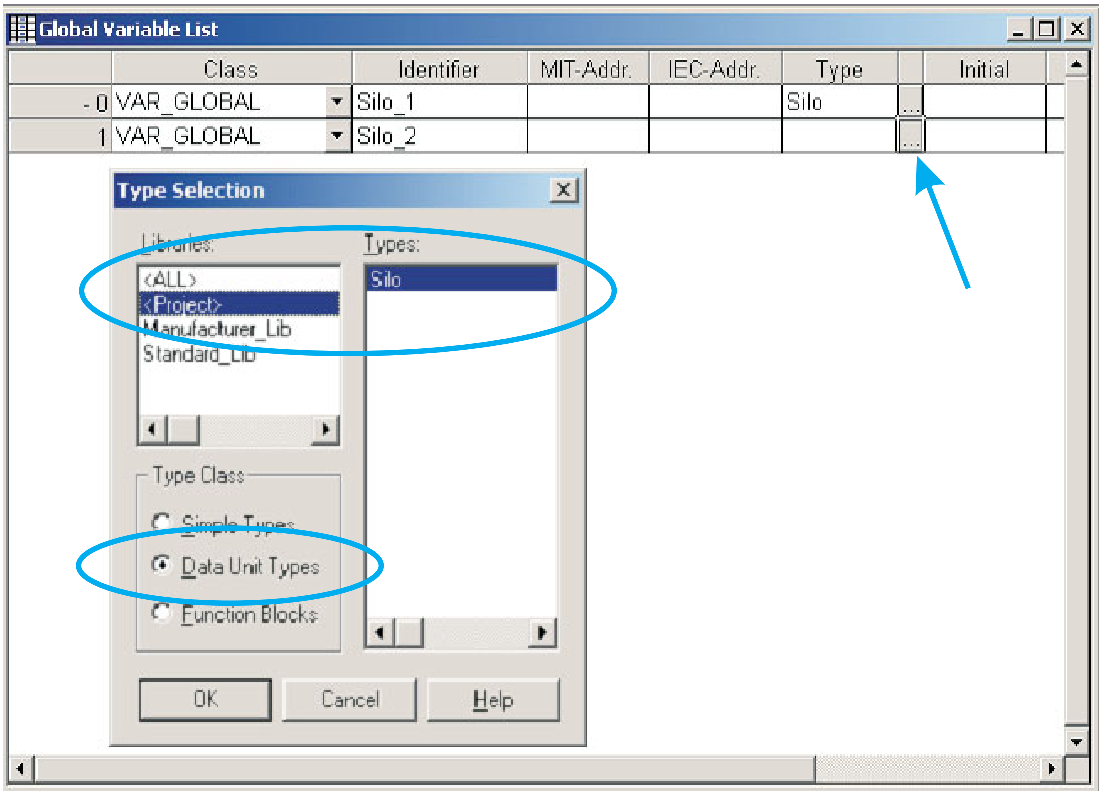
Example use of a DUT

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.

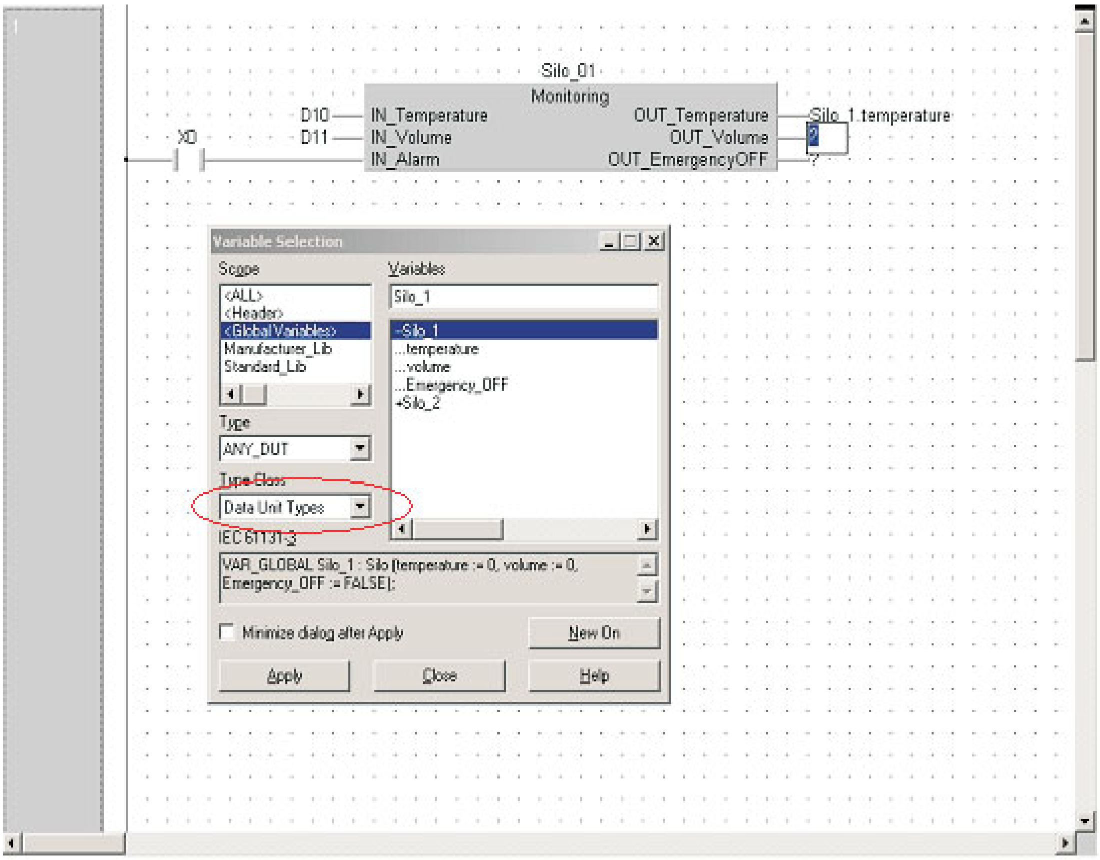


How to declare the DUT

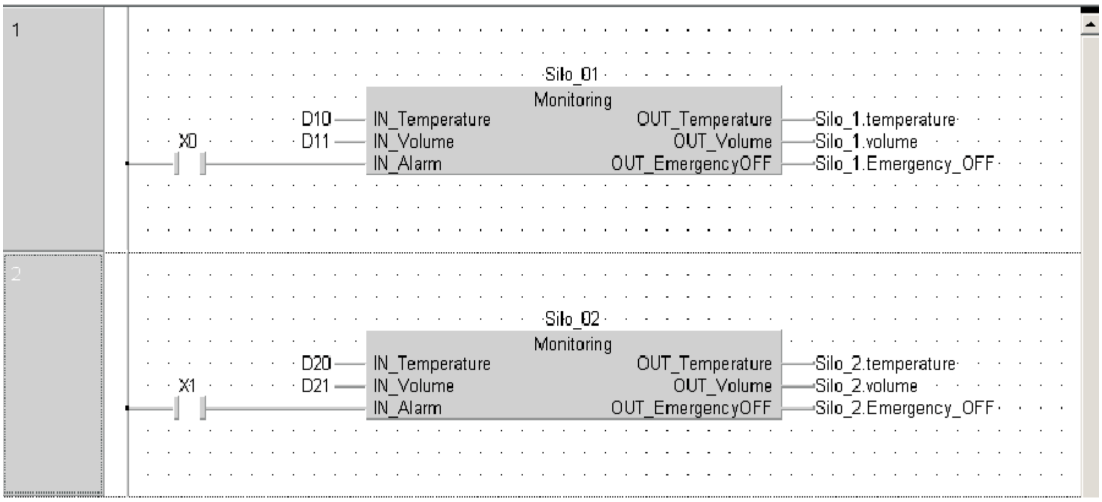
Double-click on ***Global_Vars*** in the Project Navigator window and enter the following lines in the global variables declaration table.



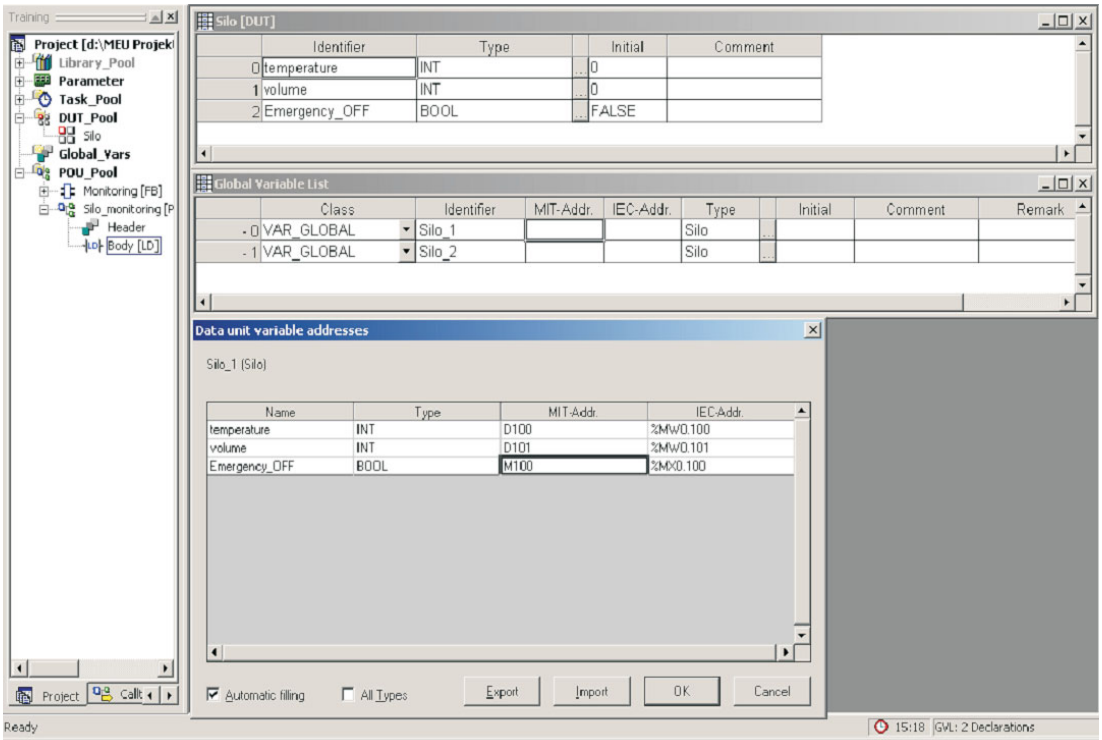
The variables are stored in the Global Variable List. The structure of both variable, Silo_1 and Silo_2, is identical, so to reference the individual variable of each DUT you only need to prefix their names with the name of the respective global variable.



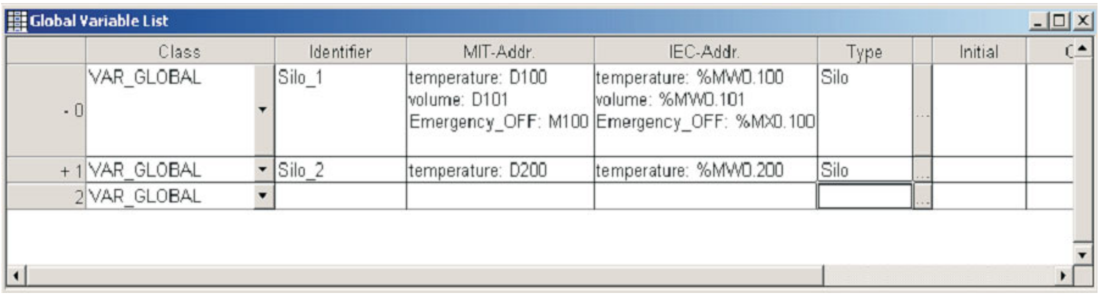
In this example a function block of the type “Monitoring” has been programmed for assigning the register value and the Boolean input to the elements of the DUTs. Two separate instances (Silo_01 and Silo_02) of this function blocks were then created for two silos.



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



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.

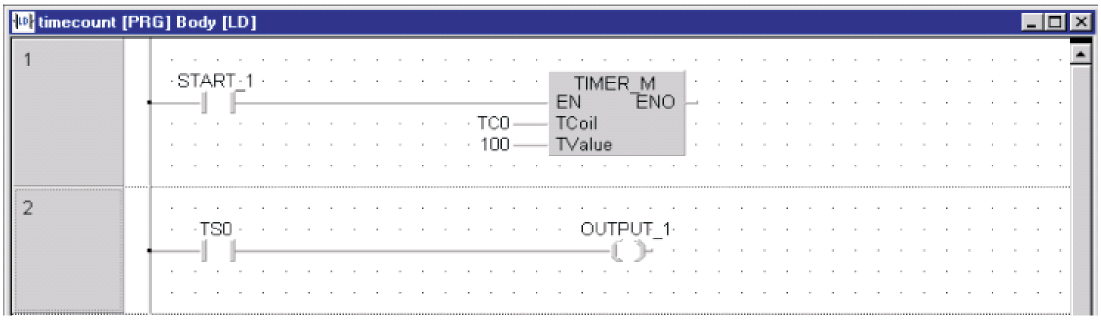


3.4.3 MELSEC Timers and Counters

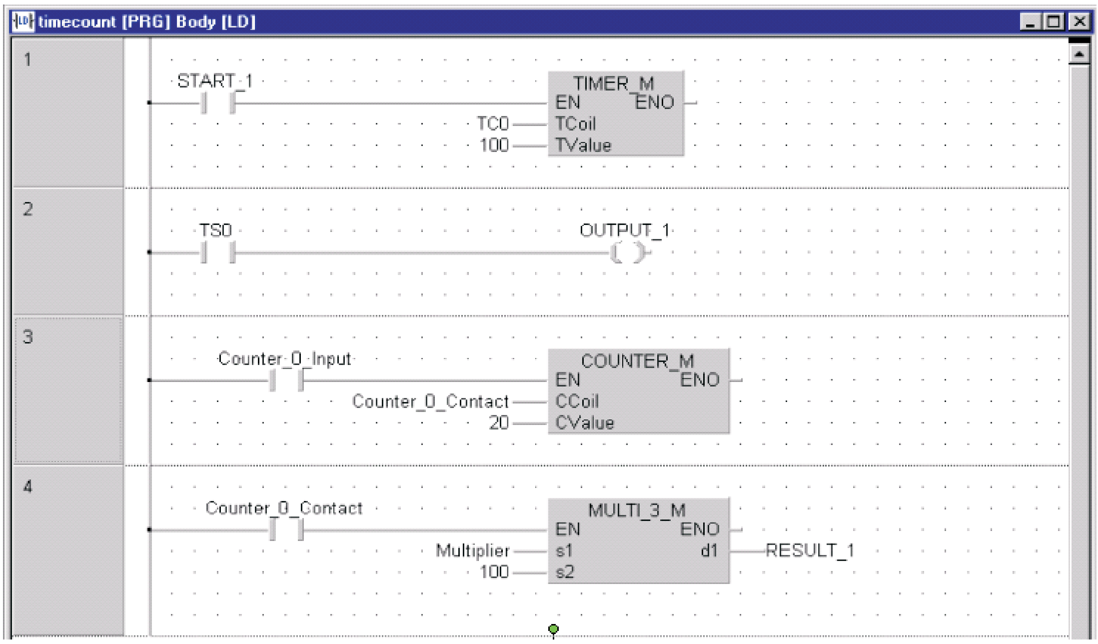
When programming standard Timers/Counters, an IEC convention must be observed:

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

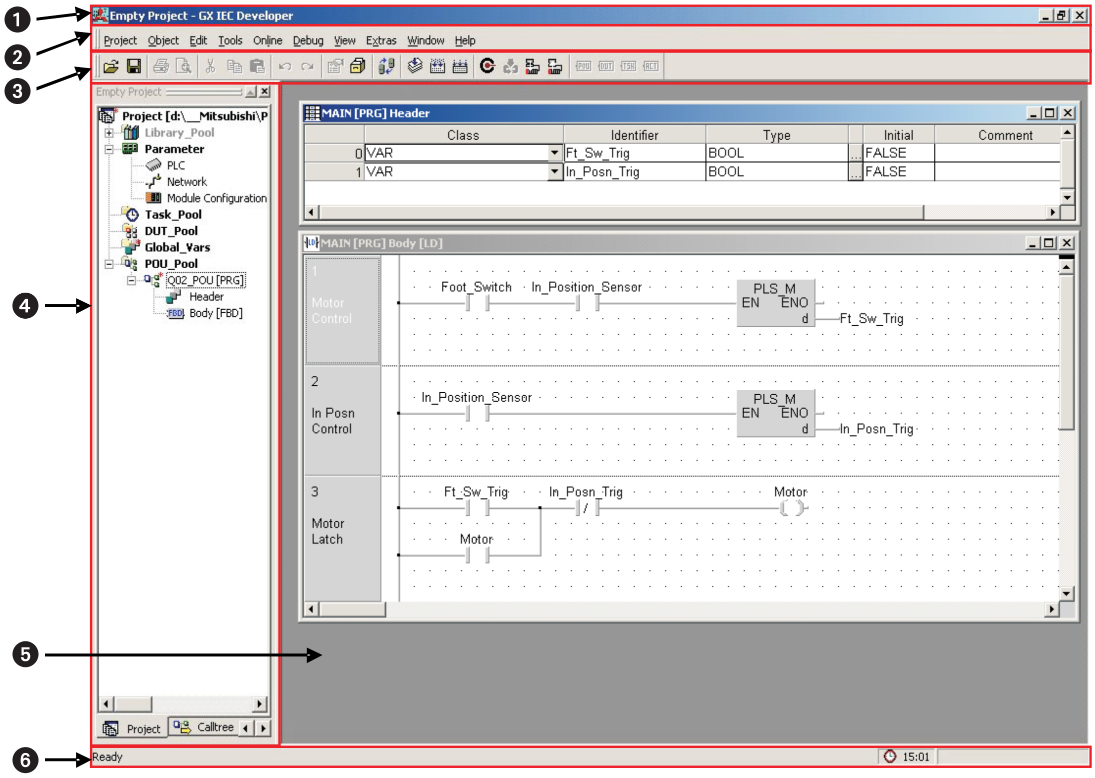
In the next section, we will build our first project, initially using the Ladder Diagram editor.

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:



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

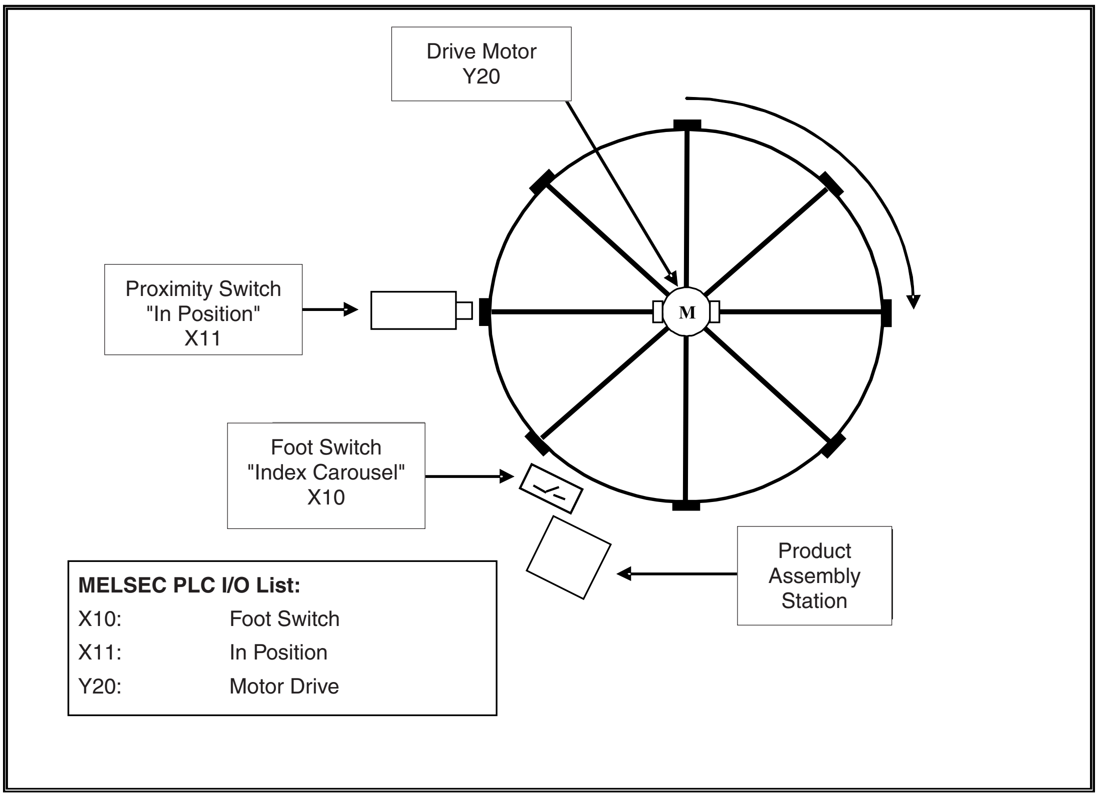
① Momentarily operate foot switch to Index Carousel.

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.)



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:

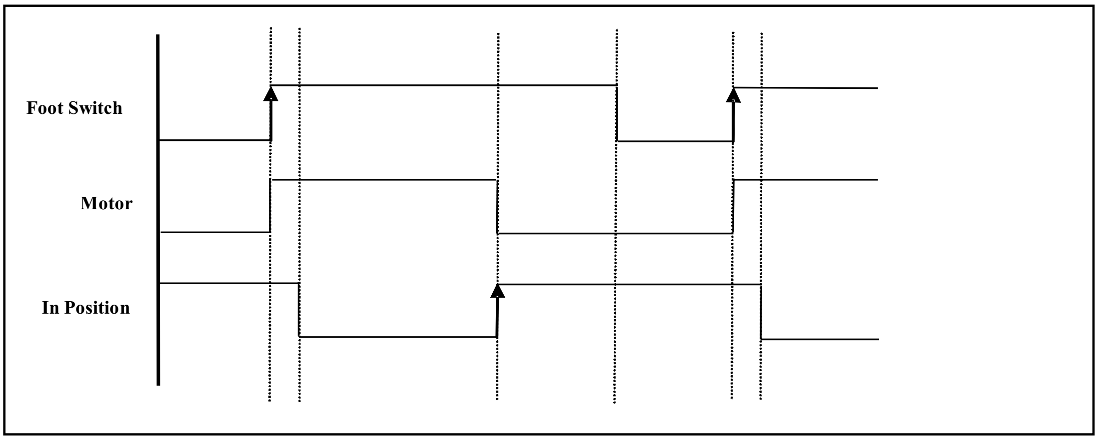
- The foot switch may be operated at random. Once activated, it may be possible for the operator to forget to release the switch which may cause the table to continue to rotate past its index position.
- Once "In-Position" X11 operates, it remains on, thus the table is prevented from re-indexing.

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.

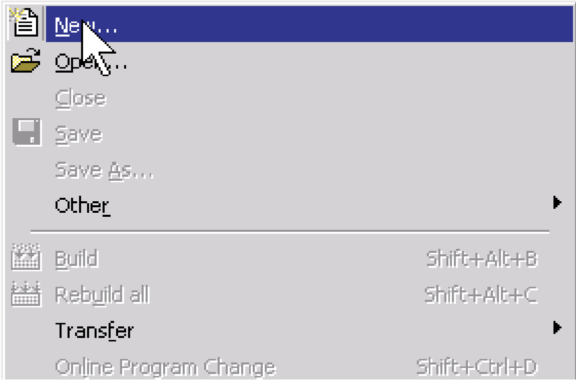
The following diagram illustrates the order of sequencing of the Carousel control. Note that the rising edge of the foot switch triggers the motor ON, irrespective of the “In Position” sensor being ON. When the table begins rotating, the “In position” sensor turns OFF a little later. The motor continues 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:

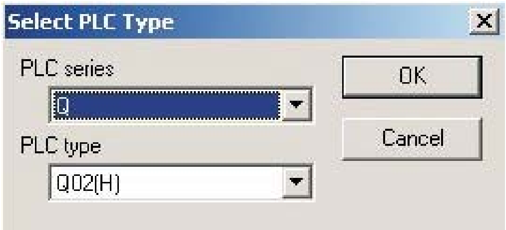


4.2.2 Creating a New Project

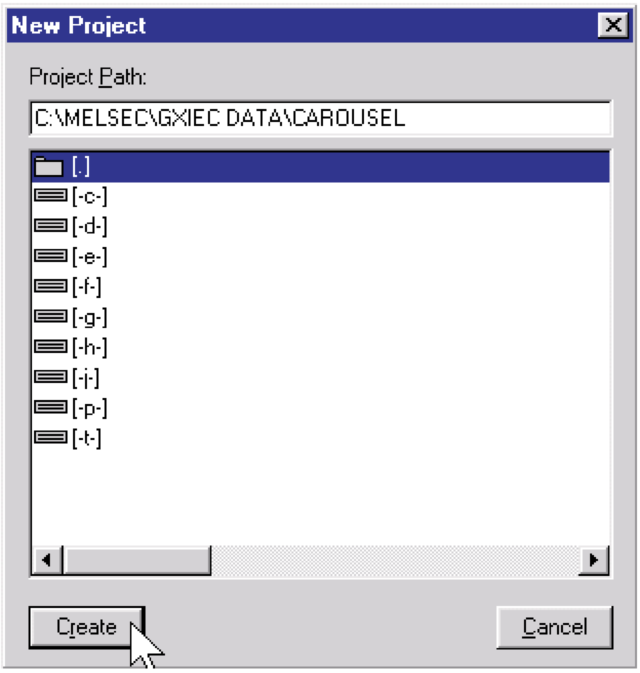
① From the ***Project*** menu, select ***New***.



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

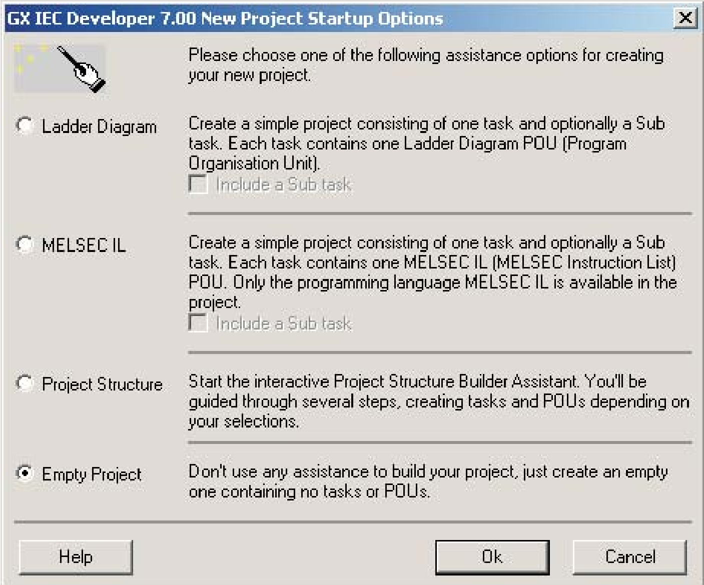


③ Provide a name for the project in the project path field. In this case use “\GX-IEC DATA\CAROUSEL” and click on ***Create*** – as in the following illustration:



The Wizard

The Project Startup Wizard will be displayed:

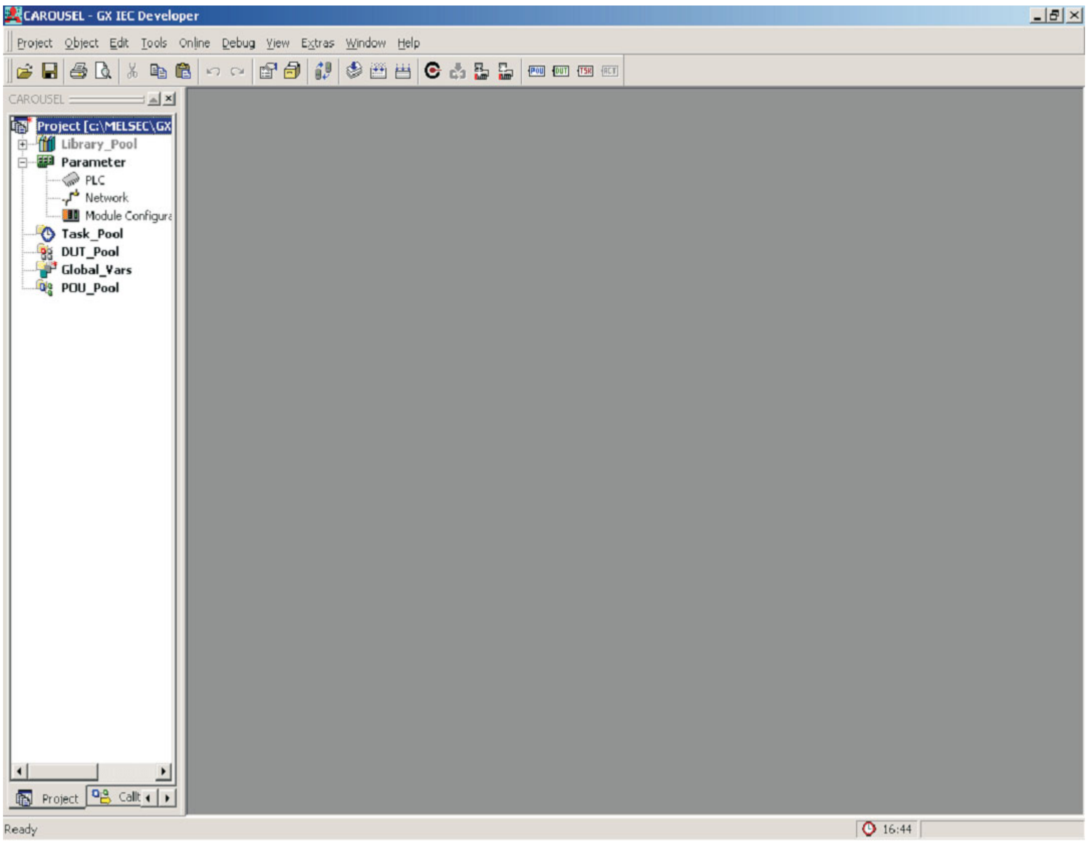


The Wizard provides a quick way to begin projects. It will thus create the basic starting structures for simple projects.

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:

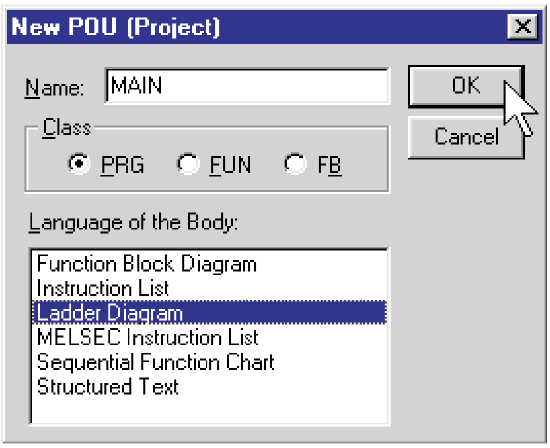


This is the primary display of the project.

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.

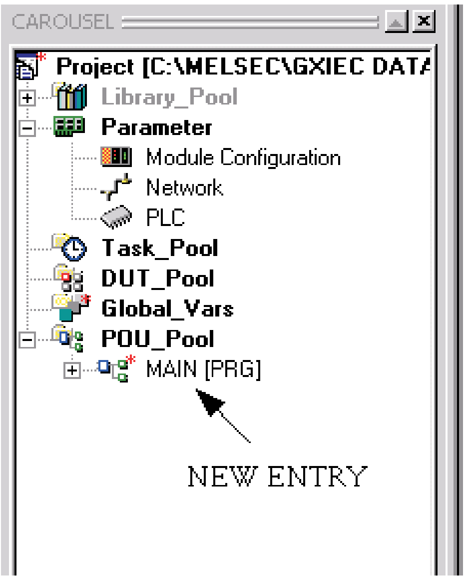
4.2.3 Creating a new "POU"

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:

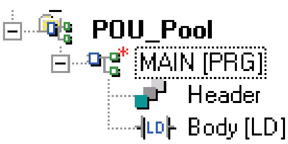


The name of the POU will be 'MAIN' and it should be specified as a **Ladder Diagram** of type ***PRG*** (Program).

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



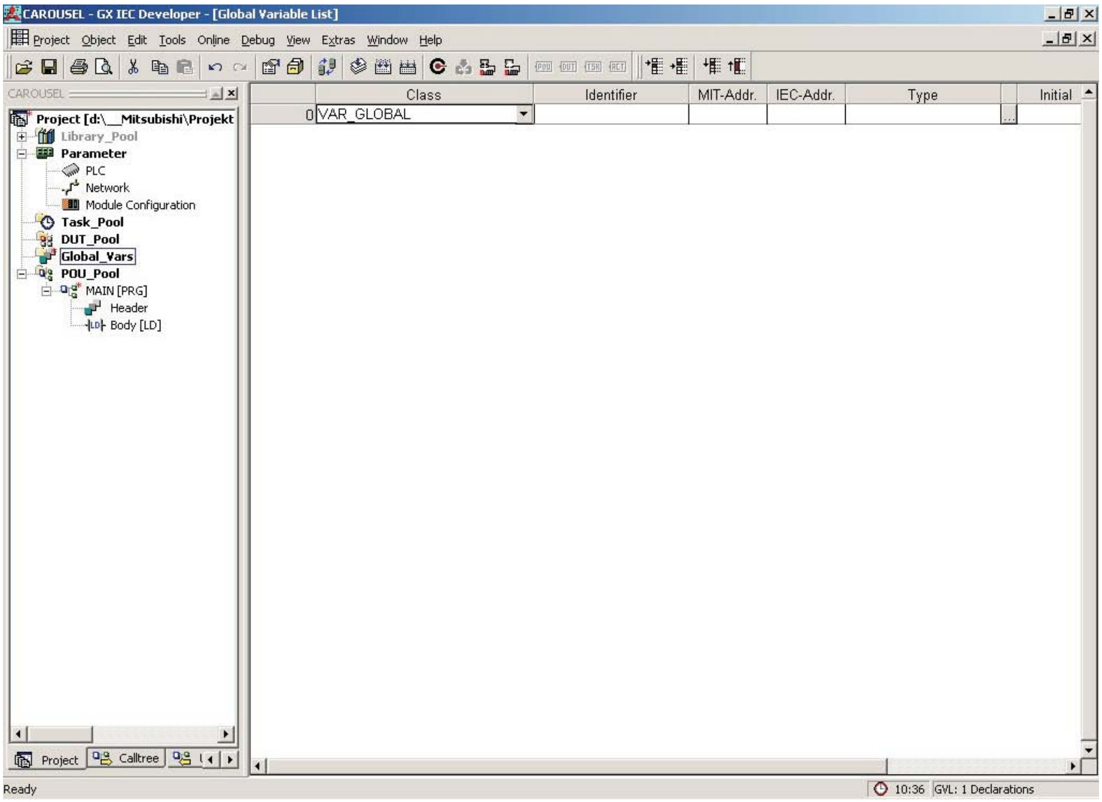
③ Double click on ***MAIN*** program icon or click the symbol on the POU Pool in order to expand the directory branch and display the Header and Body entries:



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.



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

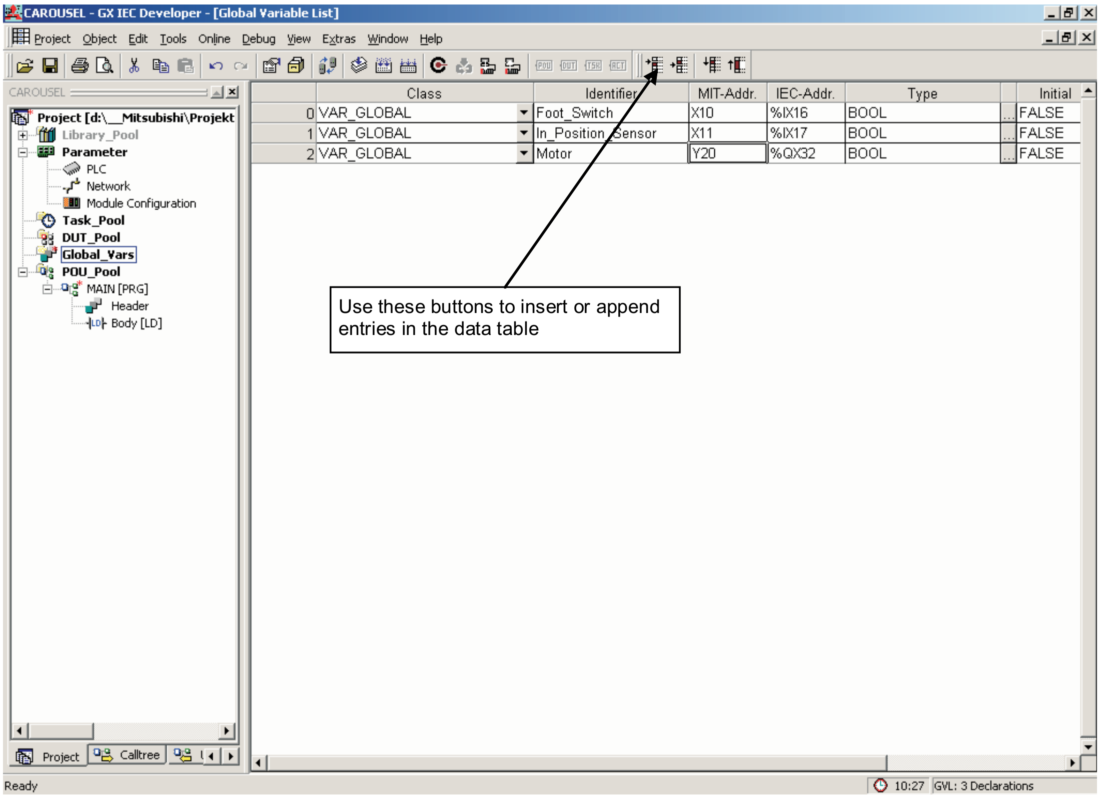
  The initial values are set automatically by the system and cannot be changed by the user.

- 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.

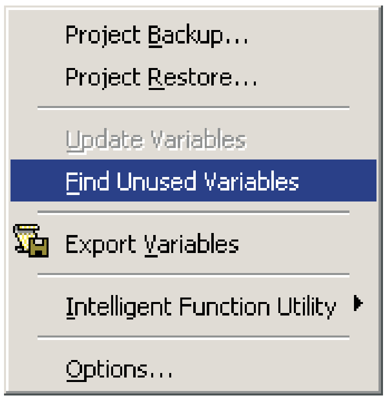
Fill out the table as shown in the following illustration. The variable “Type Selection” is automatically recognised and placed by GX IEC Developer upon entry of the ‘Address’ but can be input manually or modified by clicking on the type select arrow in the ***Type*** field area. When the Mitsubishi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



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

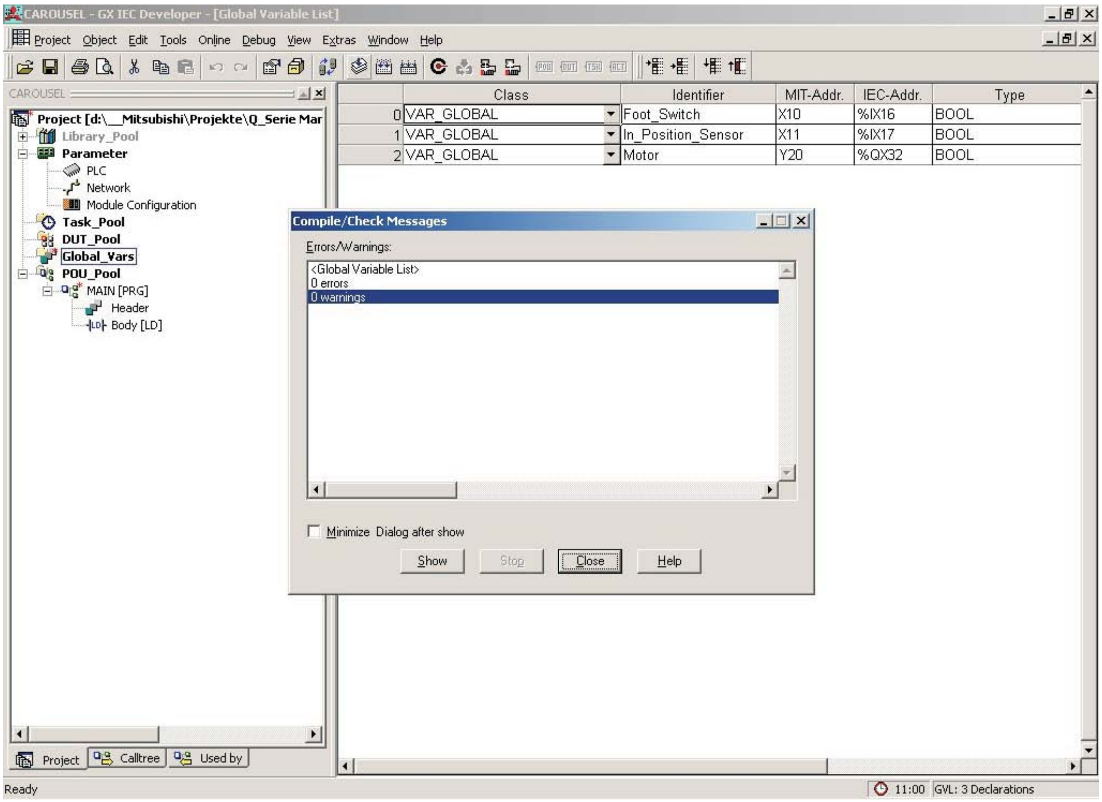
Finding unused variables can only be performed if the project has been built and was not changed since then. Otherwise a warning message will be displayed.

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.

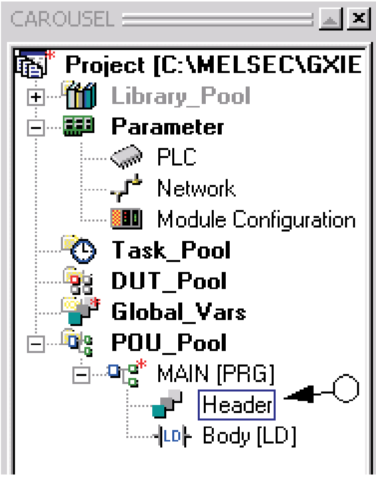
For all FX2N, FX3U, Q & AnA(S) type CPU’s or better, IEC Type REAL (Floating Point) values are fully supported.

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

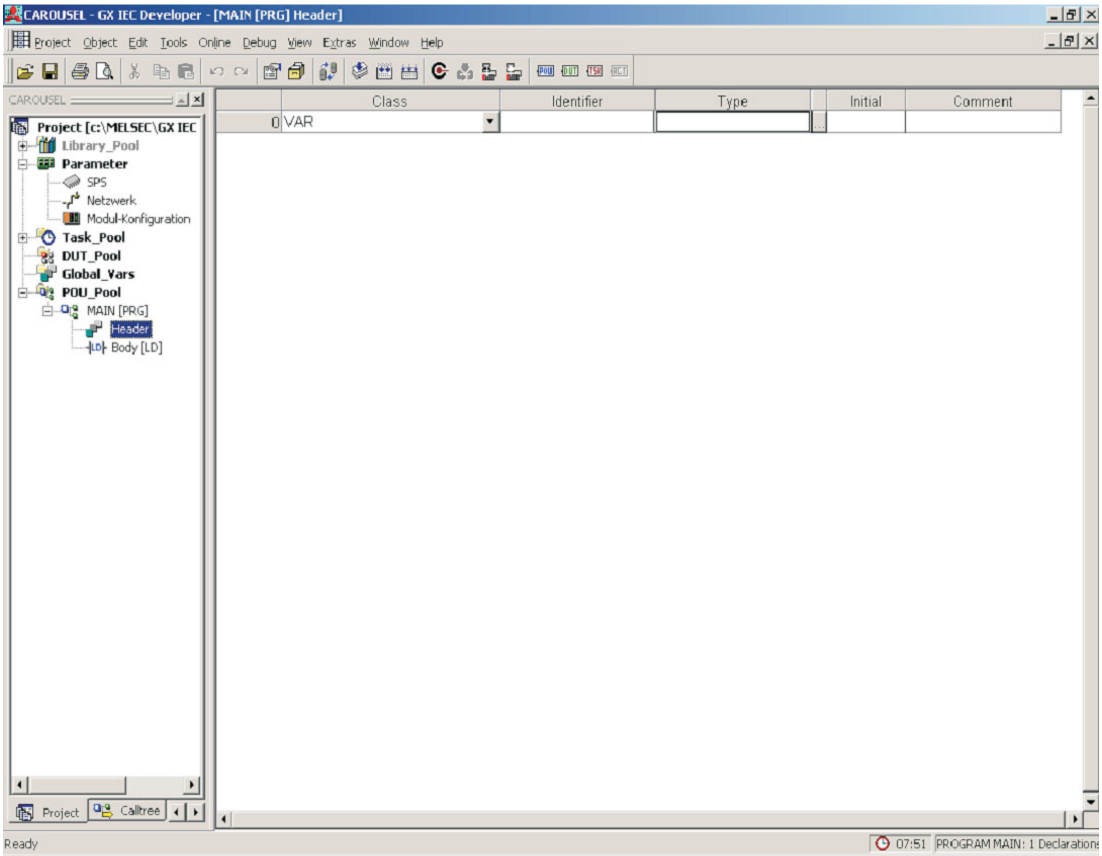


Opening the POU Header

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



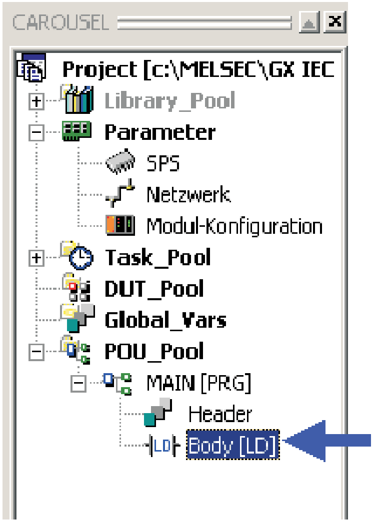
The following screen will be displayed:



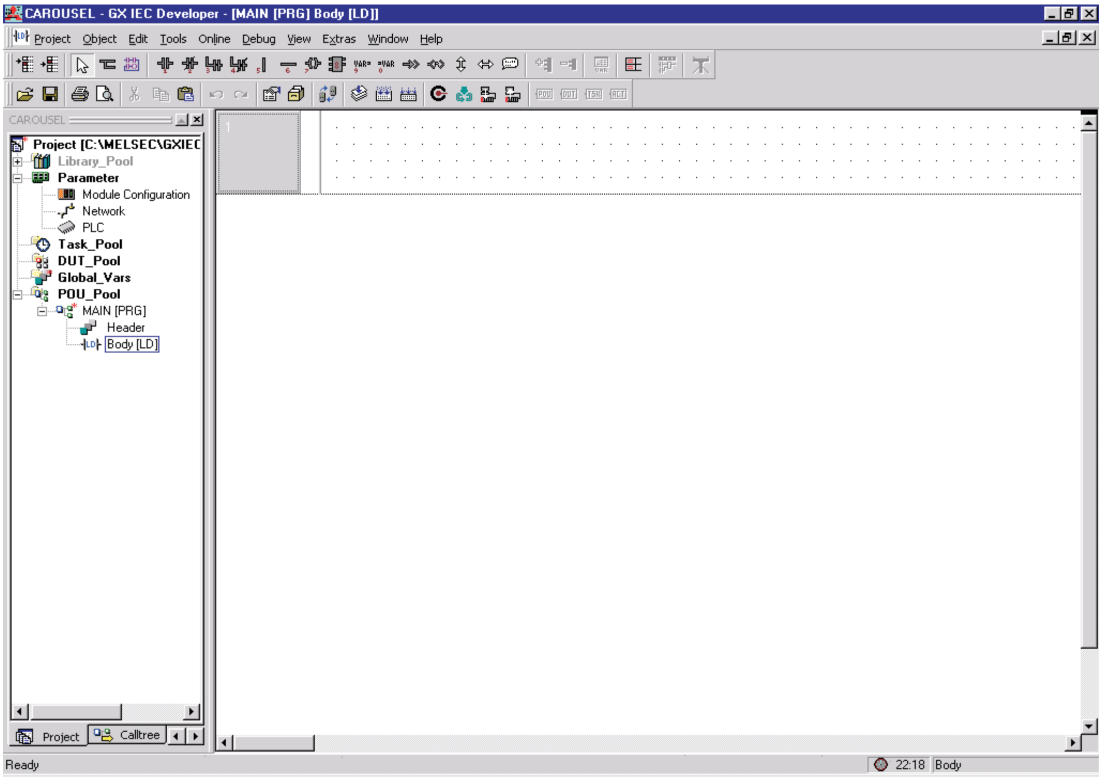
Close this POU Header display.

4.2.5 Programming the POU Body

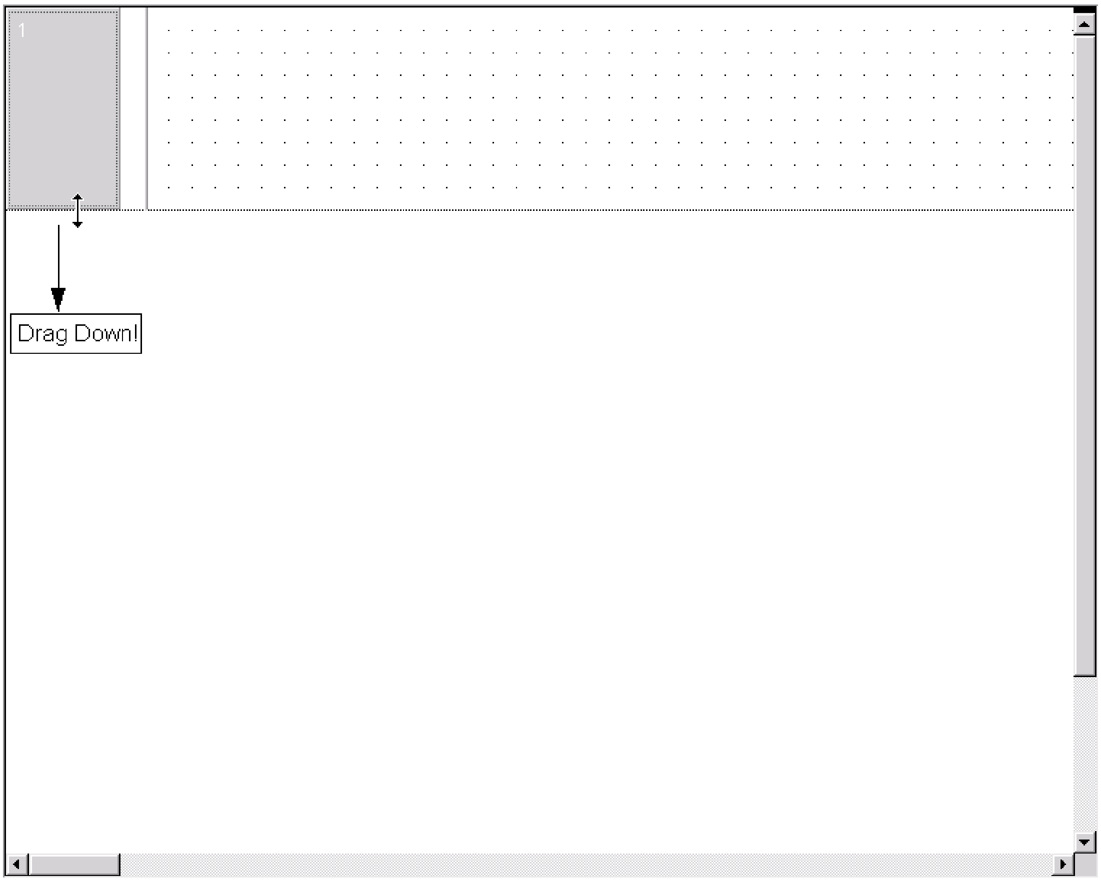
① To open the Ladder diagram editor, double click on the Body selection under the POU pool in the project navigation window:



The following window is displayed:

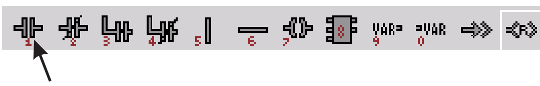


② With the pointer over the window boundary, click and drag downwards to increase the vertical size of the network:

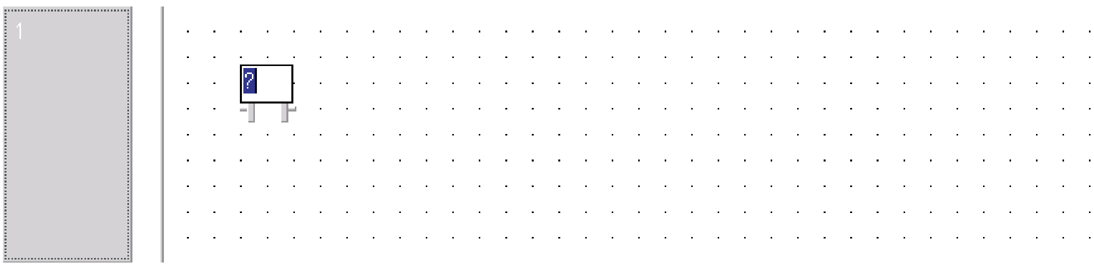


Using the Toolbar Ladder Symbol Selection

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

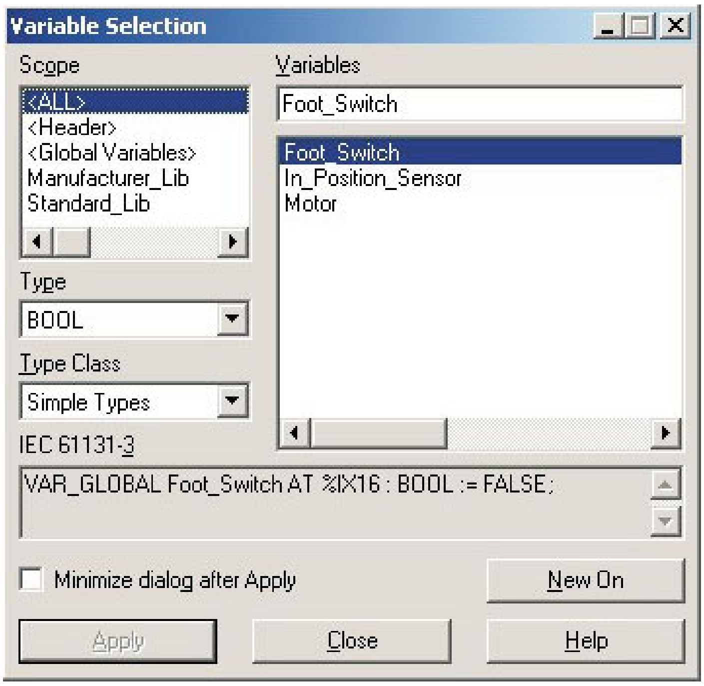


④ Move the mouse pointer over the work area and click to fix the drop position on the window:



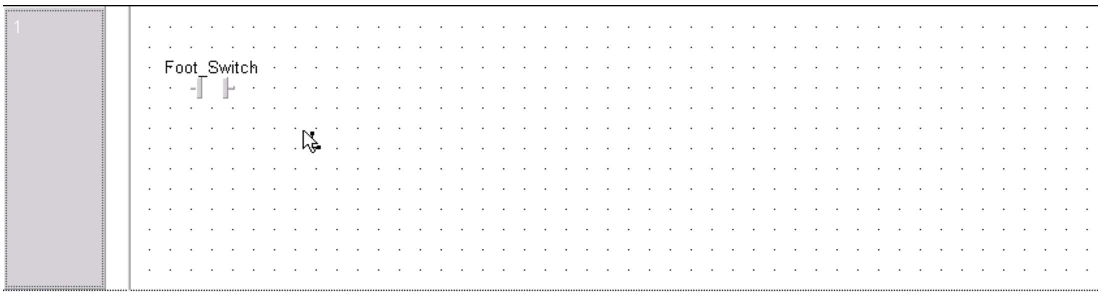
Selecting variables from the POU Header

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:



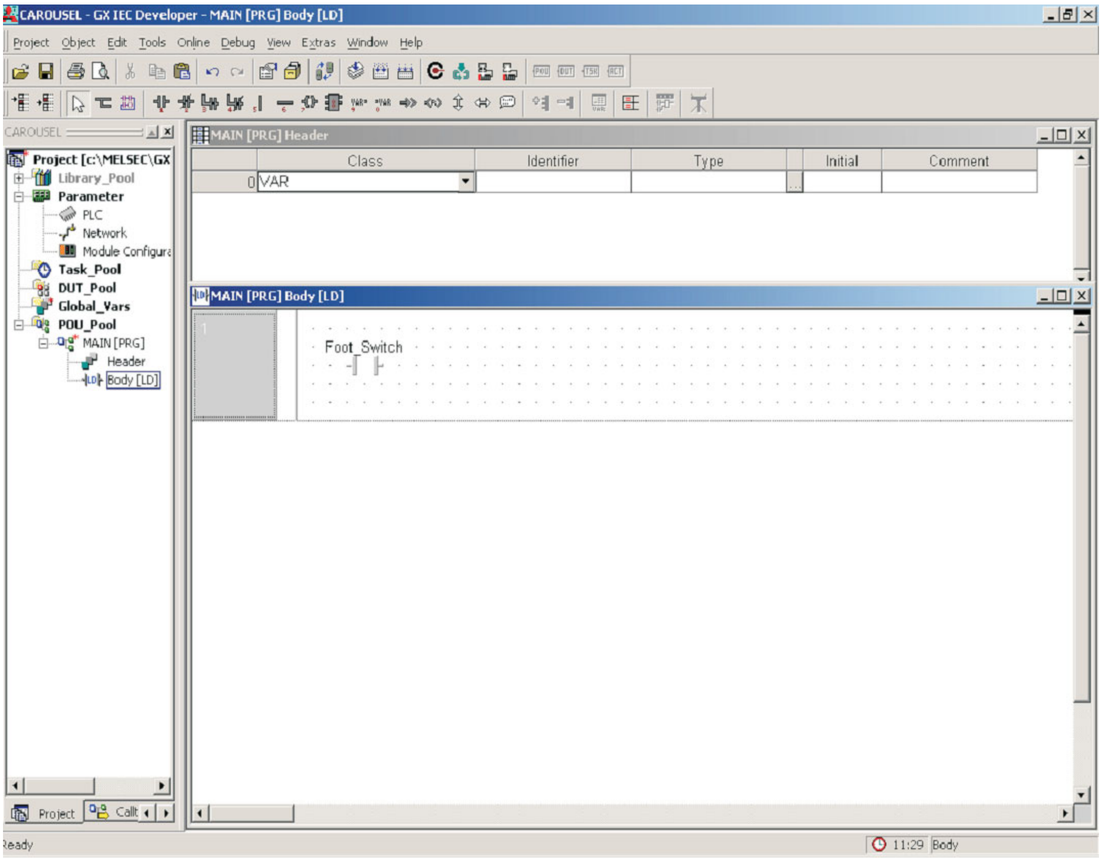
Note that the current ‘Header’ should be selected under the ***Scope*** dialogue area.

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



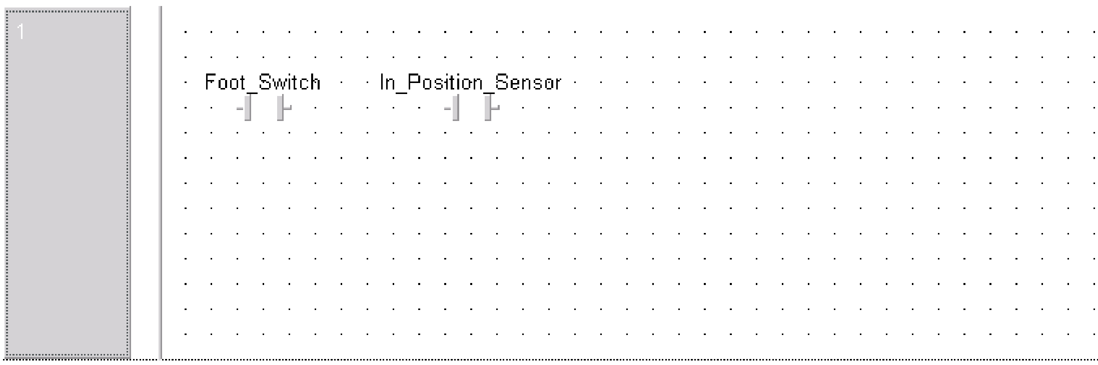
Alternative Variable Specification Method: Editing in Split Screen

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



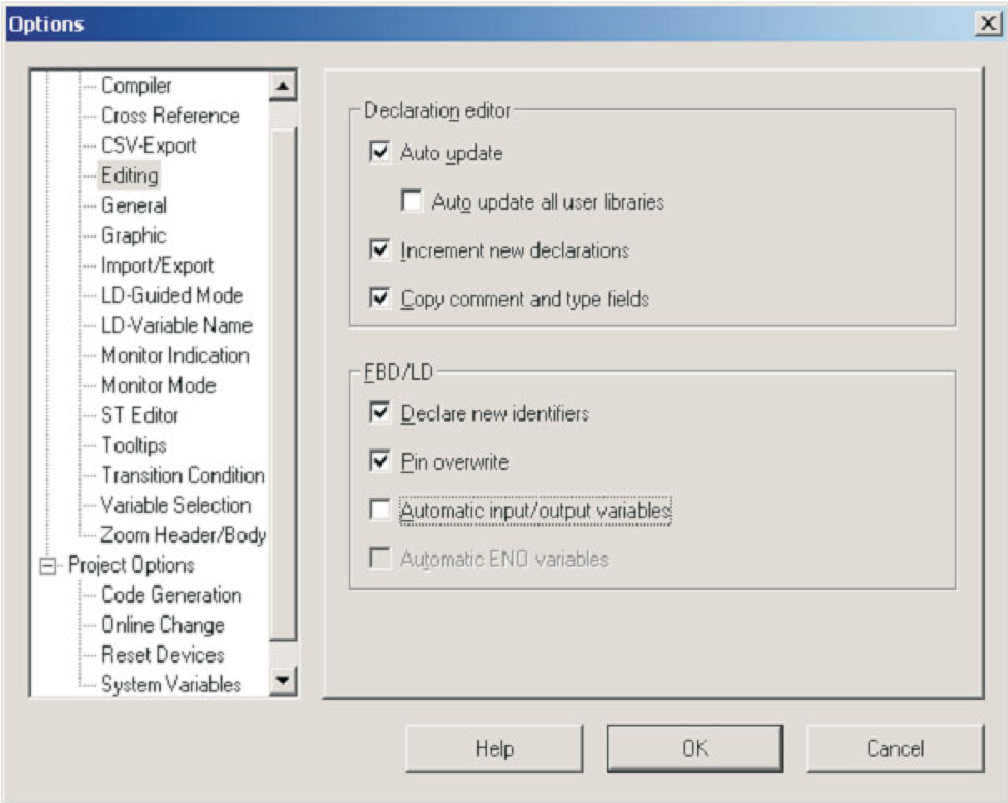
Continue editing Project ‘Carousel’

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



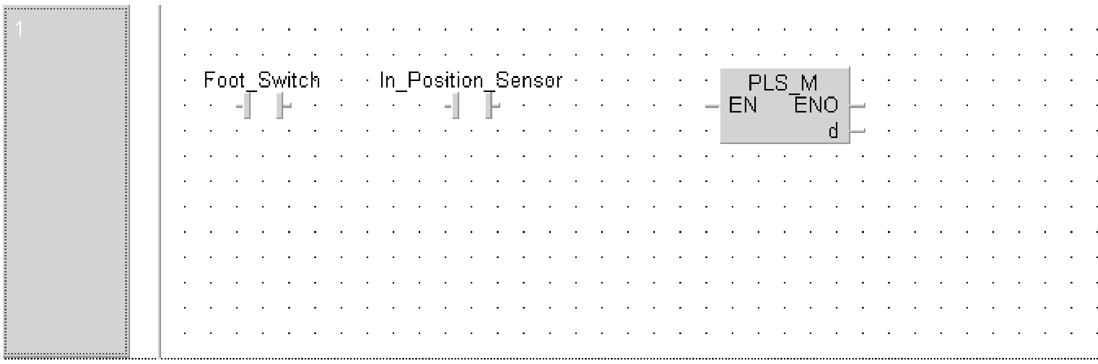
Entering a Function Block command into the Ladder program

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:



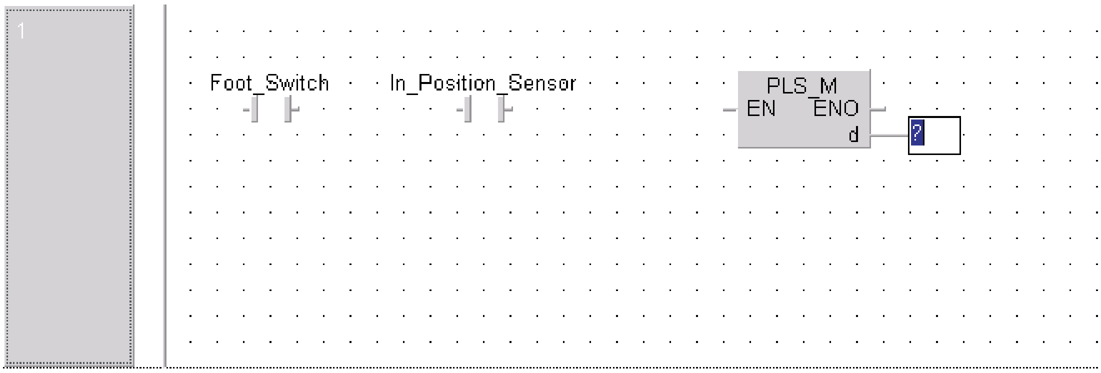
The MELSEC Function Block command, 'PLS_M' will be added to the program as the output function.

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:



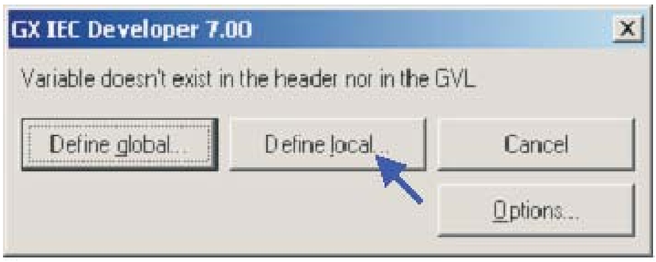
Assigning a Variable to an Instruction

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.

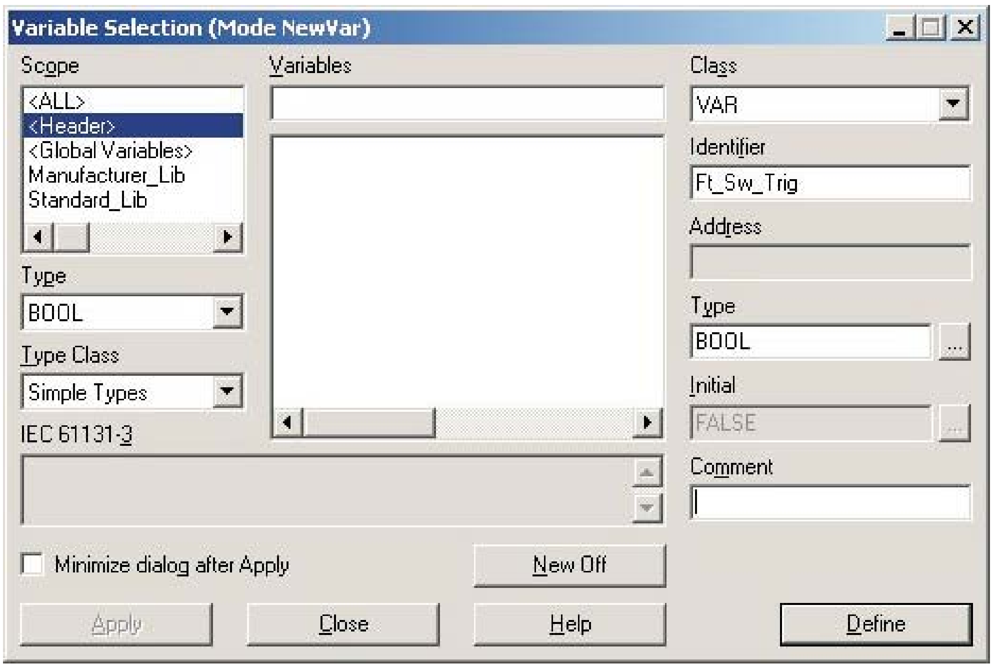


③ Enter the variable name Ft_Sw_Trig into the empty ‘?’ box.

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’:



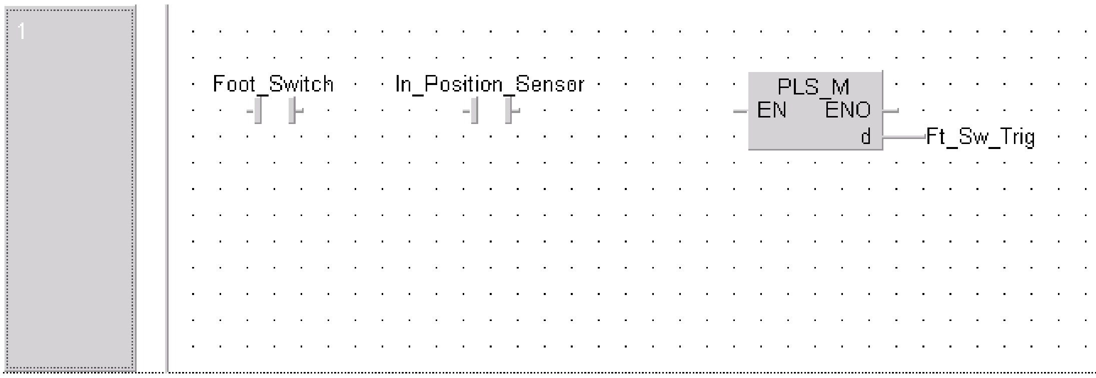
④ Click on ***Define Local*** to define a new Local Variable ‘LVL’. The ***Variable Selection*** window is displayed, prompting a new variable to be defined:



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

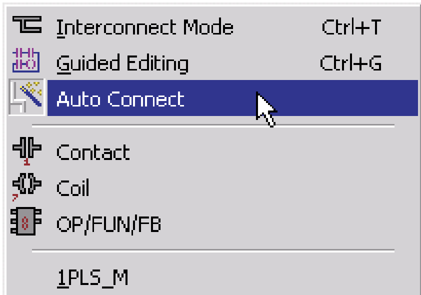
NOTE | To confirm the above operation, check the local header!!

The display should be as follows:

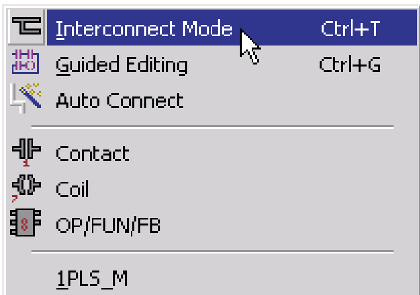


Finally, the ladder network must be finalised by connecting up the elements as follows.

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

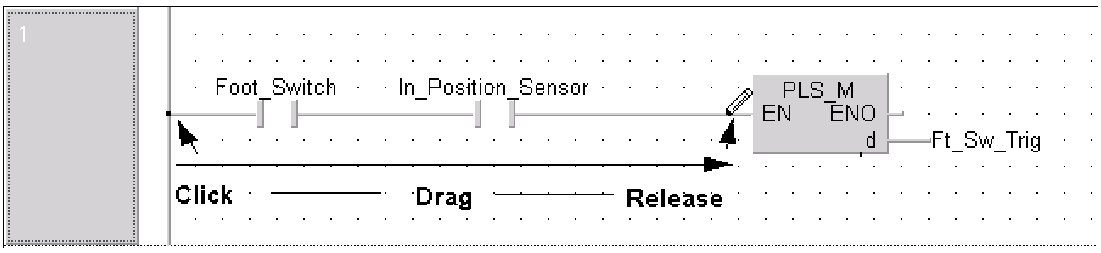


⑦ In the same manner, click to select ***Interconnect Mode***.



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:



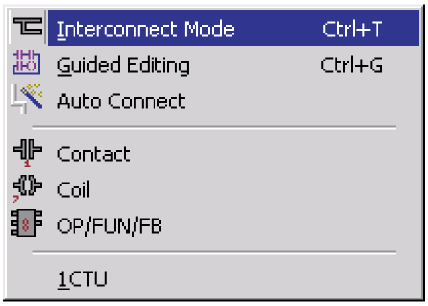
The circuit is now complete.

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.

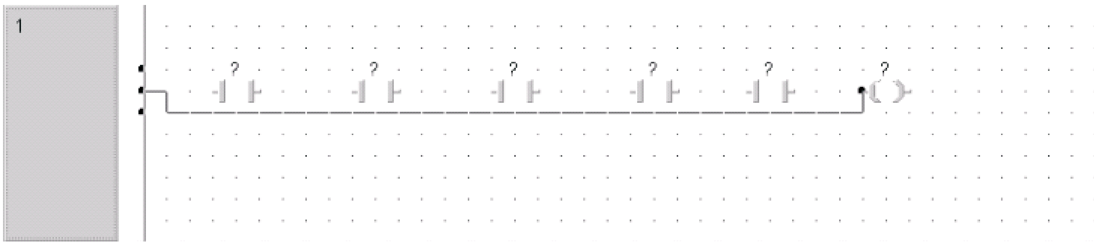
The following text is for illustration purposes only:

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.

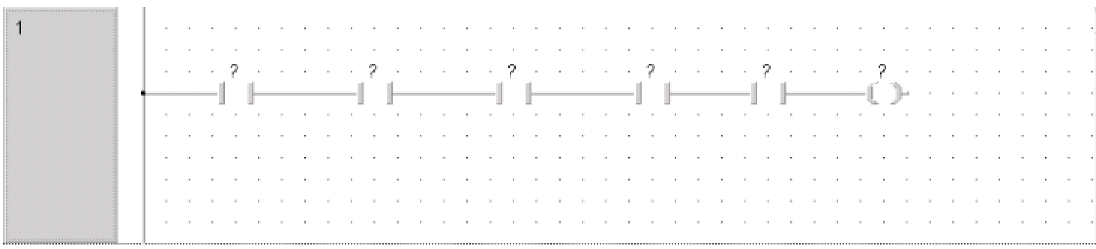


Precautions when using the Ladder Editor

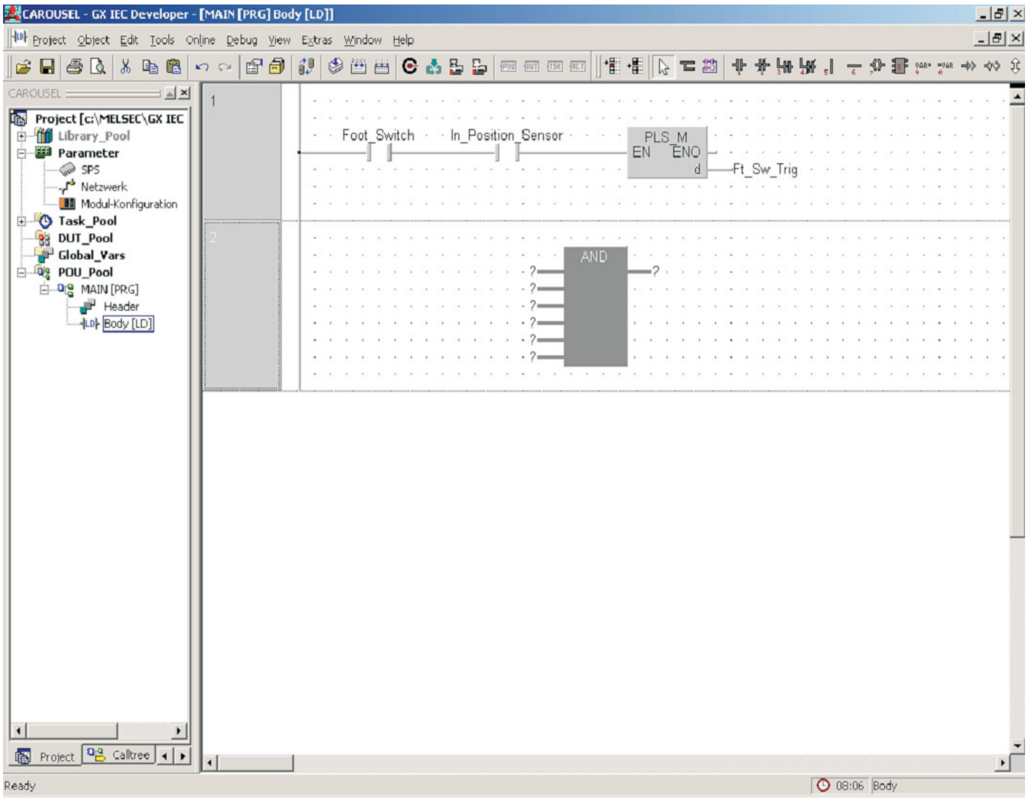
As can be seen from the screen below, because ***Auto Connect*** connects between two points, for a row of contacts the line tries to connect as shown. With ***Auto Connect*** on, the only way to connect these contacts is to connect between each individual pair:



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.

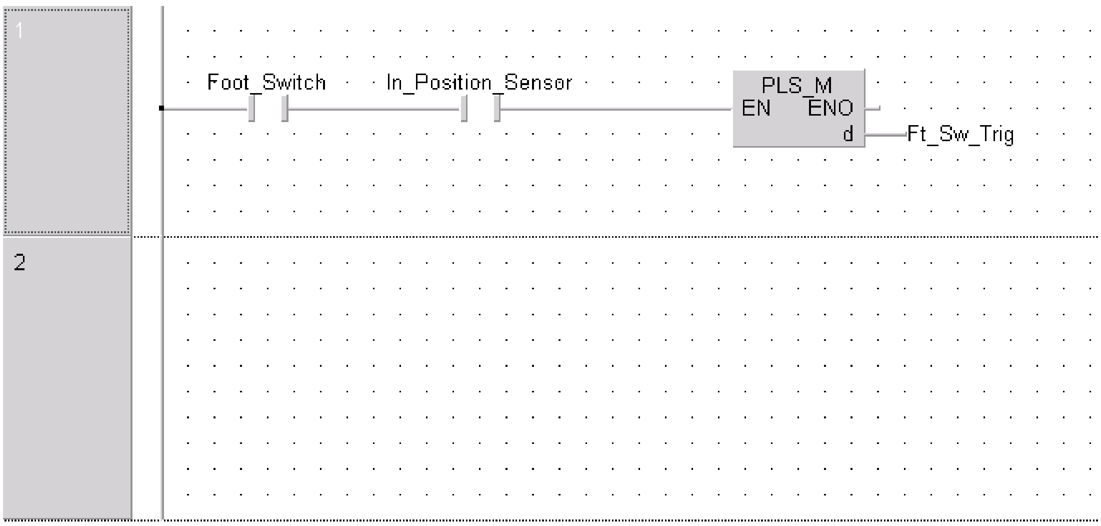


When using multi-legged or ‘pinned’ functions such as MUL, the number of input parameter legs, can be incremented/decremented by using the special toolbar, icons shown. This can also be achieved by placing the cursor at the bottom edge of the function, holding down the left hand mouse button and then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

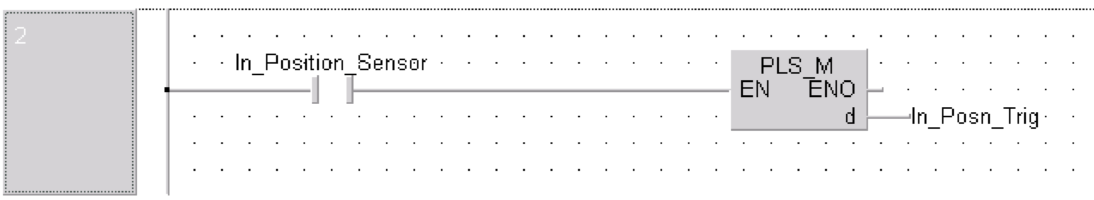


Creating a new Program Network

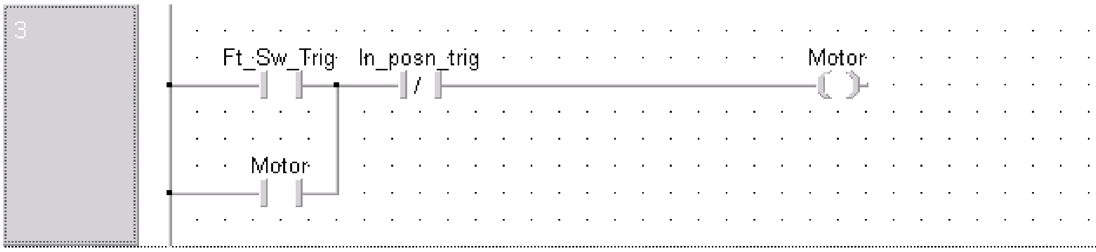
① To create a network below the current one, click the 'insert after'  button. A blank network space will appear:



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

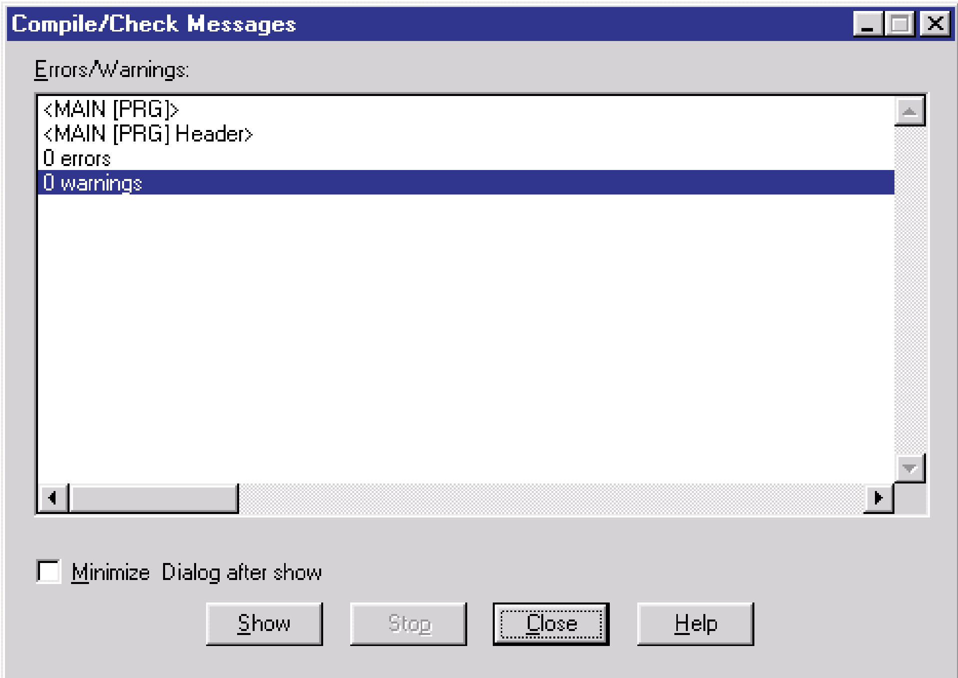


③ Finally, enter the following network as shown:



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:



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.

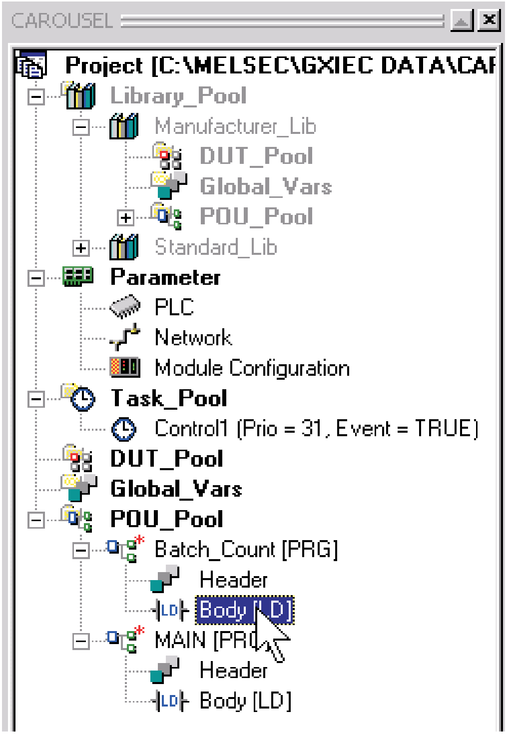
Task:

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.

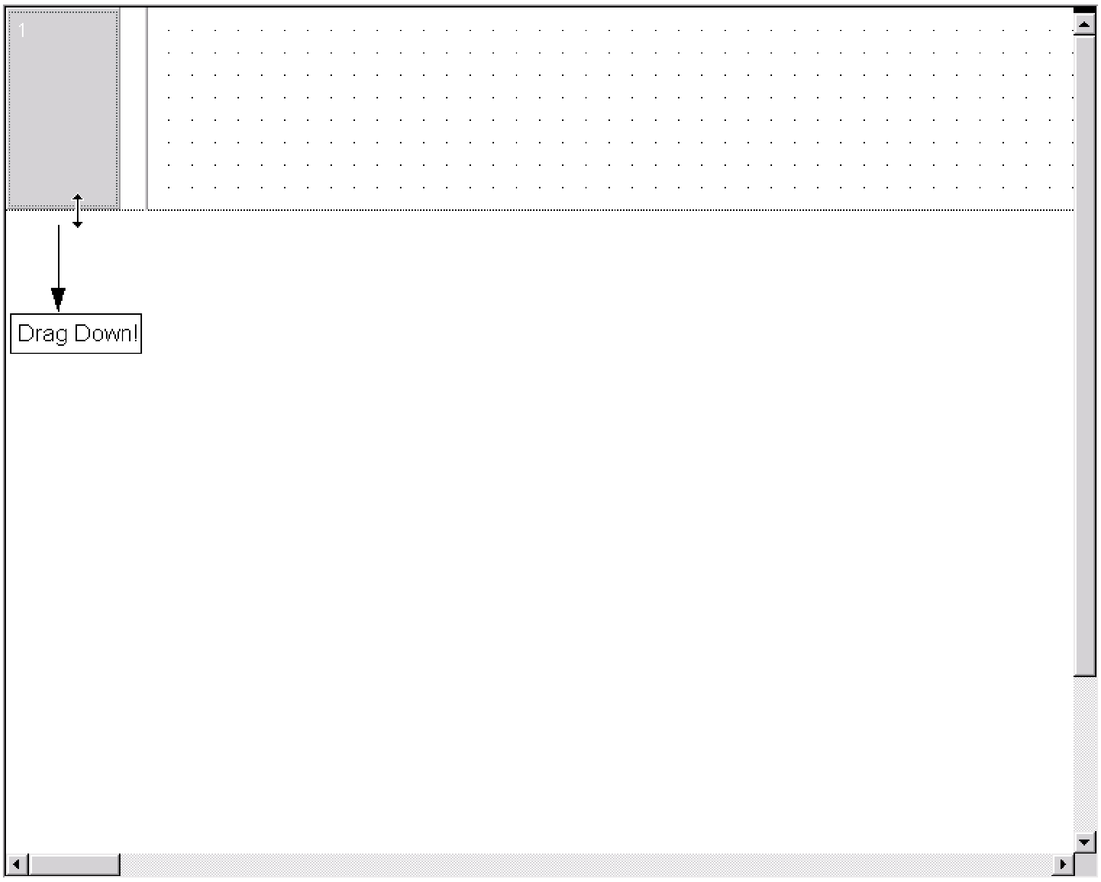
Enter the following POU ladder routine, using the 'free-form' editors as shown:

① Create a new POU by clicking on the button.



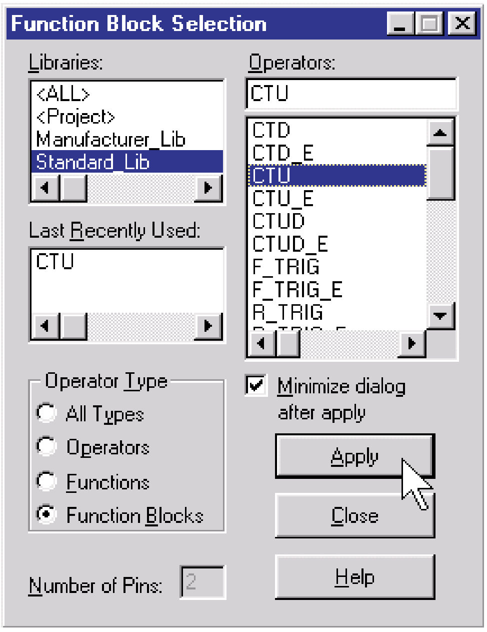
② Select the Body of the new POU by opening the newly created entry in the Project Navigation Window.

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:

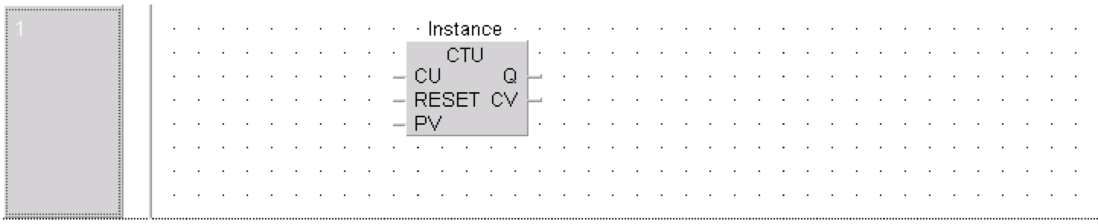


Counting function

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



Drop the IEC Function Block onto the empty Ladder network:

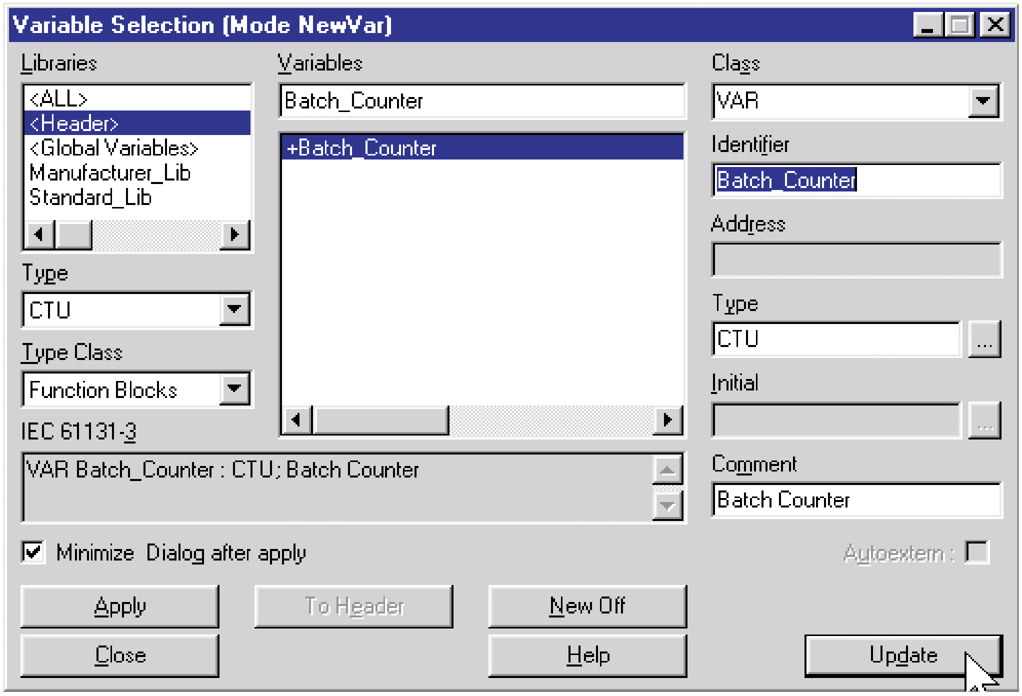


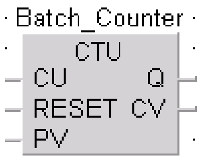
Instances of Function Blocks

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

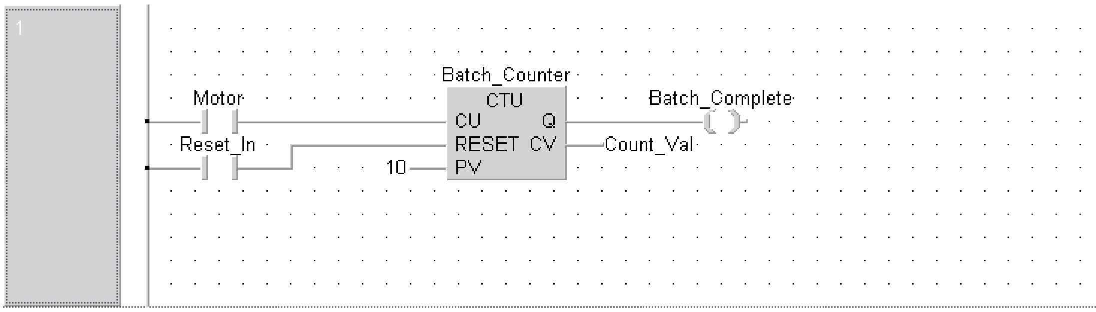
① To create a new name for this instance of the CTU Function Block in this POU, click on the variable name ***Instance*** above the CTU function block. And press F2 to bring up the ***Variable selection*** dialogue. Fill in the resulting window as shown below:





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

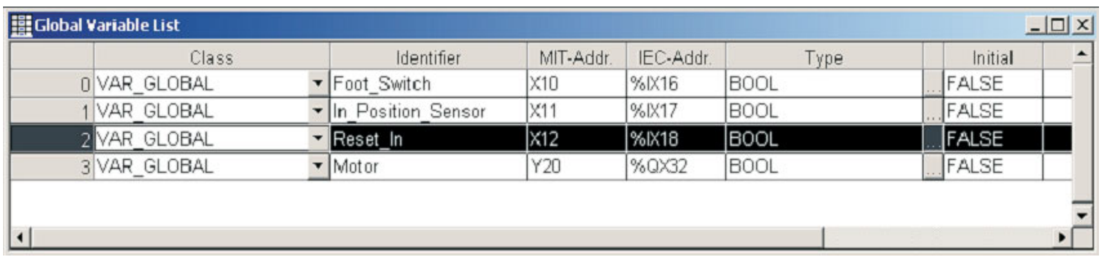
③ Continue to enter the program as previously described so that the following display is achieved:



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

Adding entries to the GVL

Note, in particular: “Reset_In” (Global) - is a new Input mapped from the MELSEC Boolean address X12 or IEC %IX18. This requires a new entry into the GVL as follows:


Global Variable List

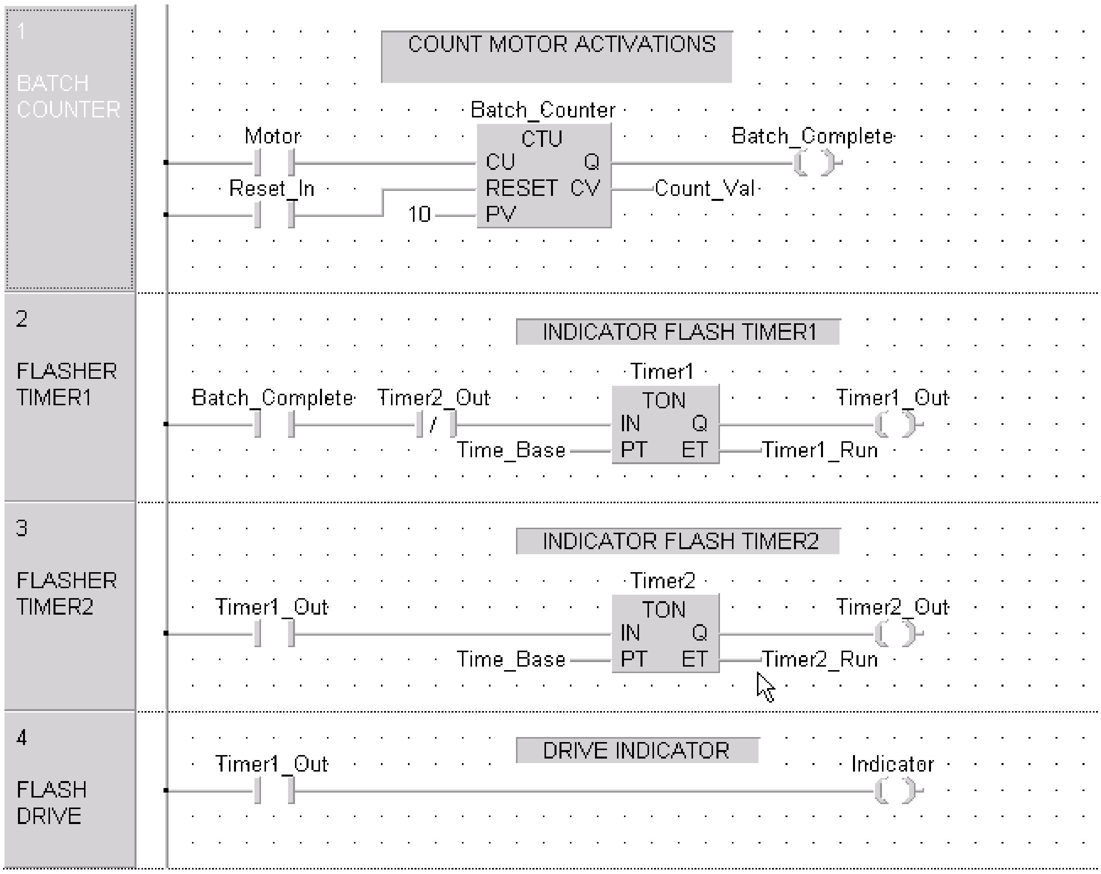
	Class	Identifier	MIT-Addr.	IEC-Addr.	Type	Initial
0	VAR_GLOBAL	Foot_Switch	X10	%IX16	BOOL	FALSE
1	VAR_GLOBAL	In_Position_Sensor	X11	%IX17	BOOL	FALSE
2	VAR_GLOBAL	Reset_In	X12	%IX18	BOOL	FALSE
3	VAR_GLOBAL	Motor	Y20	%QX32	BOOL	FALSE

	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	X10	%IX16	BOOL	...	FALSE
1	VAR_GLOBAL	In_Position_Sensor	X11	%IX17	BOOL	...	FALSE
2	VAR_GLOBAL	Reset_In	X12	%IX18	BOOL	...	FALSE
3	VAR_GLOBAL	Motor	Y20	%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:

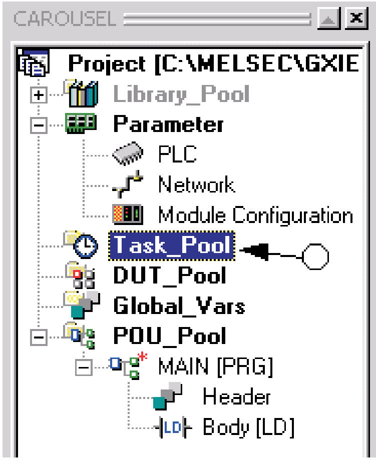
MAIN [PRG] Header

	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

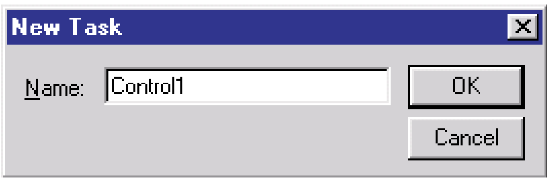
In order for the POUs “MAIN” and "Batch_Count" to be assembled and executed in the PLC, they must be specified as valid tasks in the ***Task Pool***.

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

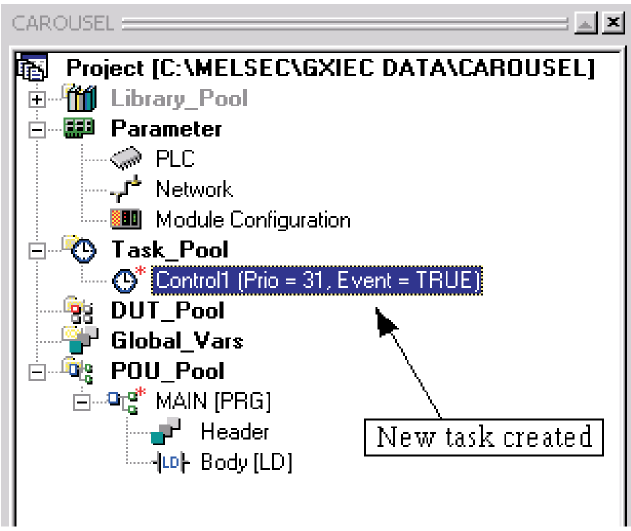


② Then click on the Task button on the Toolbar. Alternatively, ‘Right Click’ the task pool icon in the Project navigation window and select the ***New Task*** option from the menu.

③ Enter the name of the New Task ("Control1") in the prompt window.



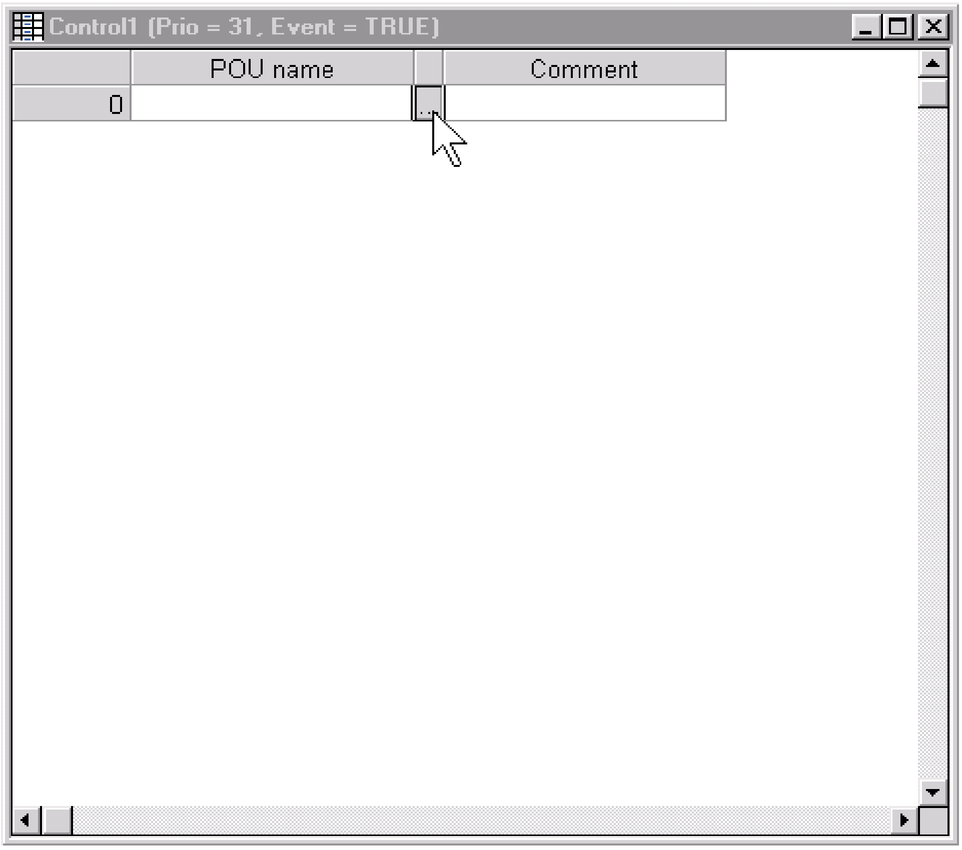
④ Click ***OK*** and the Project Navigation window now shows the newly created task called “Control1”:



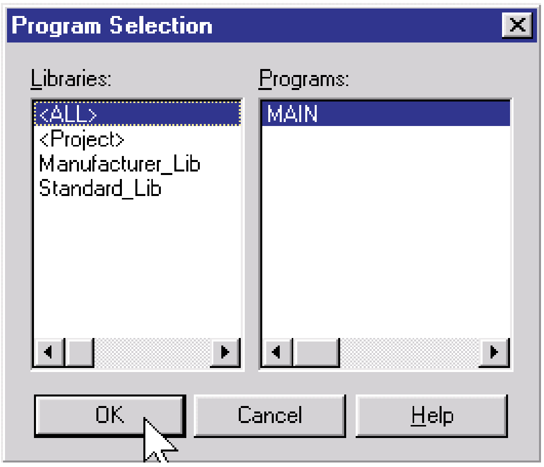
Assigning the POU to Task

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:



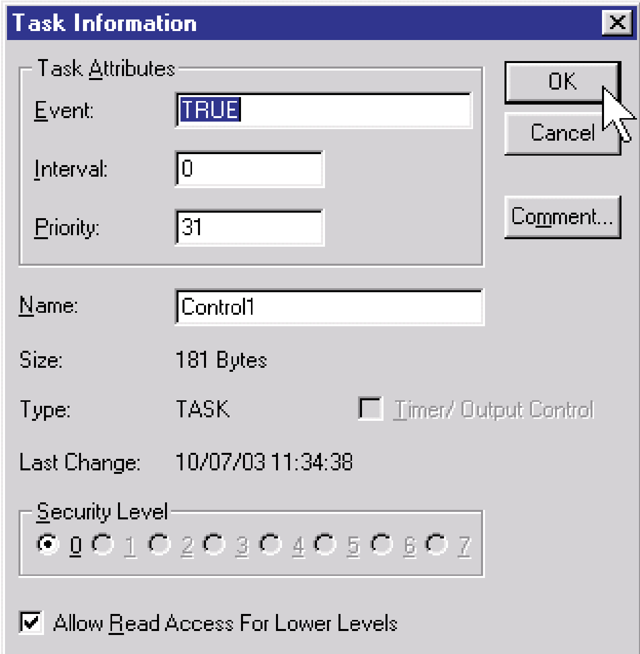
② Click on the centre ‘choice browse’ Ellipsis as shown above. The following prompt dialogue is displayed:



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 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.

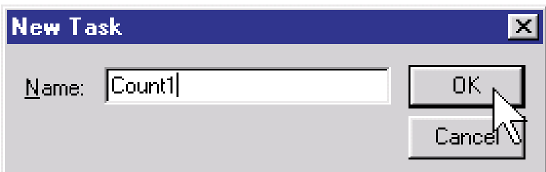
Before continuing, it is a good idea to “SAVE” the project; click on the Save  Button.

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

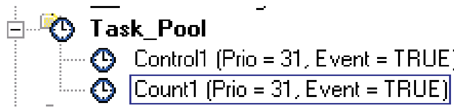
The POU "Batch-Count" needs also to be referenced (called) by a task in the ‘Task Pool’.

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’  icon on the toolbar.

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



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



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.

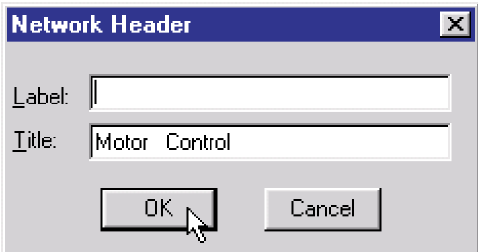
Save the project using the save  button. The project is now complete and must therefore be transferred to the PLC.

4.2.7 Program Documentation

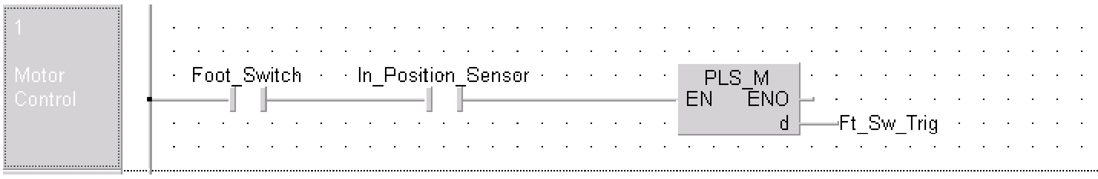
Network Header

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:



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

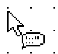


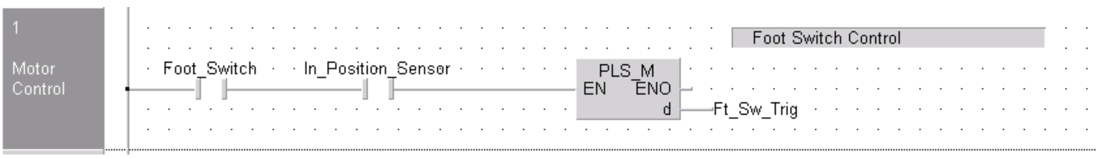
Note that the title may require pre-formatting (Padding with spaces), depending on the screen resolution set, to read correctly as the text auto wraps to fit into the horizontal space available (22 characters max).

Network Comments

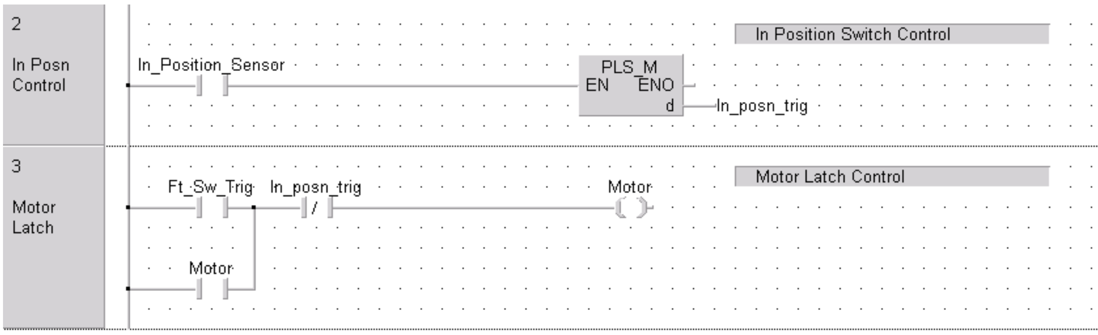
Comments enable virtually freehand text descriptors to be added anywhere inside the ladder network area. This is vital to provide descriptions of the operation of the program.

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

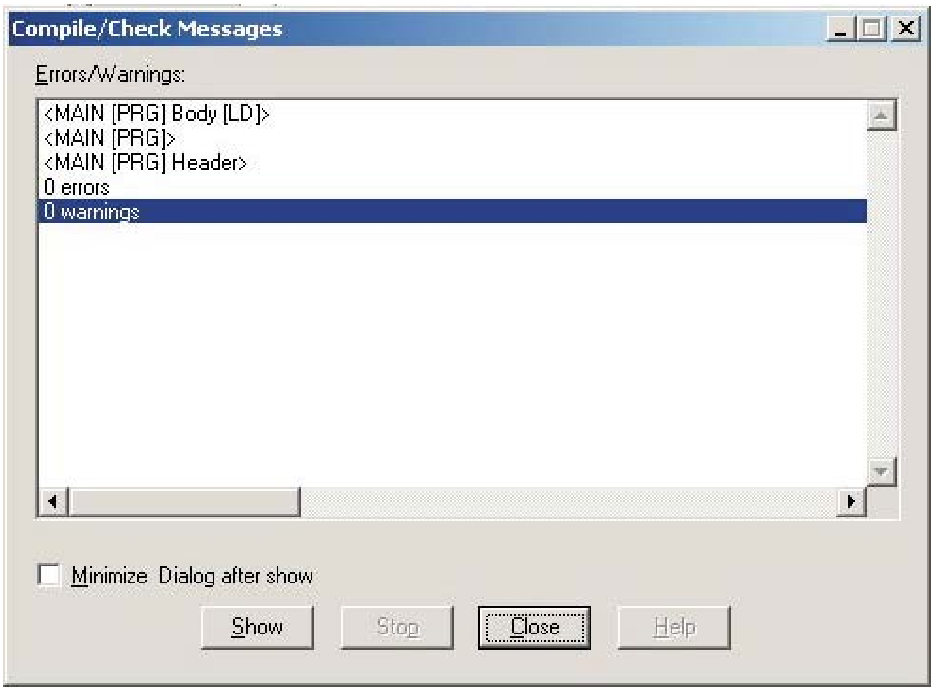
Click once on the comment to highlight and press the <Delete> key on the keyboard.

Cutting / Copying a comment

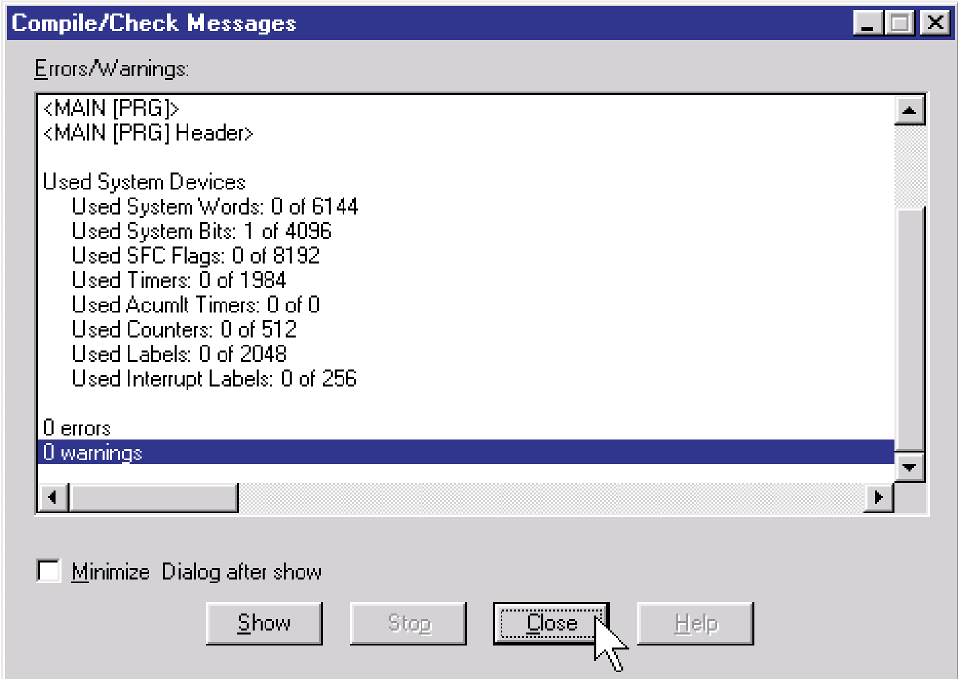
Duplication of comments is achieved by clicking on the left hand end of the source comment to highlight it. Use windows cut/copy – paste procedure and click the mouse once again to set position of destination comment in another network.

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:



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:

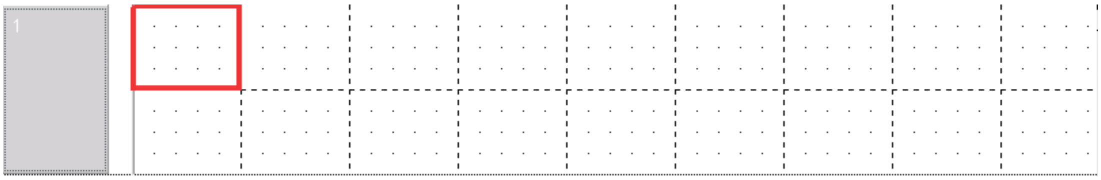


③ Click ***Close*** to exit this display.

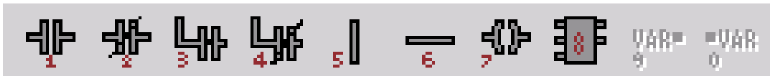
4.2.9 Illustration: Guided Ladder Entry Mode

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.

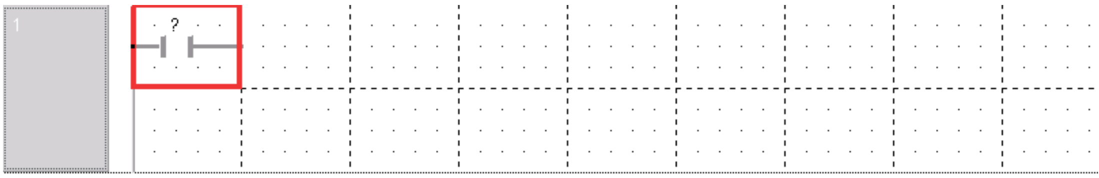
① Enter the ***Guided Entry Monitor*** mode by pressing the  button on the tool bar. The following matrix is placed into the edit area:



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 Q-SERIES PLC.

There are a variety of different methods of connecting GX IEC Developer to a MELSEC PLC:

- FX-Series / A-Series / QnA PLC Programming port

  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.

- Q-Series PLC Programming port

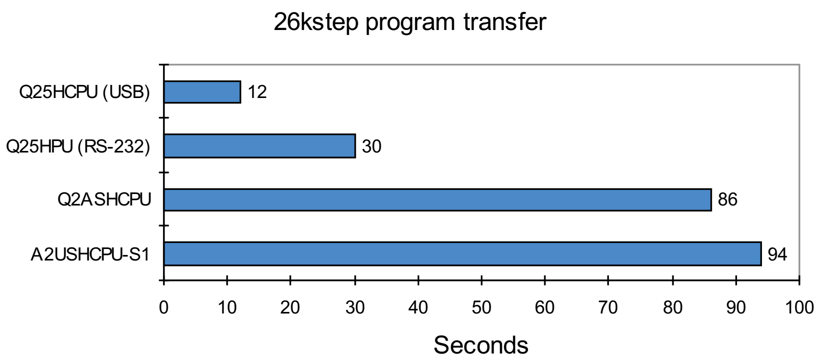
  RS232 using special programming cable.

- Q-Series PLC Programming Interface

  USB - Preferred: Standard USB A-B communications cable.

For the Mitsubishi Training Rigs, connect the computer to the Q PLC as shown in the diagram below:

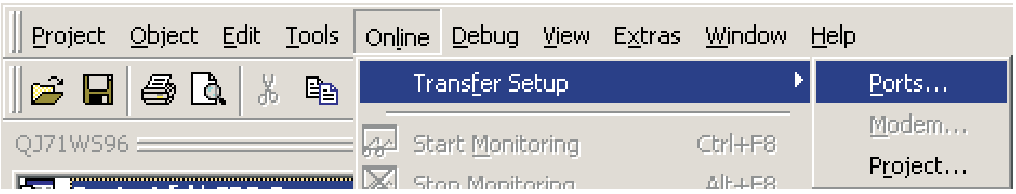
The Table below illustrates the comparison of program transfer times between fastest A-Series CPU with QnA and Q-Series Processors. Note the significant speed of Q- Series increase over A-Series PLC’s:



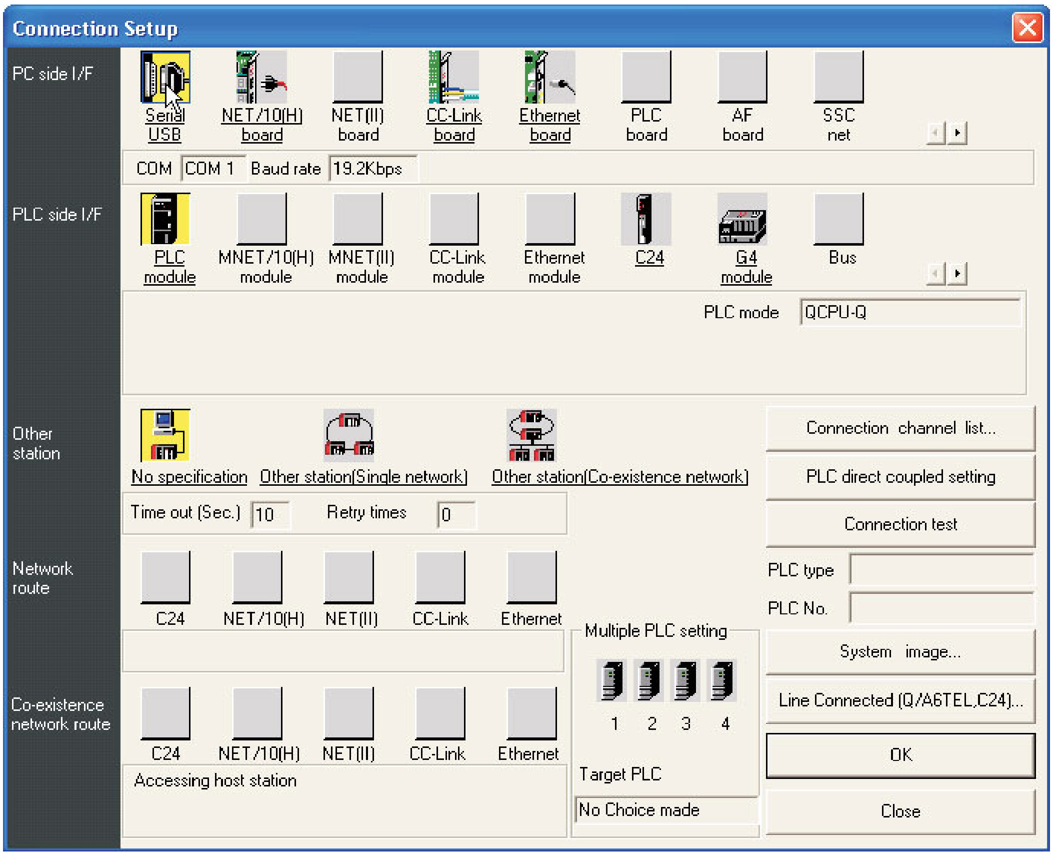
4.3.2 Communications Port Setup

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

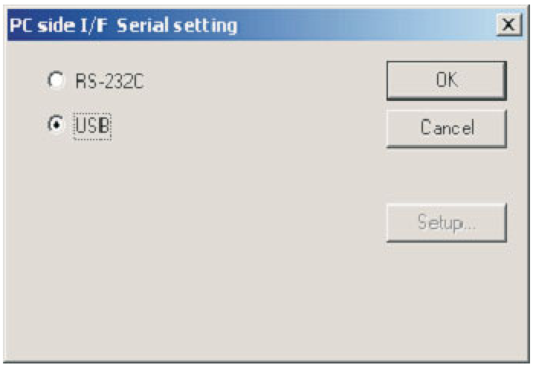
① From ***Online*** Menu, select ***Transfer Setup*** and then ***Ports***:



The following ***Connection Setup*** window will be displayed:

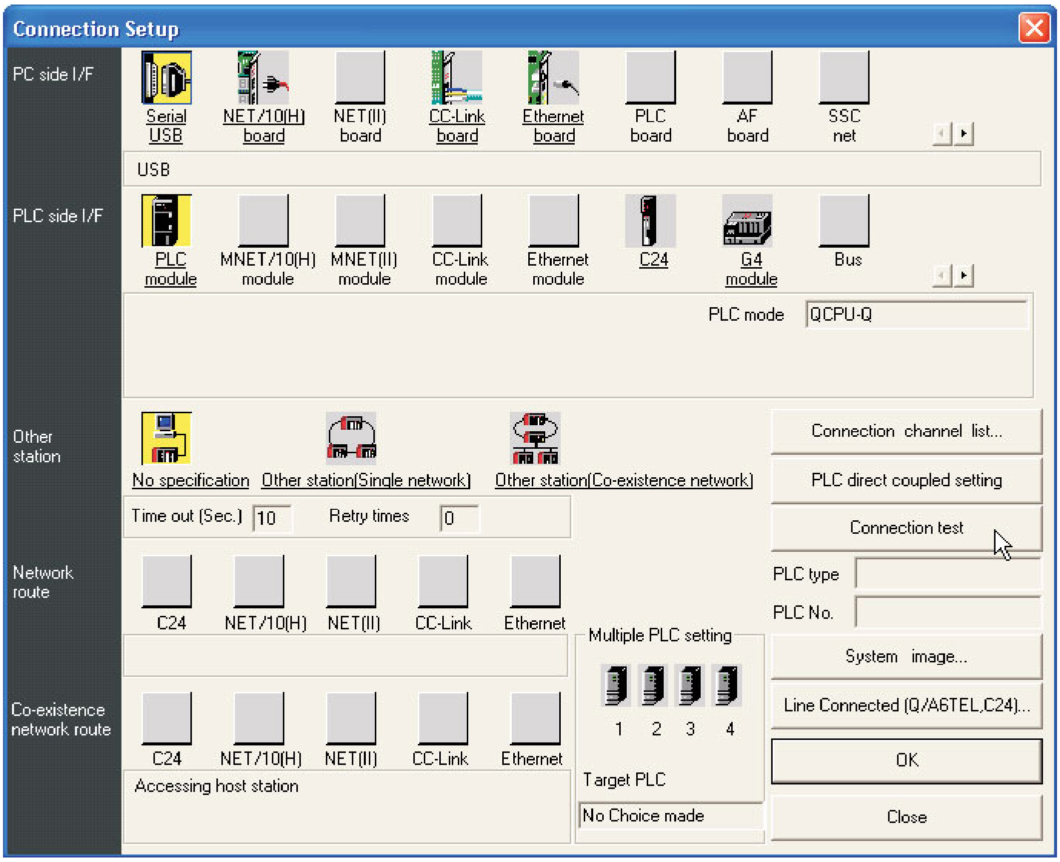


② Double click the mouse on the yellow ***PC side I/F*** – ***Serial*** Button and the following dialogue window is displayed:



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

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



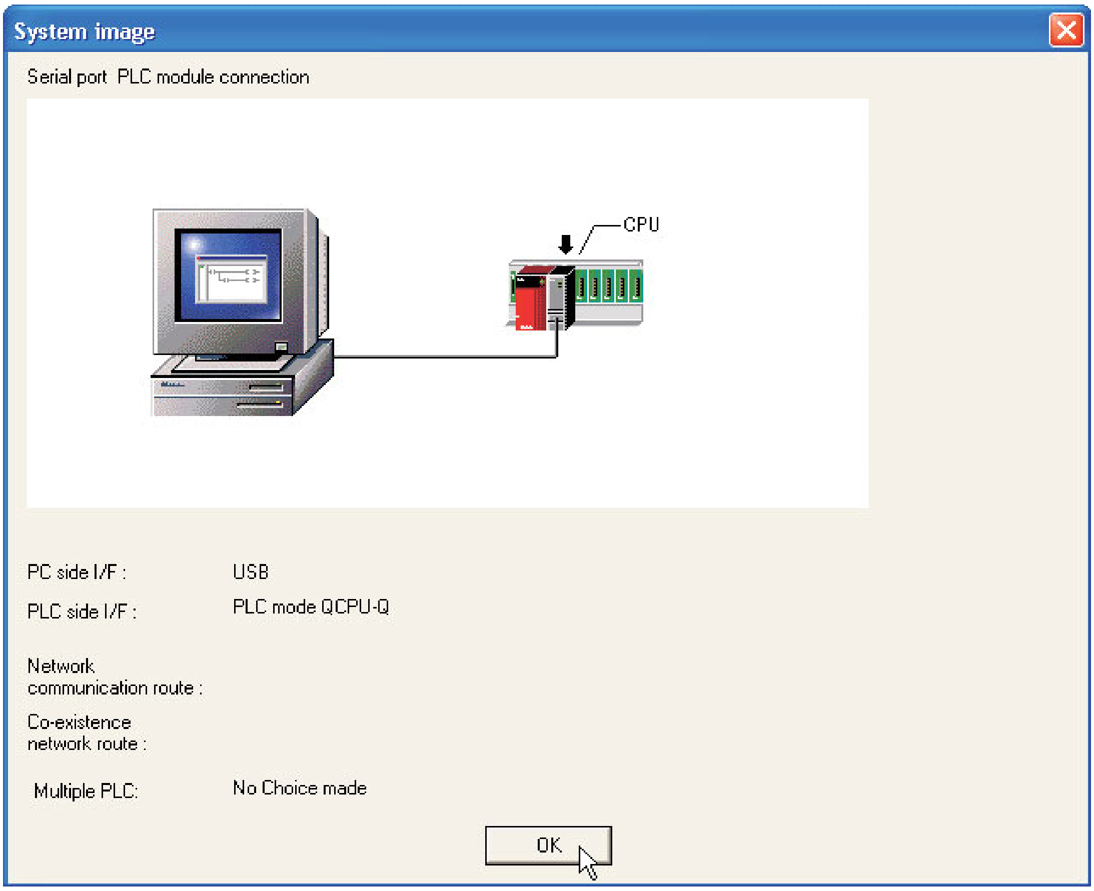
⑤ The following message should be displayed:

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



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

As can be seen from the previous display, these particular Connection Setup parameters utilise the USB Interface.

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 Formatting PLC Memory Q/QnA PLC's

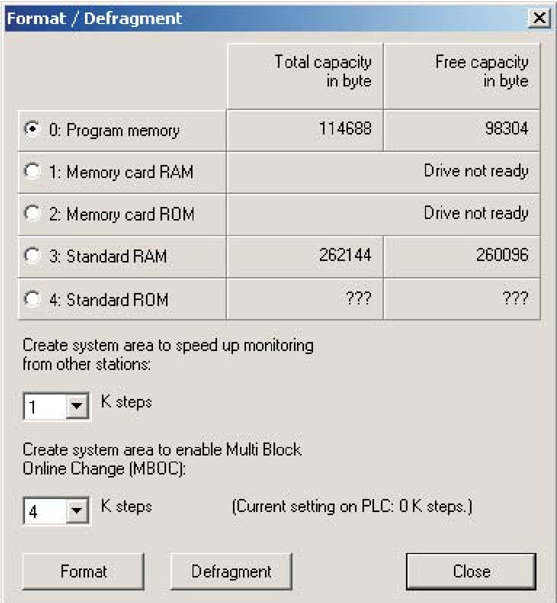
Before proceeding further; when using Q and QnA PLC's, it is highly recommended that the memory be formatted first.

The Q/QnA's File Structure

How to format and defragment the Q/QnA series drive

Before you can use the memory in the MELSEC Q/QnA series CPUs you must first format the corresponding drive. This applies both for the internal RAM and the extrernal memory cards.

Select ***Format Drive*** in the ***Online*** menu. The ***Format / Defragment*** dialog box is displayed.



- 0 ... 4: Drive to be formatted or defragmented.
- Create system area to speed up monitoring from other stations

  If a Q/QnA is connected to GX IEC Developer and a remote Q/QnA is to be monitored, both PLCs (host and remote) have to be formatted with at least 1-K steps system area available in the dialog ***Format / Defragment*** . If one of both PLCs is not formatted with this system area, the remote Q/QnA cannot be monitored.

- Create system area to enable Multi Block Online Change (MBLOC)

  In the CPUs supporting the MBLOC function the number of steps which can be exchanged in the online mode has been expanded to 1024 steps. These 1024 steps do not have to be within one block. It is also possible to make changes in several small blocks. The maximum number of blocks is fixed to 64. However, the number of 1024 steps must not be exceeded (Q series only except Q00(J), Q01CPUs)

- Format

  Selecting this button starts the format procedure

- Defragment

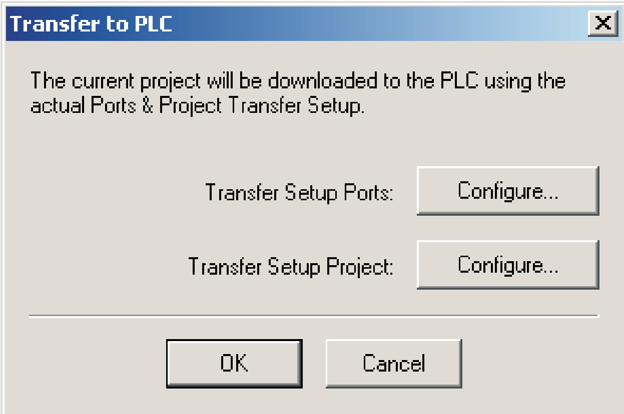
  Because of the file structure used in the Q/QnA series CPUs it is possible for the drives to become fragmented after data have been written to the CPU, just as with a normal hard drive. Selecting the ***Defragment*** button executes a procedure that defragments the contents of the drive for better performance.

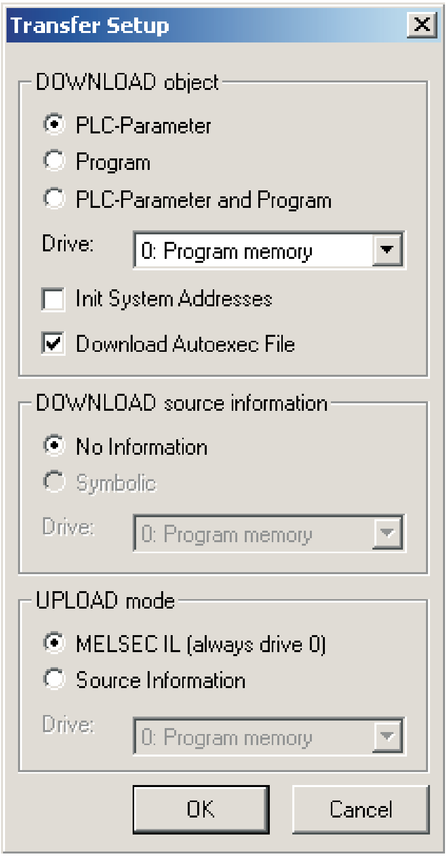
4.3.4 Downloading the project

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.

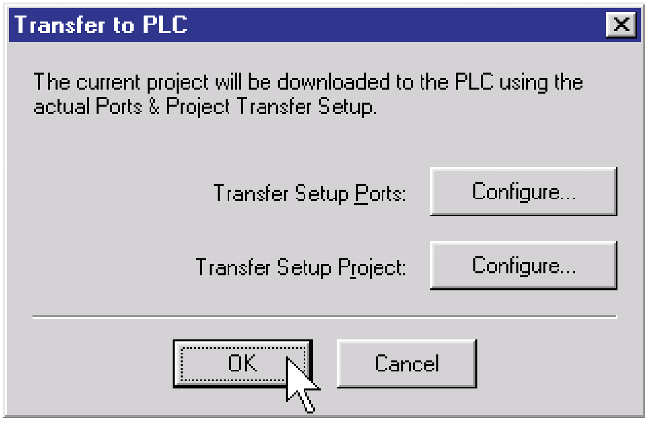




② Click on ***PLC-Parameter and Program***

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

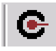
④ To send the project to the PLC, click the ***OK*** button to execute the transfer.

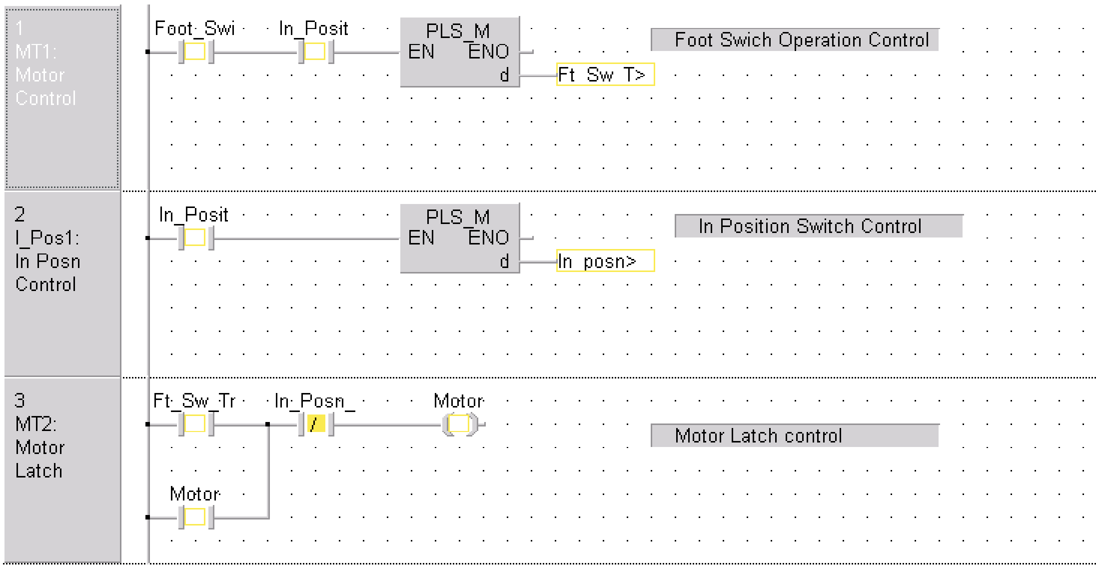


4.4 Monitoring the Project

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.

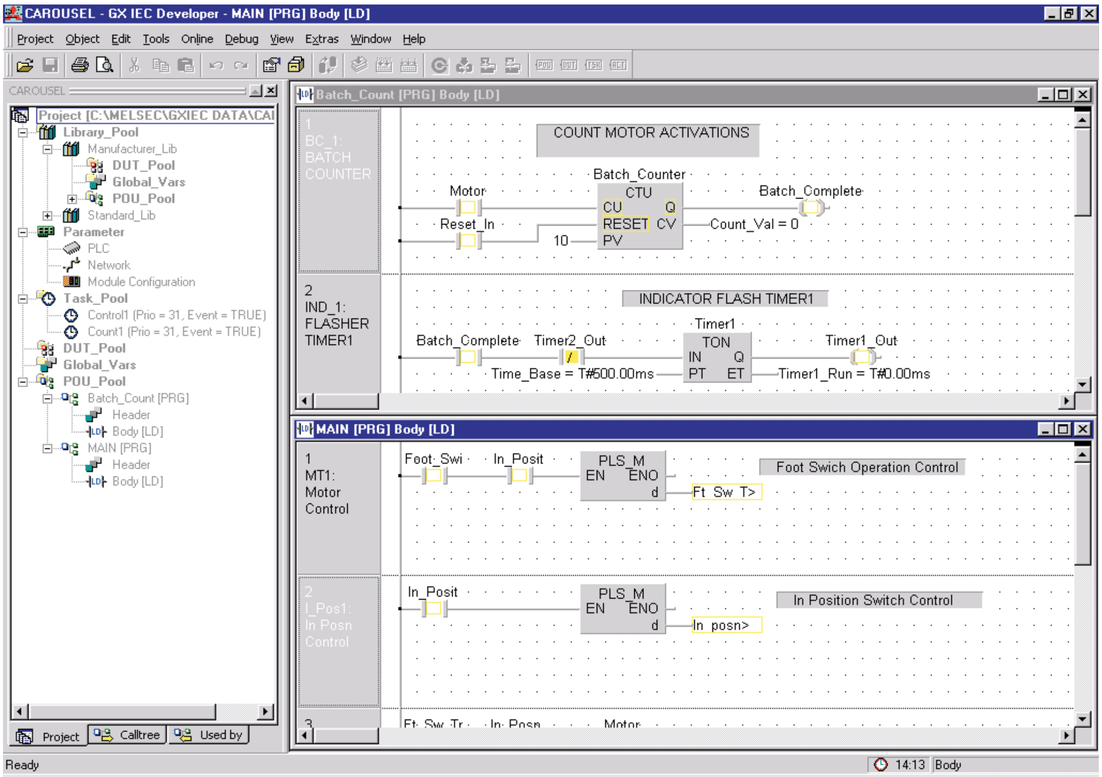
4.4.1 Split / Multi Window Monitoring

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).

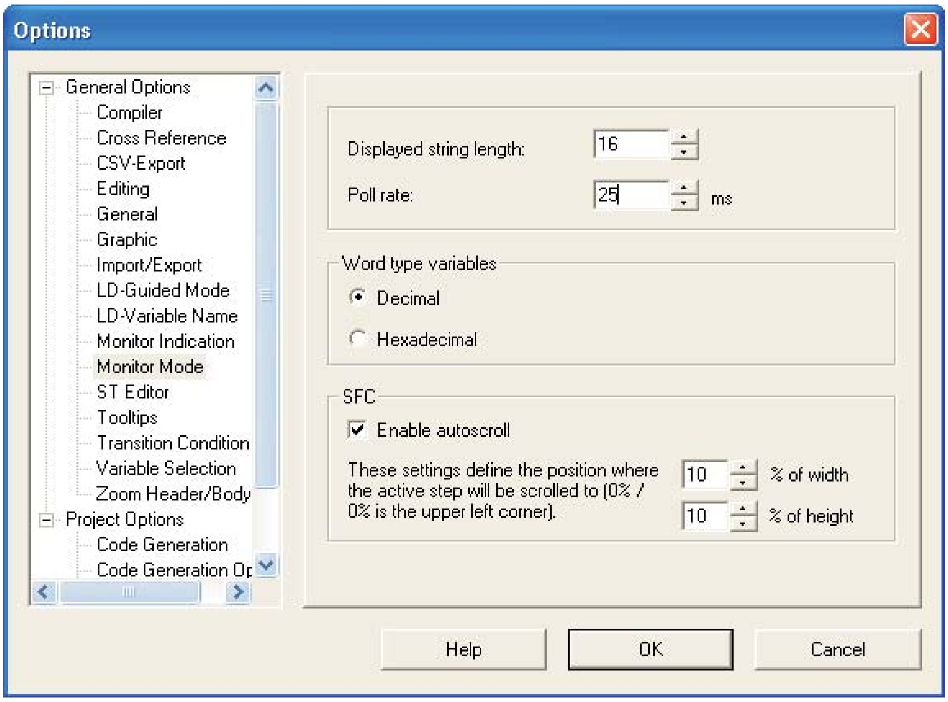
To begin monitoring the content of additional windows, click inside that window and select ***Start Monitoring*** from the ***Online*** Menu:



NOTE

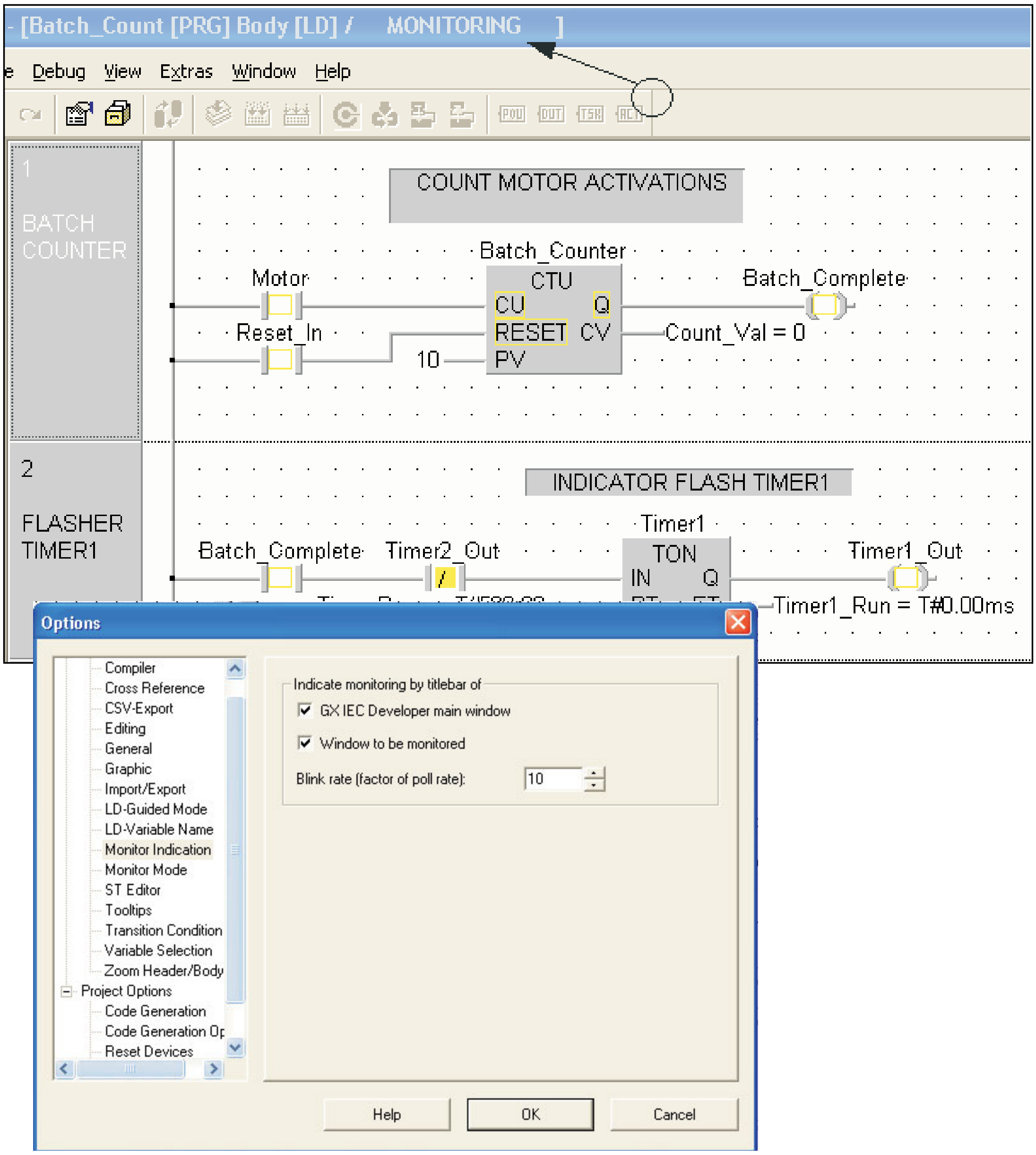
Due to the Serial Communications handshake, be prepared to wait a few seconds for the monitor information to be registered between GX IEC Developer and the PLC.

The rate of communication polling from GX IEC Developer to the PLC may be increased by adjusting the following parameters from the ***Extras*** / ***Options*** menu and select ***Monitor Mode***; alter the poll rate setting:



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:

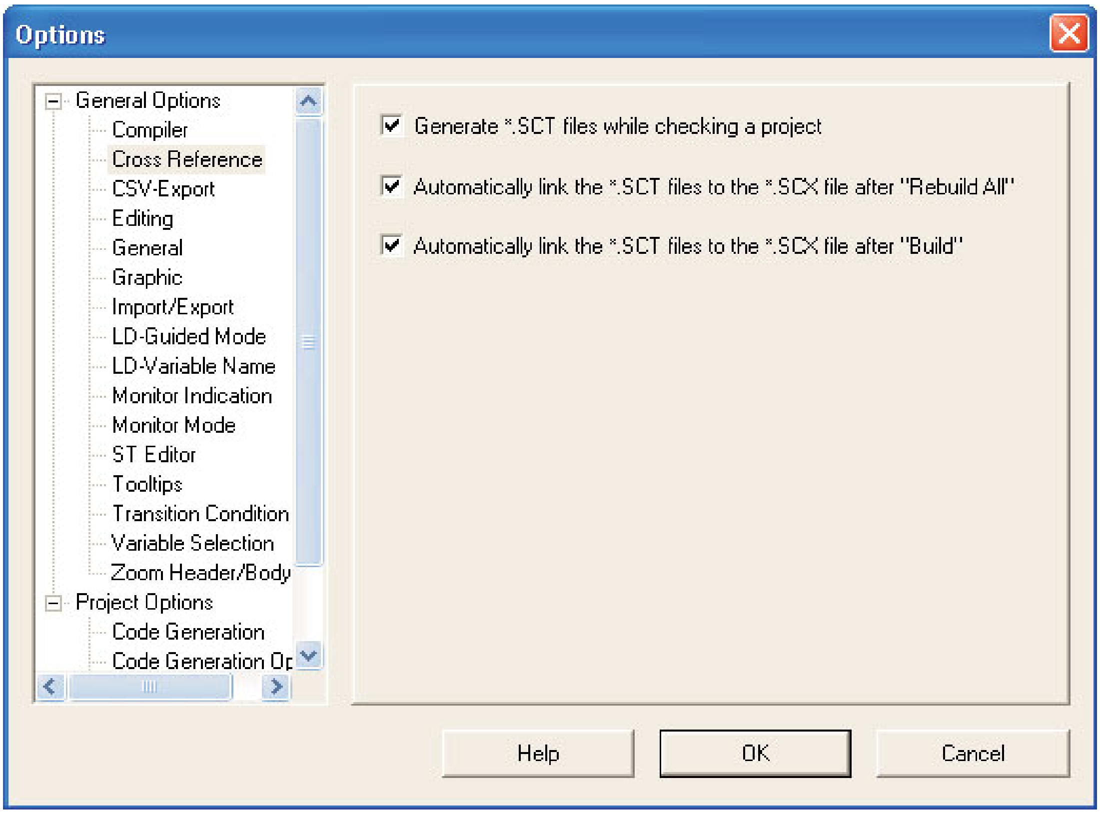


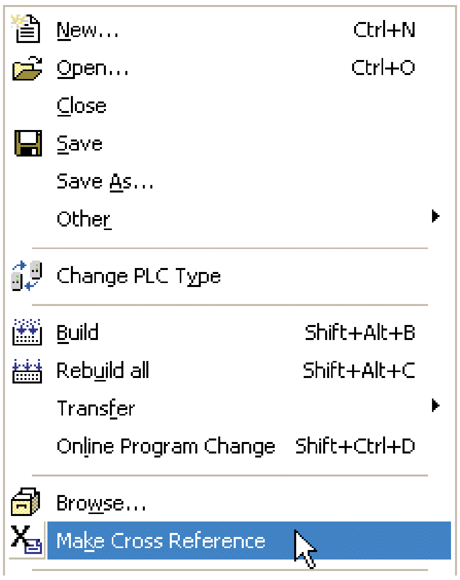
4.5 Cross Reference List

To generate a Cross Reference List:

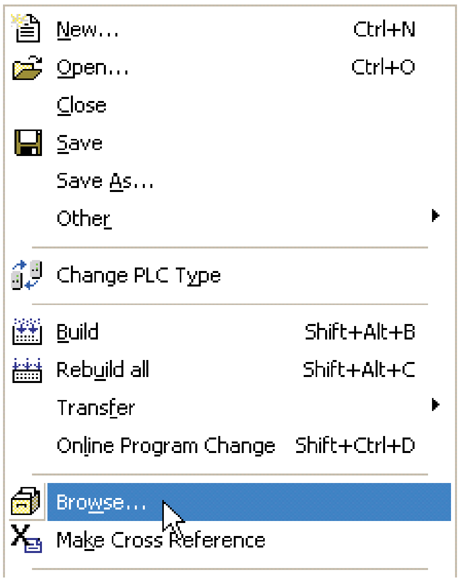
① Open the ***Extras/Options*** Menu and select ***Cross Reference***

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



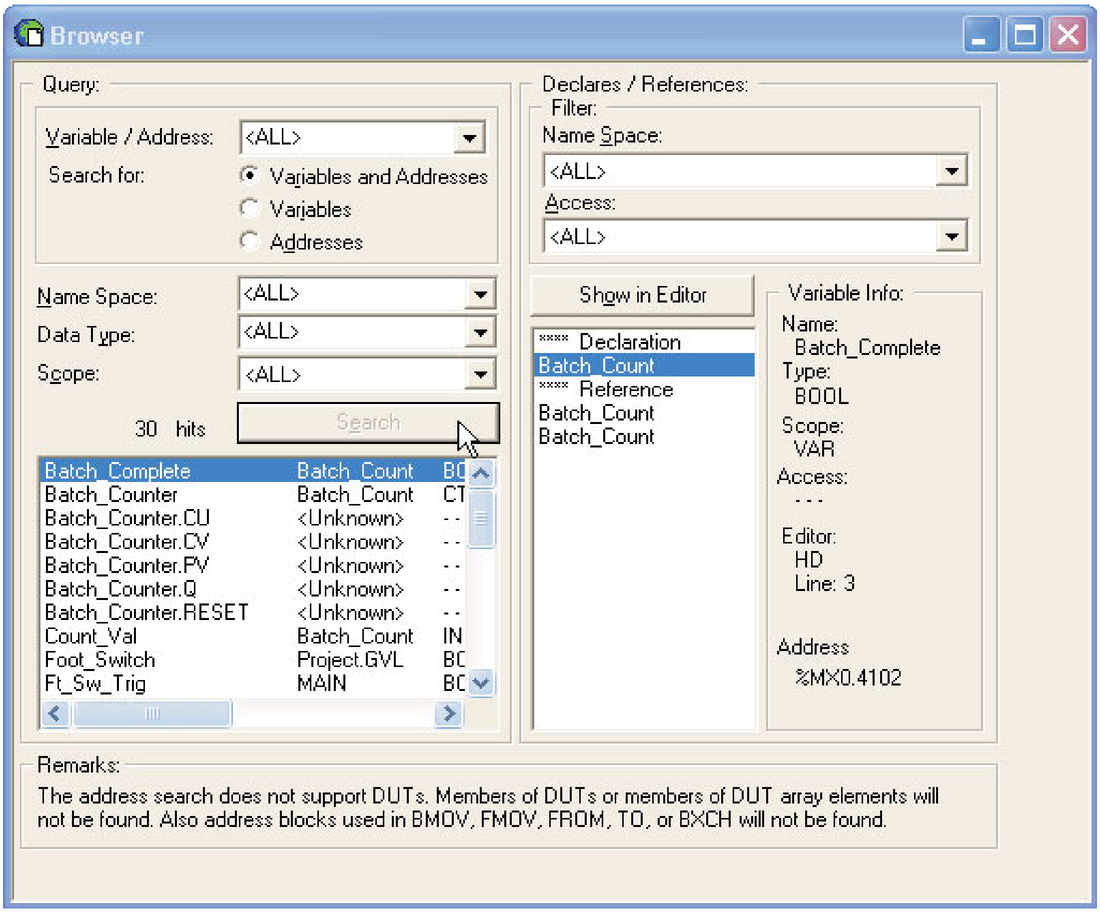


③ Then select ***Make Cross Reference*** from the ***Project*** Menu and the list is generated.



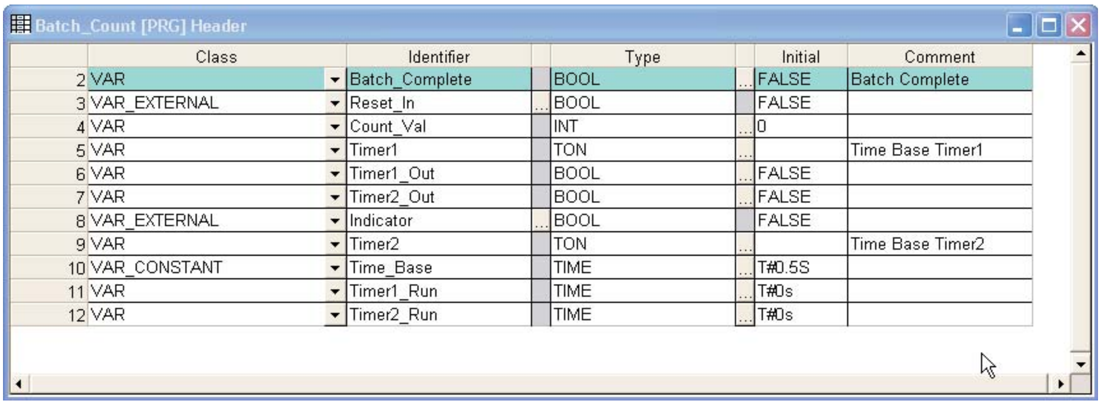
④ Open the Browser, either from the ***Project*** menu, or via the toolbar icon .

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



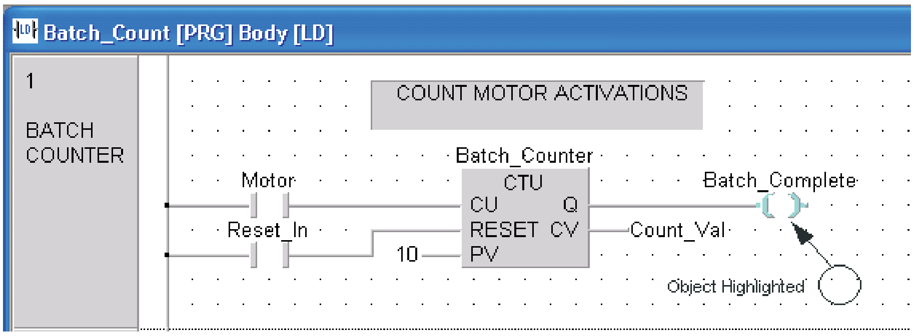
Specific variables etc can be searched by using the query selection boxes.Individual details of the highlighted entry are then shown on the right hand side of the window.

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



	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	

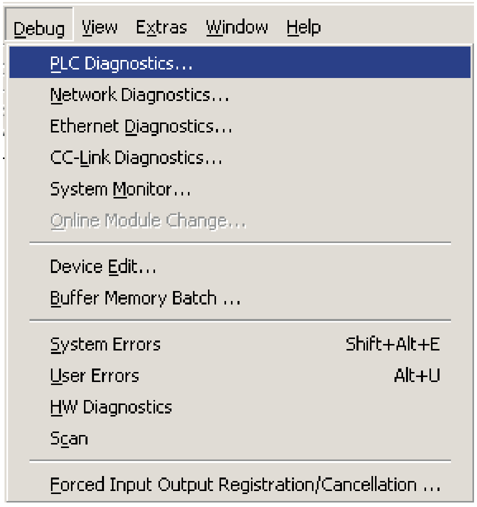
Or



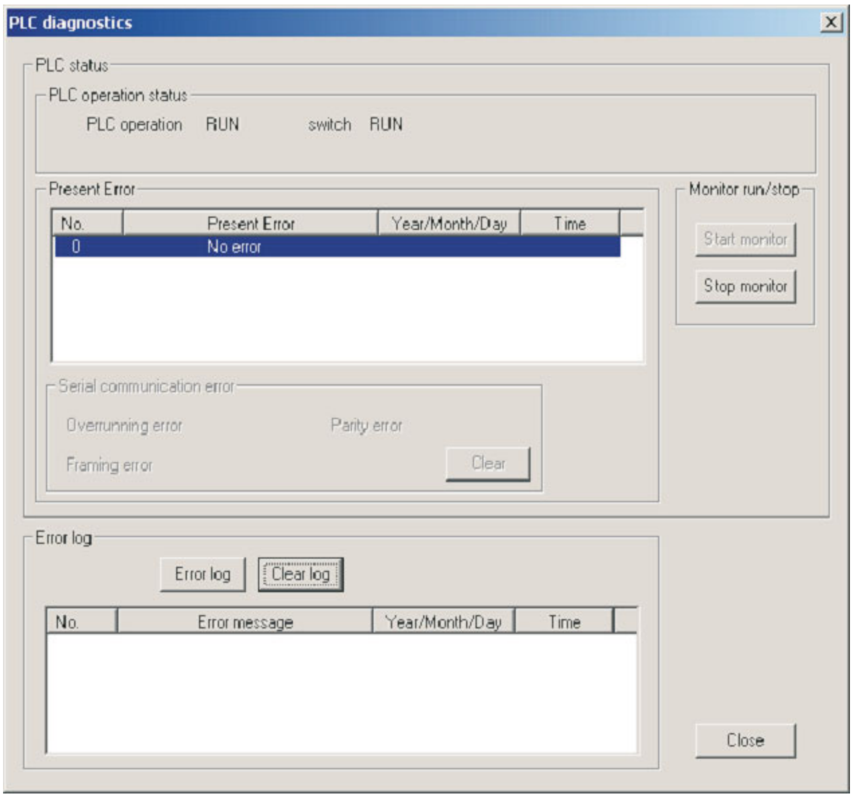
The Cross Reference List may be printed out, using the print facility within GX IEC Developer.

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.



Clear Text Error Message

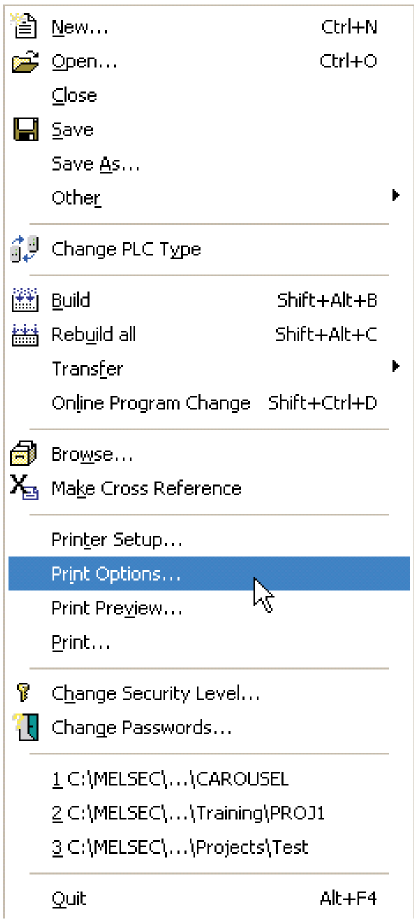
The error data registers of the PLC are evaluated with clear text and respective help texts.

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

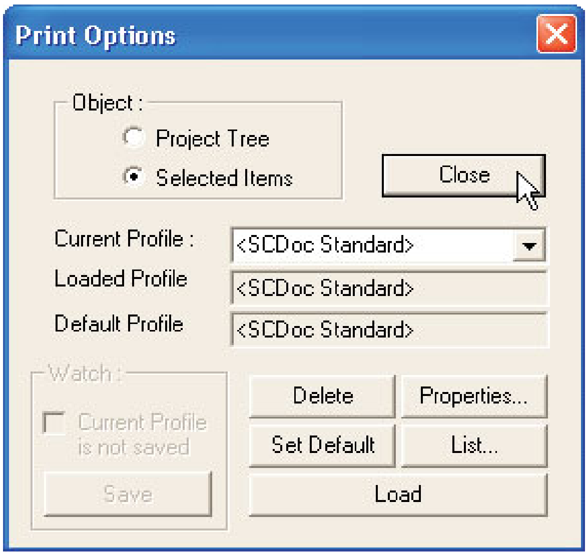
User errors can be determined. These user errors are stored with a self-created text file (USER_ERR.TXT) and allow a quick error correction. The last eight user errors are stored into a FIFO register and only be removed when they no longer occur.

4.7 Project Documentation

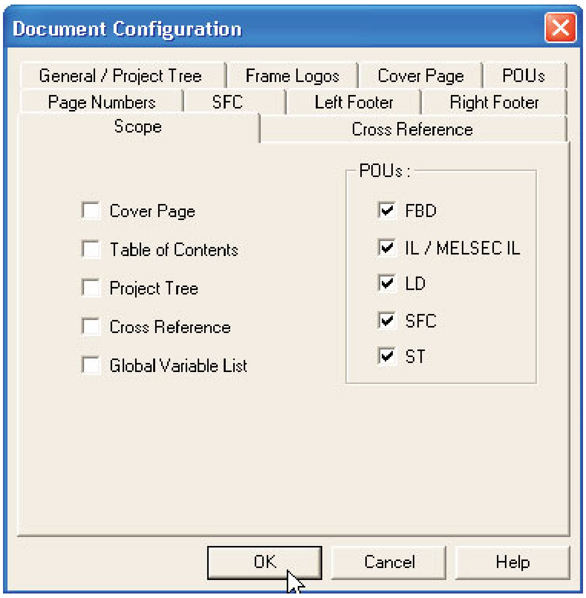
Project documentation can be set up using the ***Print Option*** facility from the ***Project*** Menu:



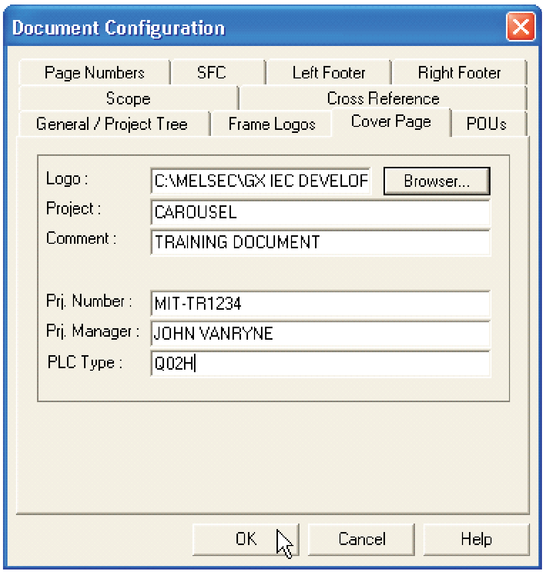
The “Change Configuration” dialogue box can then be seen. Previous project profiles can be retrieved here, or work with the default profile. Either select the ***Project Tree*** for all elements, or ***Selected Items*** for specific highlighted items, open ***Properties***:

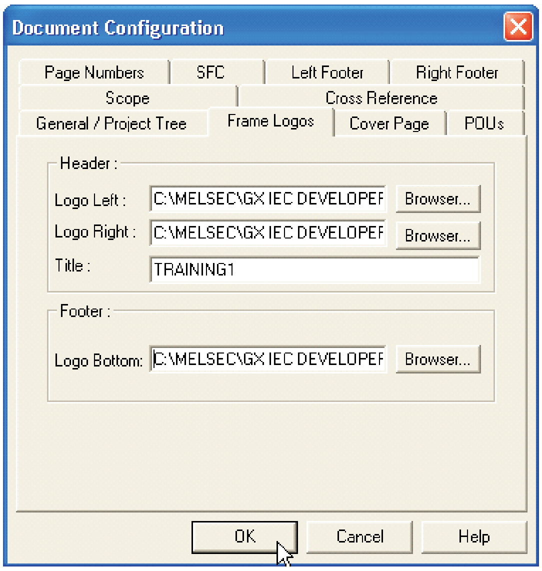


The ***Document Configuration*** folder is shown below. Select the tabs to configure the document as required. In this example, only the COUNTER_FB_CE will be printed, as the ***Selected Items*** option was chosen:

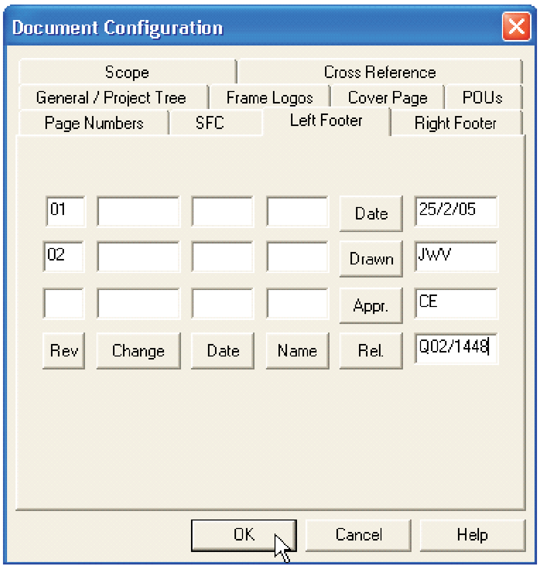


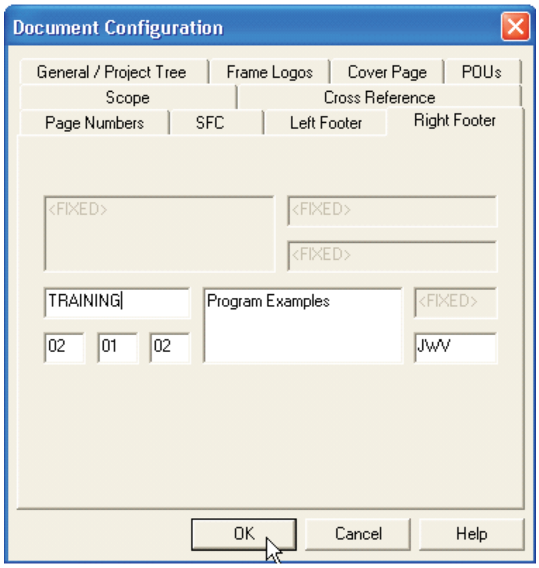
User defined logos and information can be assigned, in the ***Cover Page*** tab, for the front sheet and for the frame from the ***Frame Logos*** tab:



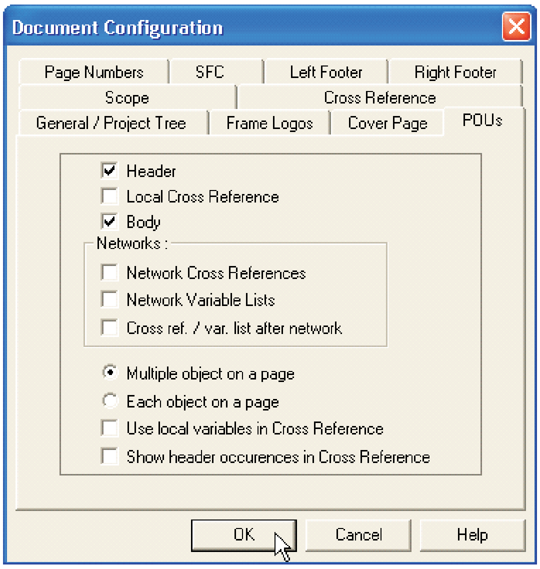


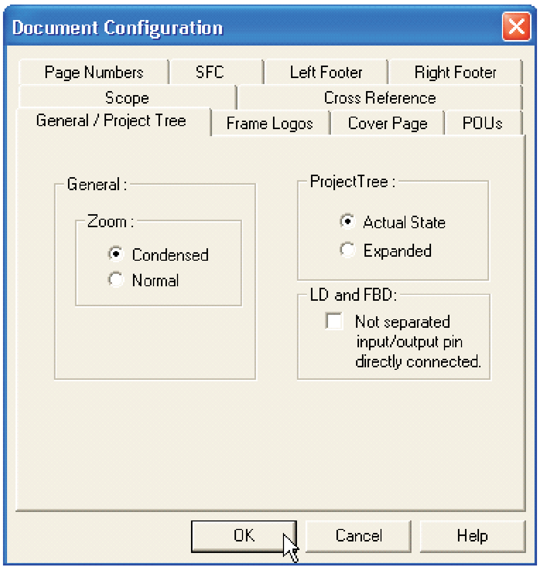
Detailed information can be assigned, to the left and right footers. The field labels in the ***Left Footer*** dialogue can be renamed, by clicking on the name buttons, as required:



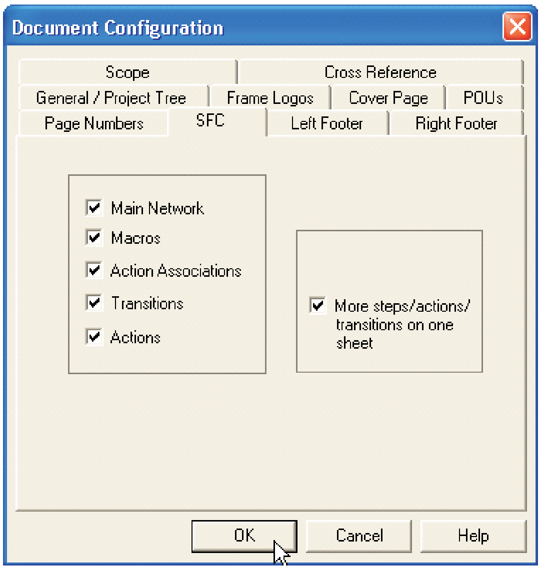


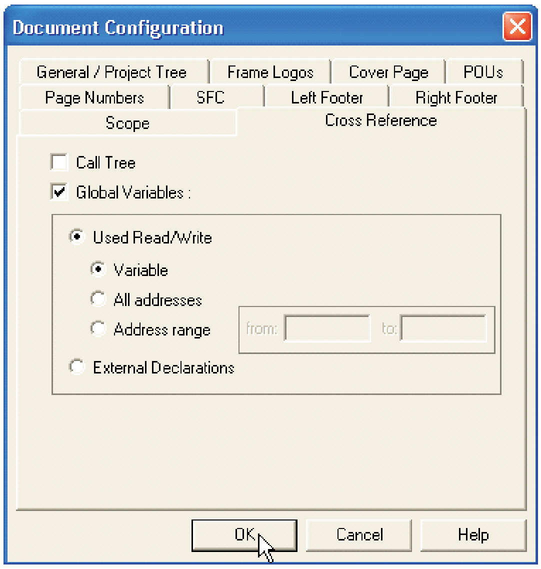
Specification for POU appearance and general project specifications are available from the ***POUs*** and ***General/Project Tree*** tabs.



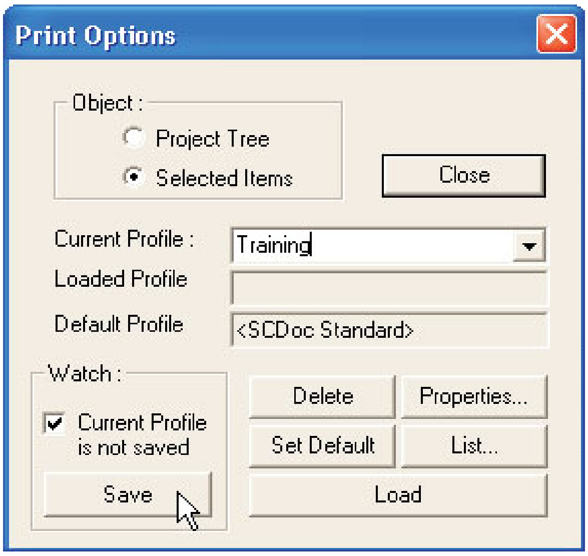


Specification for SFC appearance and cross reference specifications, are available from the ***SFC*** and ***Cross Reference*** tabs:





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:



5 Program Example

5.1 QUIZMASTER

Subjects covered:

- Timing
- Counting
- Logical Operations: Latching – Interlocks – Use of internal M device.
- Functional Instructions: Reset Function – Pulse Function

Description

A comprehensive automatic quiz game controller; Captures and latches the first player to activate respective ‘Answer Response Button’.

Only one contestant response lamp will be activated; all subsequent responses from other contestants are locked out.

Task

- Produce a PLC Ladder Diagram, which ensures that only one of the Contestant Indicator Lamps illuminates.
- When the chairman presses the Start Button, the contestants have a 10 second window to offer a response via their response push buttons.
- During the answer response period, the elapsed time (0 -10 Sec) is displayed on the analogue gauge of the training rig.
- The Chairman may reset the system at any time by using a separate button.

I/O List

Inputs

X10	-	Player1 Response Button
X11	-	Player2 Response Button
X12	-	Player3 Response Button
X13	-	Player4 Response Button
X14	-	Chairman Start Timing
X15	-	Reset Game

Outputs

Y20	-	Player1 Answer Lamp
Y21	-	Player2 Answer Lamp
Y22	-	Player3 Answer Lamp
Y23	-	Player4 Answer Lamp

Y24	-	Time-up Indication
Y25	-	Question Timing

Intelligent Function Module

U4\G1 - Address of buffer memory for analogue output channel 1

5.1.1 Method

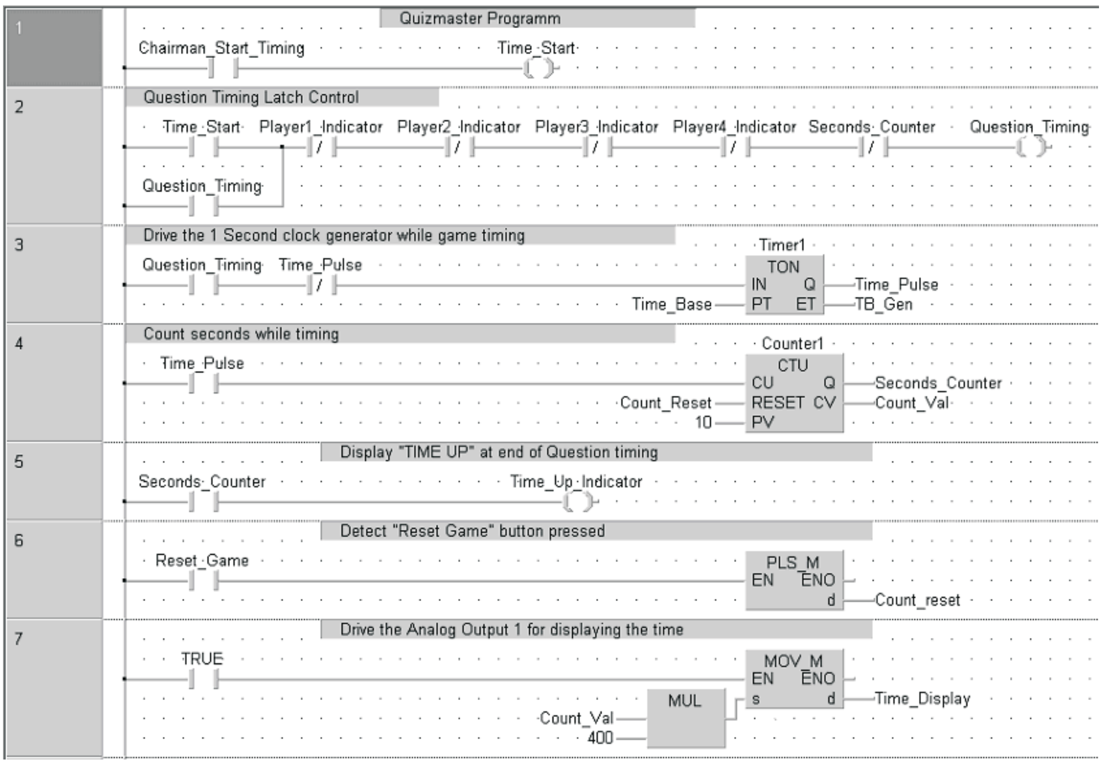
① Create a new Project and name it “Quizmaster”.

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

	Class	Identifier	MIT-Addr.	IEC-Addr.	Type	Initial
0	VAR_GLOBAL	Player1_Response	X10	%IX16	BOOL	FALSE
1	VAR_GLOBAL	Player2_Response	X11	%IX17	BOOL	FALSE
2	VAR_GLOBAL	Player3_Response	X12	%IX18	BOOL	FALSE
3	VAR_GLOBAL	Player4_Response	X13	%IX19	BOOL	FALSE
4	VAR_GLOBAL	Chairman_Start_Timing	X14	%IX20	BOOL	FALSE
5	VAR_GLOBAL	Reset_Game	X15	%IX21	BOOL	FALSE
6	VAR_GLOBAL	Player1_Indicator	Y20	%QX32	BOOL	FALSE
7	VAR_GLOBAL	Player2_Indicator	Y21	%QX33	BOOL	FALSE
8	VAR_GLOBAL	Player3_Indicator	Y22	%QX34	BOOL	FALSE
9	VAR_GLOBAL	Player4_Indicator	Y23	%QX35	BOOL	FALSE
10	VAR_GLOBAL	Question_Timing	Y24	%QX36	BOOL	FALSE
11	VAR_GLOBAL	Time_Display	U4\G1	%MW14.4.1	INT	0
12	VAR_GLOBAL	Time_Up_Indicator			BOOL	FALSE

③ Create a new POU of Class ***PRG*** (Program Type) and Language ***Ladder Diagram*** and name it “Game_Control”.

④ Enter the following code into the POU.

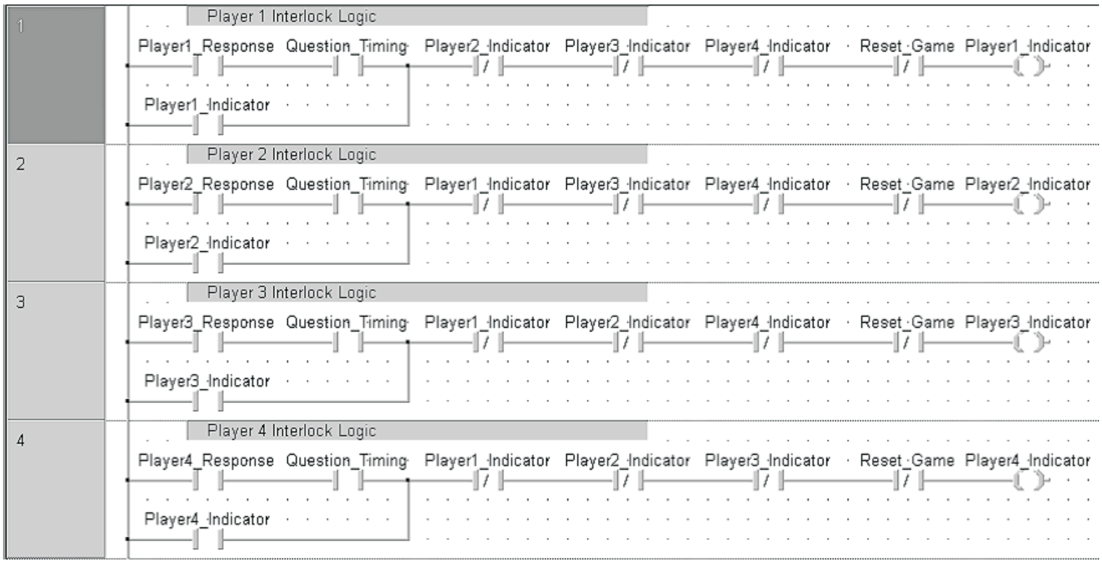


The finalised Header of the “Game_Control” POU should read as follows

	Class	Identifier	Type		Initial	Comment
0	VAR	Time_Start	BOOL	...	FALSE	
1	VAR	Time_Pulse	BOOL	...	FALSE	
2	VAR	TB_Gen	TIME	...	T#0s	
3	VAR_CONSTANT	Time_Base	TIME	...	T#1s	
4	VAR	Count_Reset	BOOL	...	FALSE	
5	VAR	Seconds_Counter	BOOL	...	FALSE	
6	VAR	Count_Val	INT	...	0	
7	VAR	Timer1	TON	...		
8	VAR	Counter1	CTU	...		
9	VAR	Config_Analog	BOOL	...	FALSE	

⑤ Create a new POU of Class ***PRG*** and of Type ***Ladder*** and name it “Player_Logic”

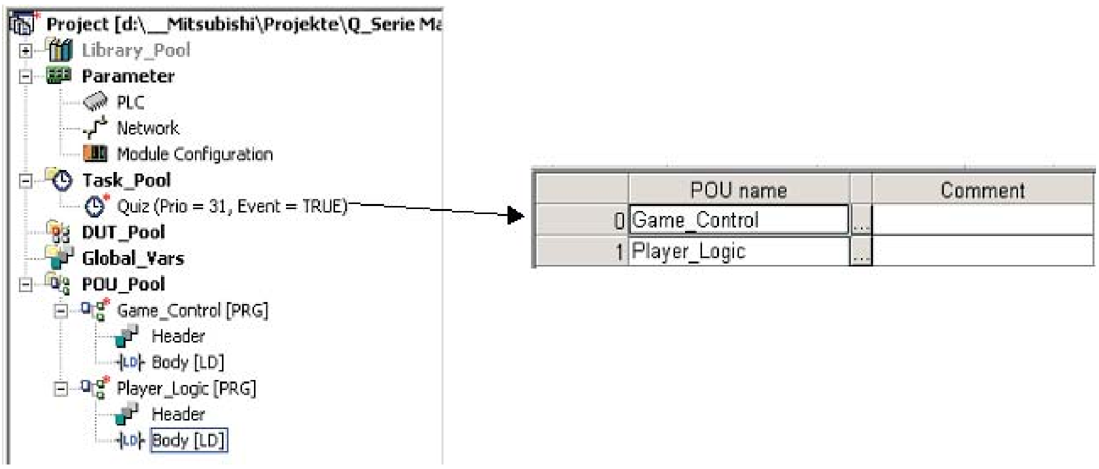
⑥ Enter the following Ladder code into the new POU:



The finalised Header of the “Player_Logic” POU should read as follows:

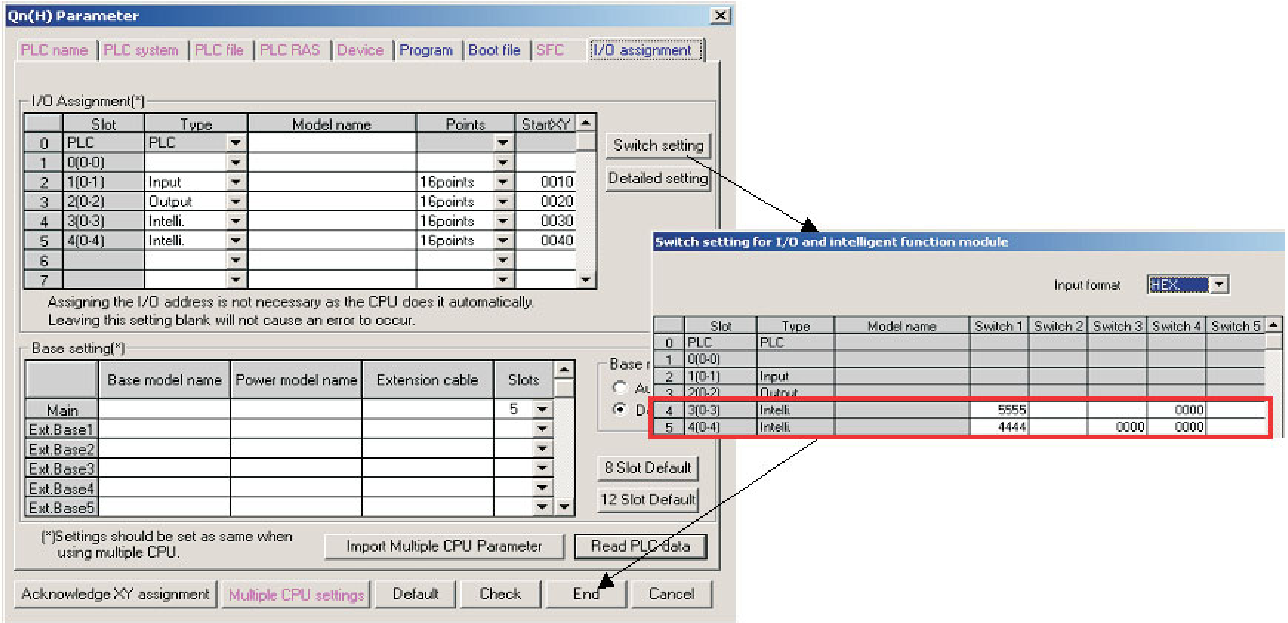
	Class	Identifier	Type		Initial	Comment
0	VAR			...		

⑦ Create a new Task in the Task Pool “QUIZ”. Bind the POU’s, “Player_Logic” and “Game_Control” respectively into the new task as shown below:

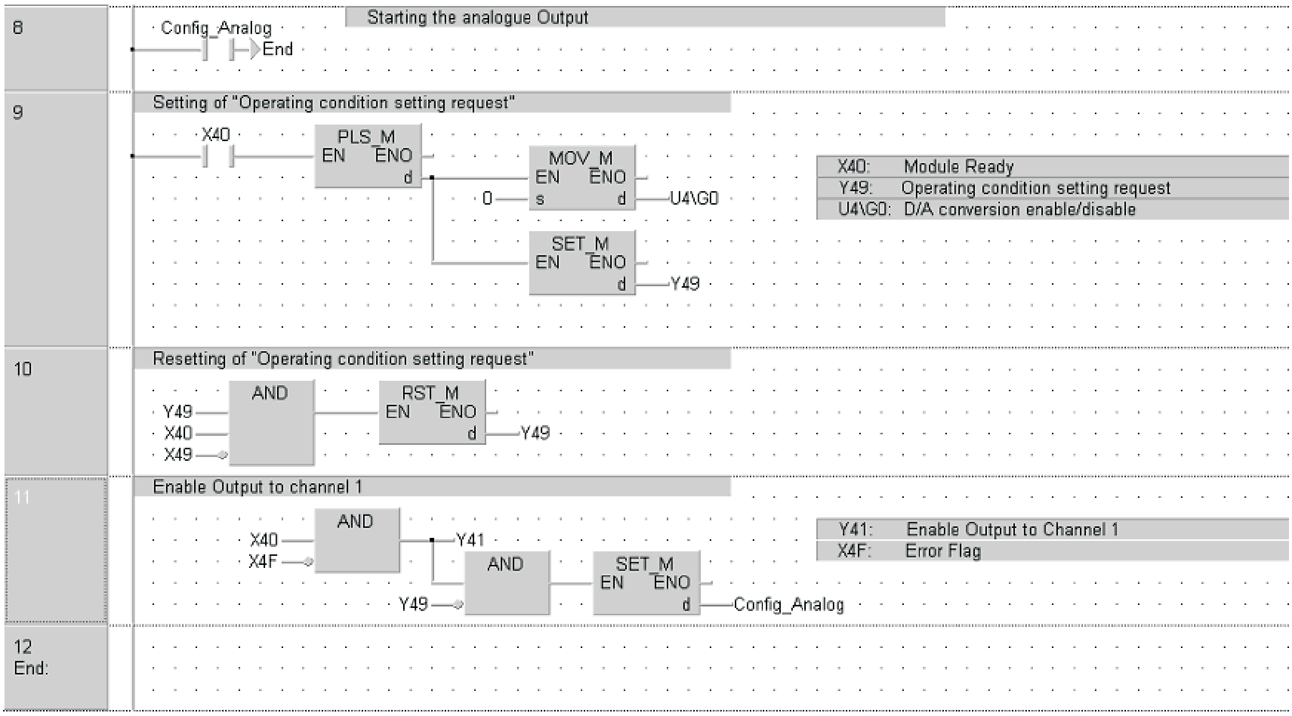


Initialization of the Analogue Output Module

⑧ To initialize the Q64DA Analogue output module, the following setting must be done in ***Parameter*** -> ***PLC*** -> ***I/O assignment***.



⑨ Add the following networks to the POU “Game_Control” to start the analogue output to channel 1, which is connected to the gauge.



5.1.2 Quizmaster - Principle of Operation

① Enter, Test and Save the project “Quizmaster” including annotation.

② Download the project to the Q-SERIES PLC.

③ Ensure the project is working correctly by monitoring the operation while operating the inputs.

④ Momentarily switch input X14 to begin contestant answer response timing.

⑤ Wait for initial contestant response from X10, X11, X12 or X13 and latch appropriate contestant indicator. Lock out further operation of all inputs.

⑥ While waiting for response, run response timer for a period of 10 Seconds and present running time on display.

⑦ At end of time period, lock any further action from all contestant response inputs, stop the time display and illuminate ‘Time Up’ indicator.

⑧ Wait for chairman to activate ‘Reset’ input X5, in order to clear all game status flags and outputs, so as to begin a new round.

⑨ Go back to step 1 or end game.

5.1.3 Quizmaster Program Description

POU “Game_Control”

- Network 1

  When the Chairman Start Timing button is pressed, Local Variable “Time_Start is pulsed via the PLS_M instruction.

- Network 2

  Question_Timing is latched providing that no player Indicators are on and the seconds counter is not counting.

- Network 3

  The Question_Timing contact enables the 1Second Time base cut throat timer to run. 1 second pulses are generated on the “Time_Pulse” output.

- Network 4

  The pulses from the Time_Pulse flag are counted using a CTU “Count UP” counter, which counts for 10 second period.

- Network 5

  When the Seconds_Counter flag operates, the Time_Up_Indicator activates and is illuminates the lamp.

- Network 6

  When the “Reset_Game” input is activated, a pulse is generated to provide a pulse to reset the seconds counter in network 7 below.

- Network 7

  The TRUE input is “always on”, therefore the Count_Val multiplied with the offset of 400 digits/Volt is sent permanently as “Time_Display” to the analogue output module.

POU “Player_Logic”

- Networks 1- 4

  These routines control the player interlocks. For example if player 1 is the first to operate his of her response button, then that Lamp illuminates and locks out all subsequent responses from other players. Each player control logic routine lock out other subsequent player responses. Players can only offer a response when the "Question_Timing" flag is active.

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

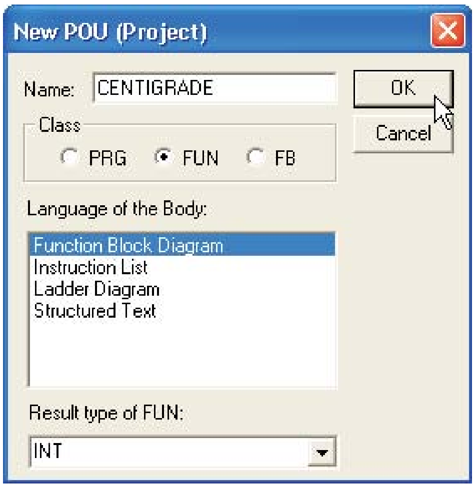
Formula is:

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

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

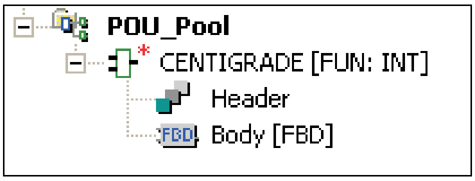
Procedure

① Select a new POU and name it Centigrade.

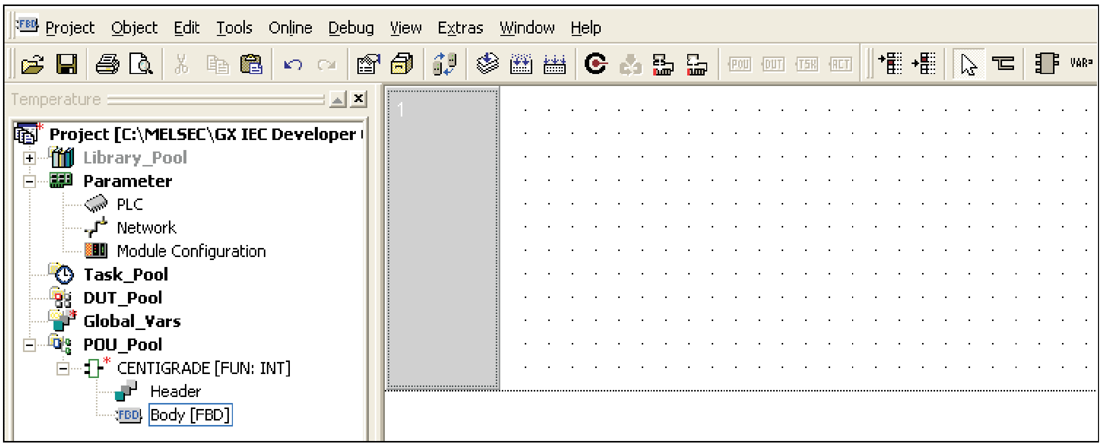


This time click the “FUN” option, instead of “PRG.” Select ***Function Block Diagram*** as the editor. The Result Type of FUN should be left as INT (Integer type).

Centigrade will now have appeared on the POU tree:

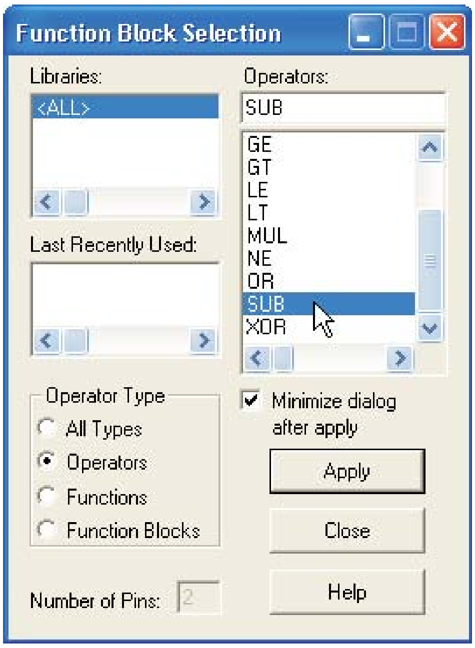


② Double click on the FBD body icon, to open the body network:

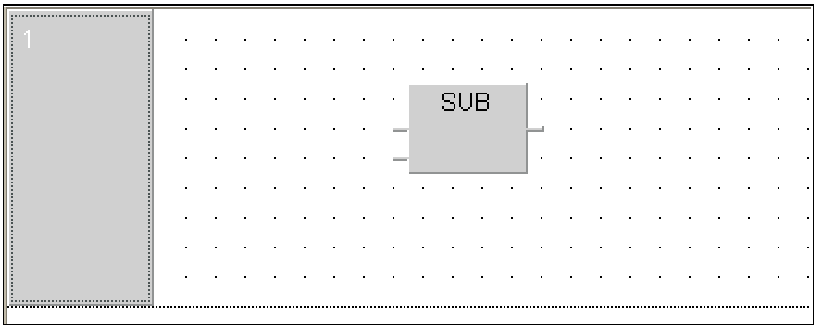


Selecting the Function:

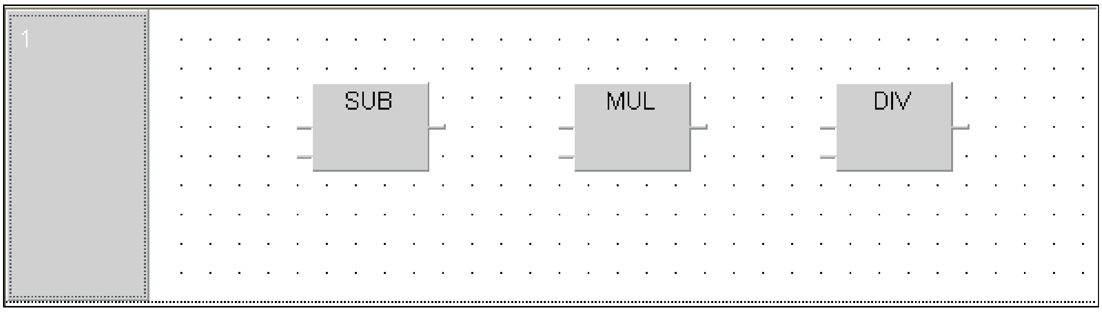
① Select the **function block** icon  from the toolbar and select ***SUB*** from the operators list:



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



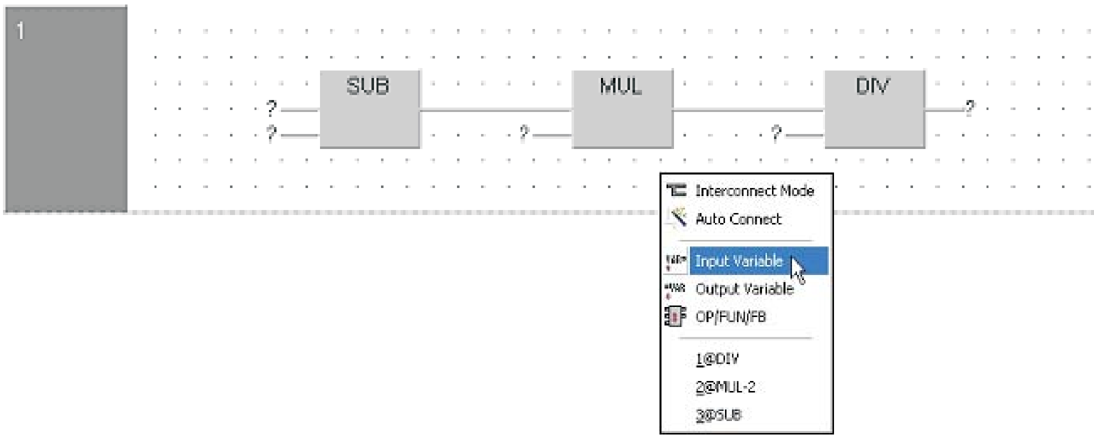
③ Repeat the above process so that the following is visible:



Declaring the Variables

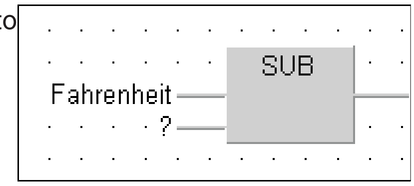
There are a variety of methods available to declare variables. The following procedure illustrates how to declare variables from the body of the FBD:

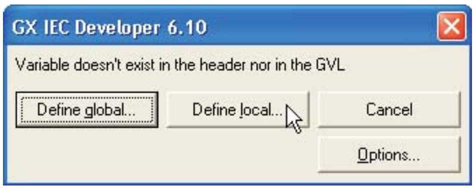
① Place input and output variables by right clicking the mouse in the work area. From the following popup menu, select and place input and output variable tags onto the FBD as shown below:



Alternatively, click on the toolbar button .

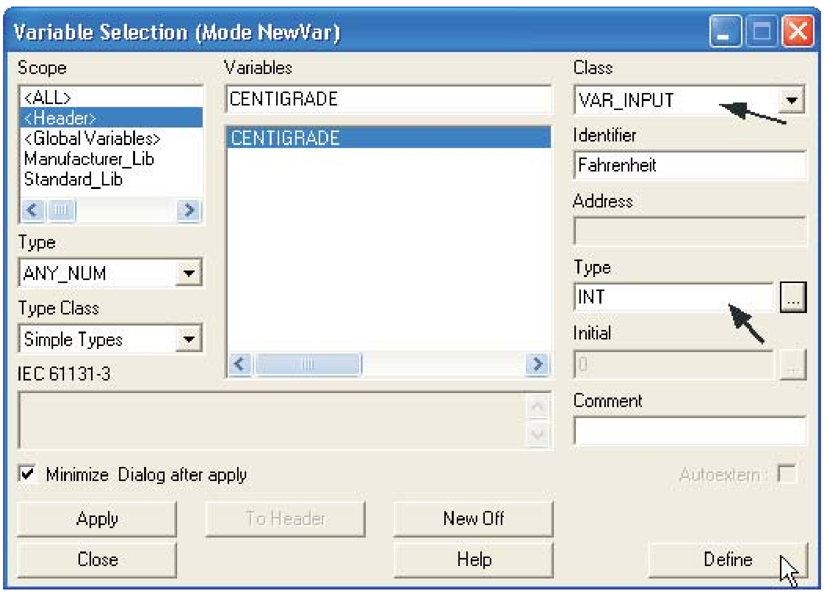
② Declare the variable “Fahrenheit” by simply typing it into the variable area:





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:



NOTES

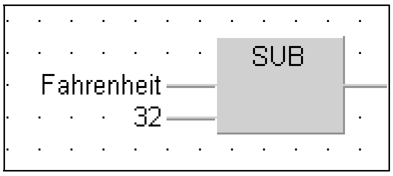
The Class VAR_INPUT is required as this variable enables values to be input into the function when it is connected as part of a program. It will produce a left hand pointing input connection point on the function symbol.

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

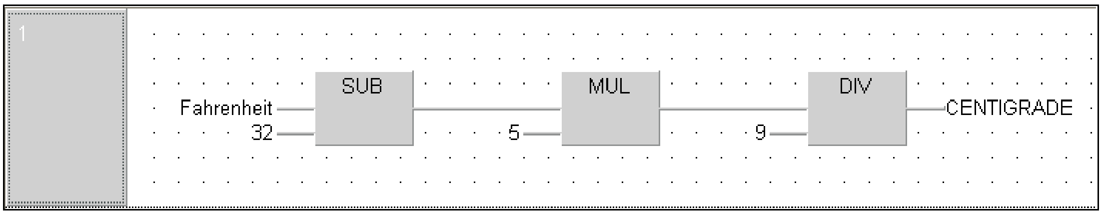
④ Click ‘Define’ and the variable will be written to the header of the Function ‘CENTIGRADE’. You can check it by opening the header.

Declaring Constants

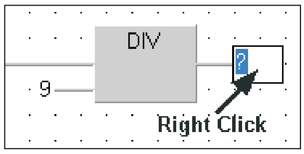
① Declare constant “32” by simply typing the number into the variable box:



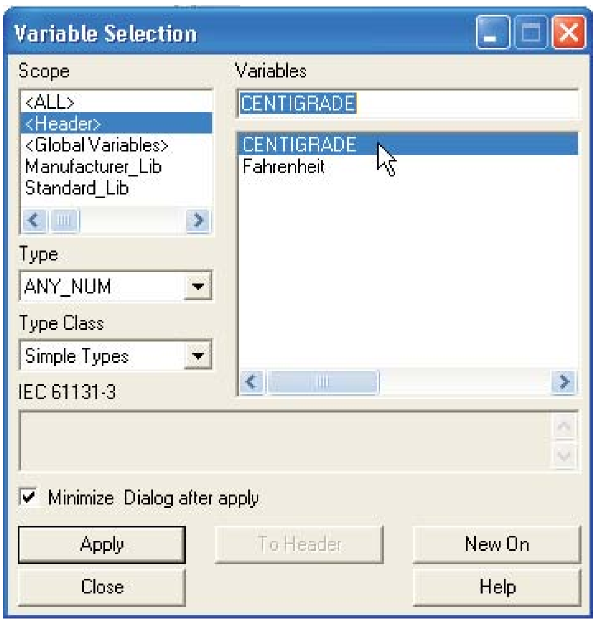
② Complete the circuit of the Function CENTIGRADE as follows:



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.

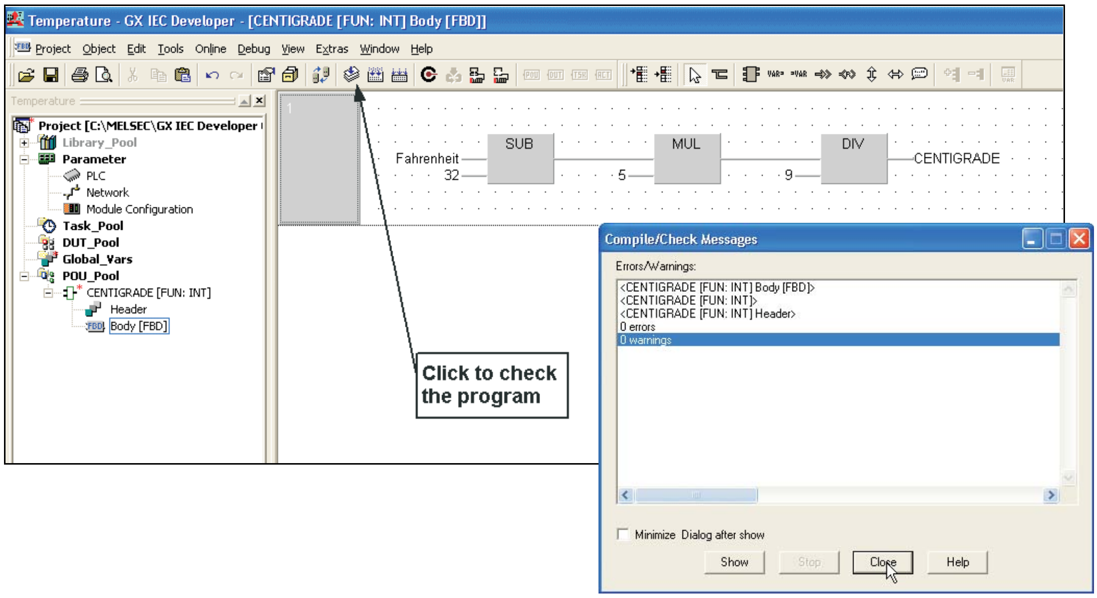
If desired, to clarify correct check the Header of the Function ‘CENTIGRADE’; it should appear as follows:

	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

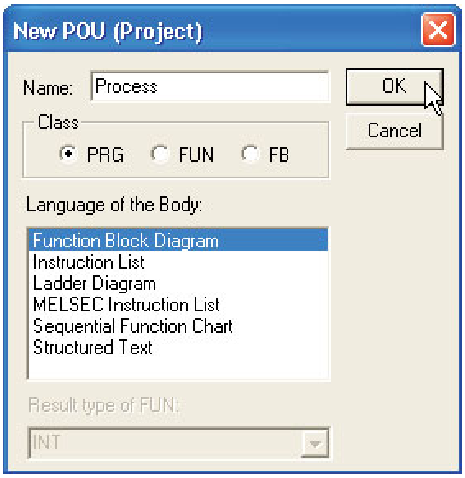
① Check the Network; you should have no errors and no warnings!



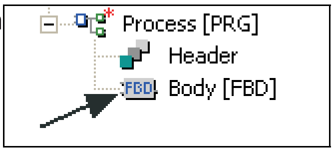
② Close down all work windows and any dialogues that may be open.

Creating a New Program POU

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

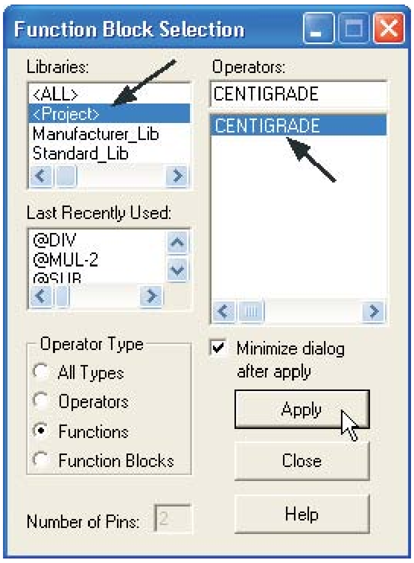


② Open up (Double Click) the body of Ladder POU "Process" in the project POU pool.



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:

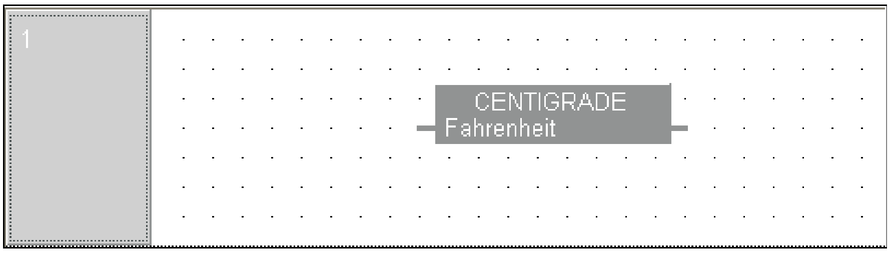


② Select CENTIGRADE and click 'Apply'.

NOTE

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

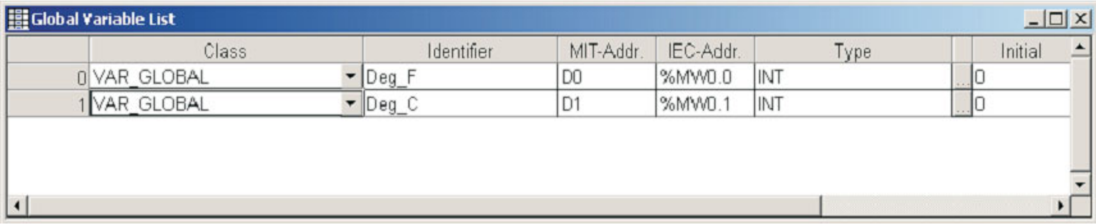
The following will be displayed:



Assigning the Global Variables

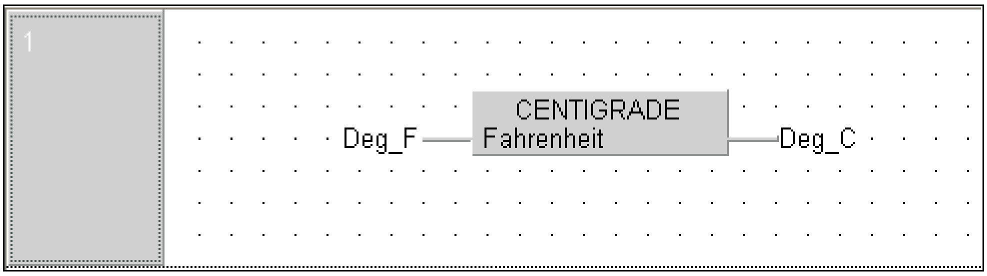
Once the function is placed on the new network assign variables to it.

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

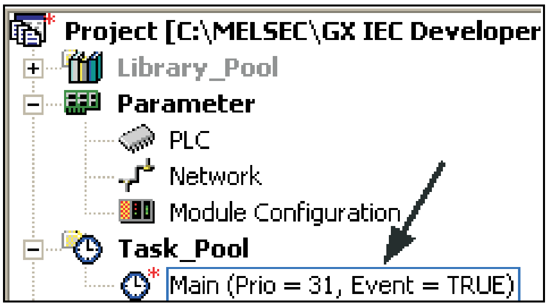


	Class	Identifier	MIT-Addr.	IEC-Addr.	Type	Initial
0	VAR_GLOBAL	Deg_F	D0	%MW0.0	INT	0
1	VAR_GLOBAL	Deg_C	D1	%MW0.1	INT	0

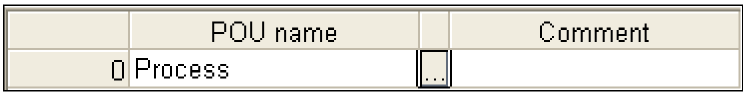
The Body of the POU “Process" should read:



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



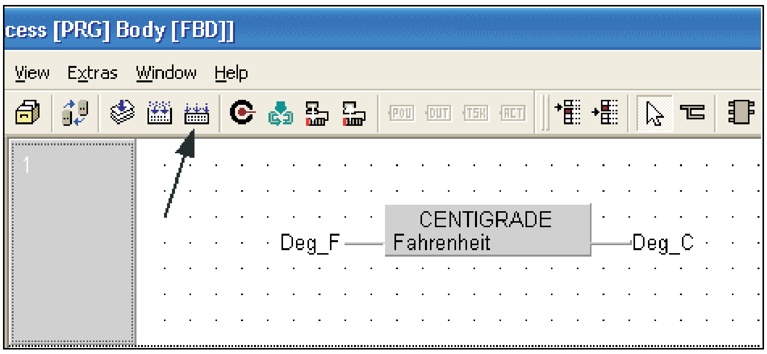
③ Bind the POU “Process” to the Task “Main”:



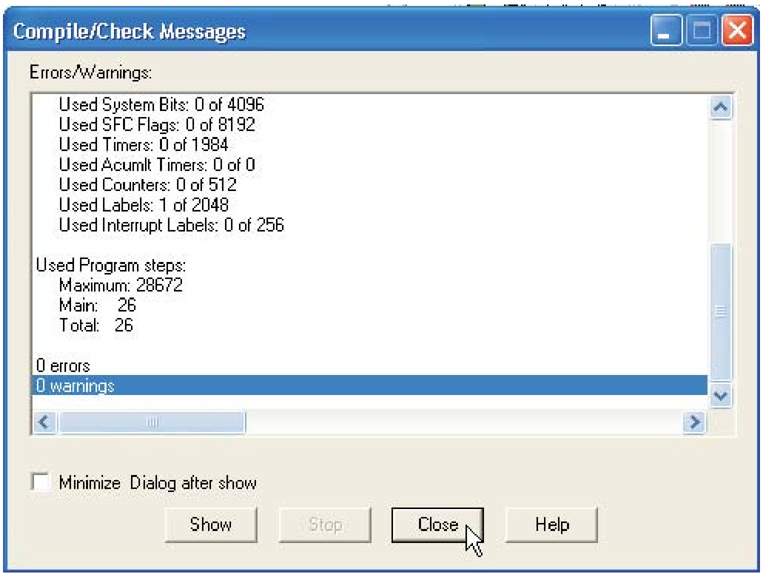
	POU name		Comment
0	Process	...	

Compiling the Program

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



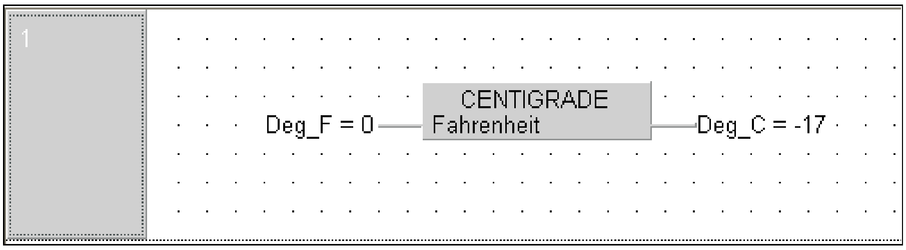
Following compilation the following should be displayed:



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

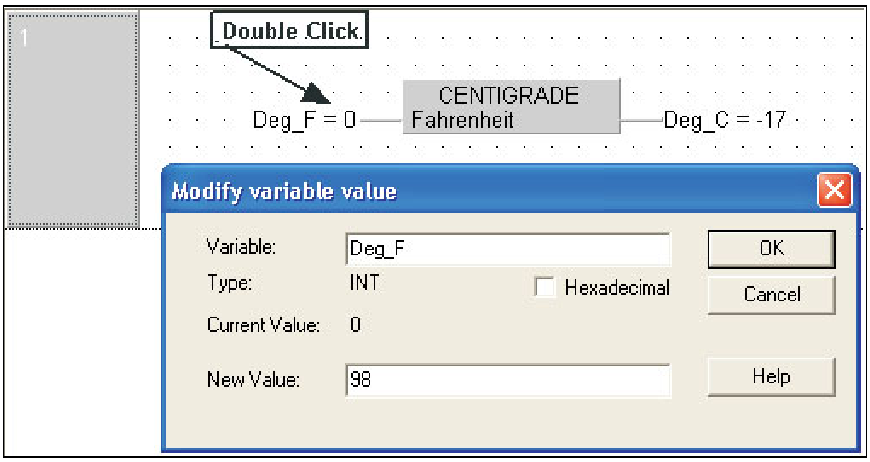
Monitoring the program

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:



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

6.1.2 Processing Real (Floating Point) Numbers

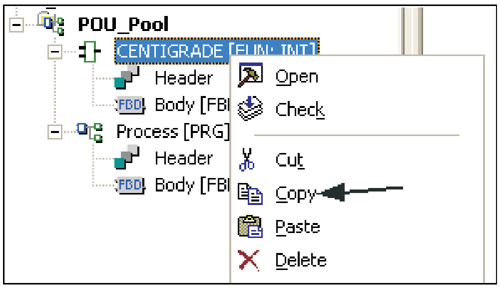
The existing CENTIGRADE function currently can only process 16 Bit Integer Whole Number (+32767 to -32768) values which is the numeric system default when creating Functions. The following example will utilise the Function ‘CENTIGRADE’, modifying it to process “REAL” floating point values*.

* Only valid on processors supporting this feature.

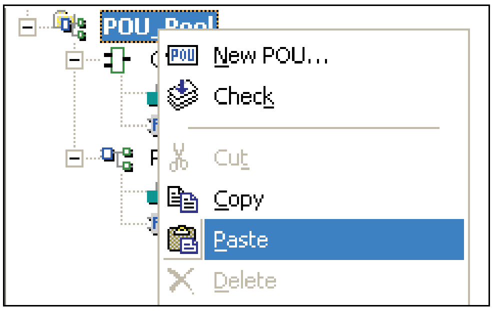
Duplicating a Function

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

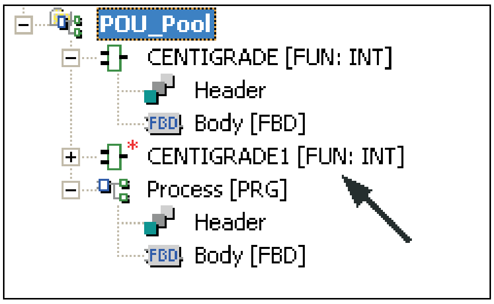
① Right Click on the CENTIGRADE Icon in the POU Pool of the project and select ***Copy***.



② Right Click on the POU Pool Icon of the project and select ***Paste***.

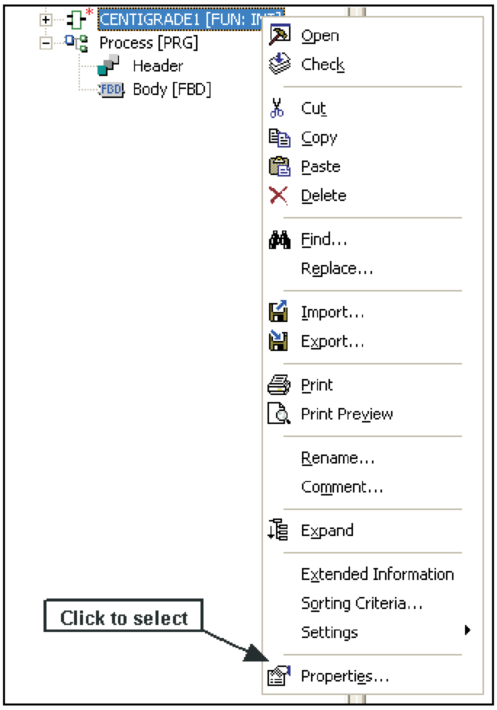


The system will automatically paste a duplicate copy of ‘CENTIGRADE’ and rename it to ‘CENTIGRADE1’:

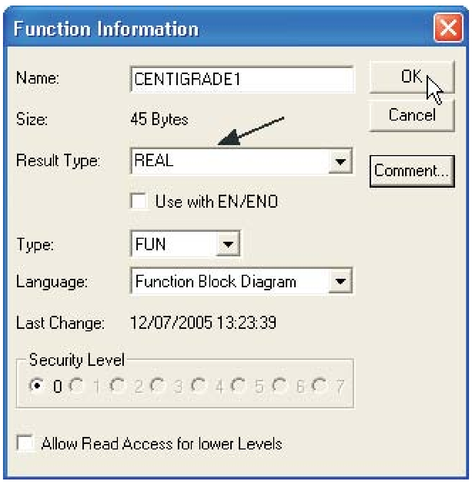


Changing the Result type of a Function

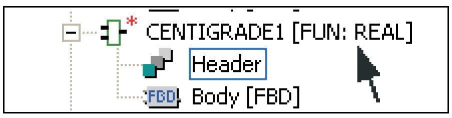
① Right Click on the newly created Function ‘CENTIGRADE1’ and click on ***Properties***.



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



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

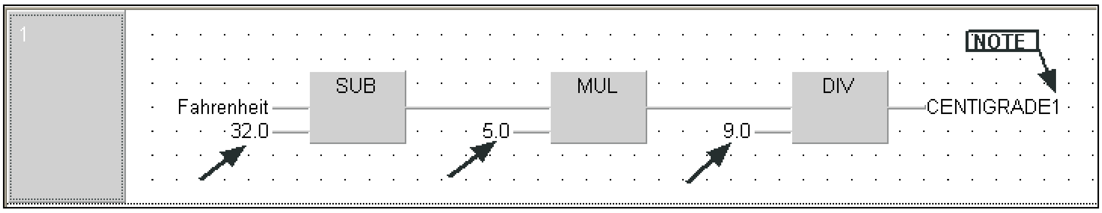


③ Modify the Header of CENTIGRADE1 so that the Fahrenheit variable is of type ‘REAL’:

	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:

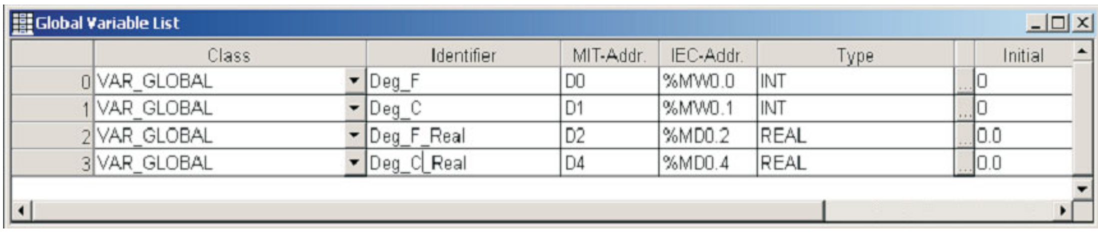


NB: Remember to alter CENTIGRADE to CENTIGRADE1.

② Close editors and save all changes.

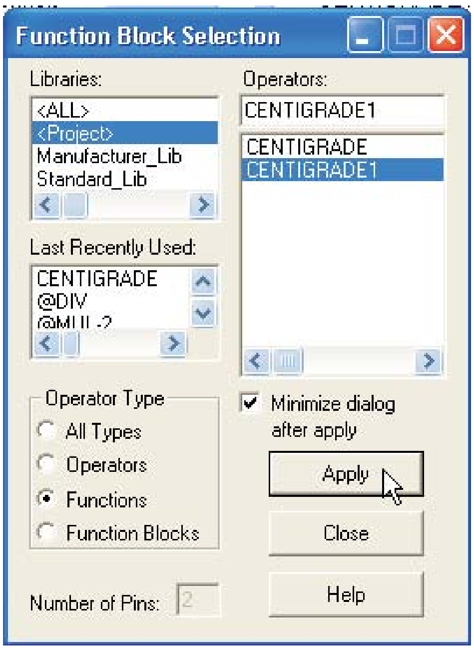
Placing the "REAL" number Function 'CENTIGRADE1' onto the working POU "Process"

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


Global Variable List

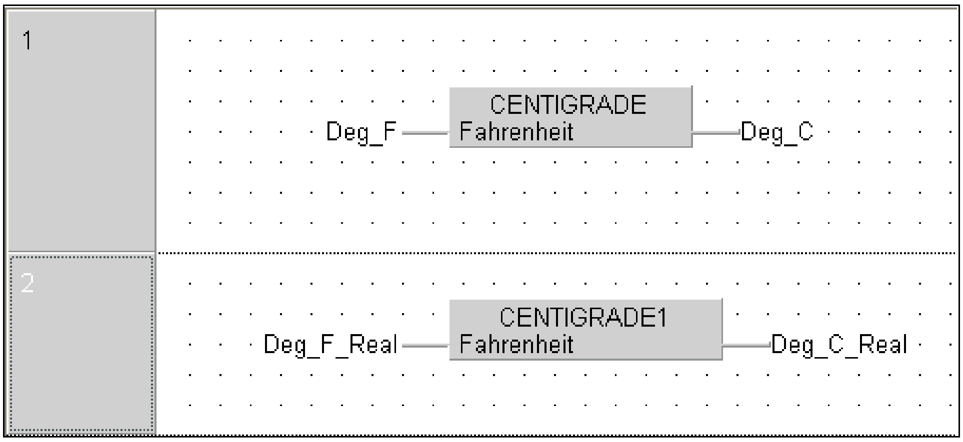
	Class	Identifier	MIT-Addr.	IEC-Addr.	Type	Initial
0	VAR_GLOBAL	Deg_F	D0	%MW0.0	INT	0
1	VAR_GLOBAL	Deg_C	D1	%MW0.1	INT	0
2	VAR_GLOBAL	Deg_F_Real	D2	%MD0.2	REAL	0.0
3	VAR_GLOBAL	Deg_C_Real	D4	%MD0.4	REAL	0.0

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



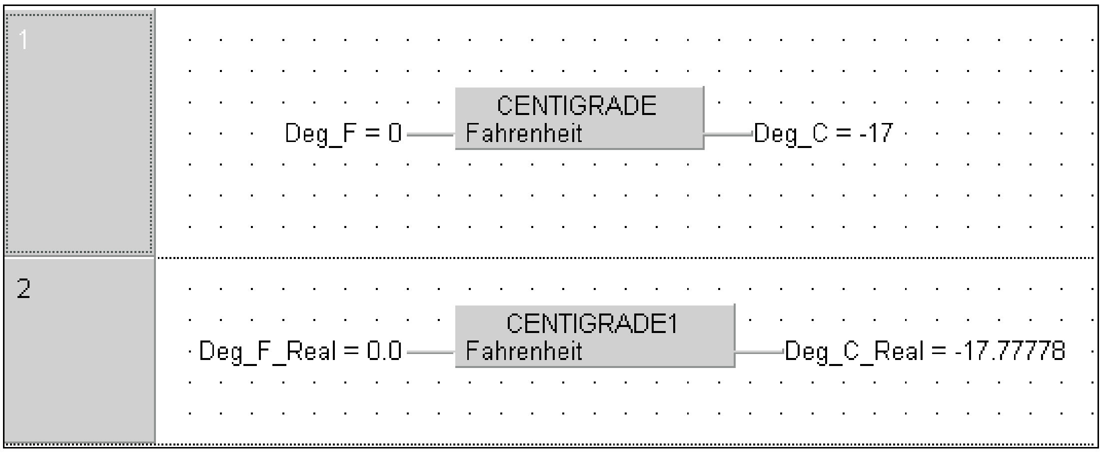
NOTE REAL numbers use 2 consecutive Registers (32 Bits) and are stored in a special portable IEE format, hence the allocation in the above GVL example.

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**

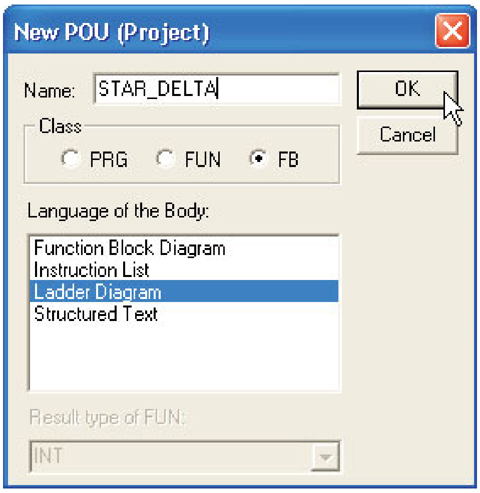
– Delta Contactor Output: **DELTA_COIL**

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***.



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.

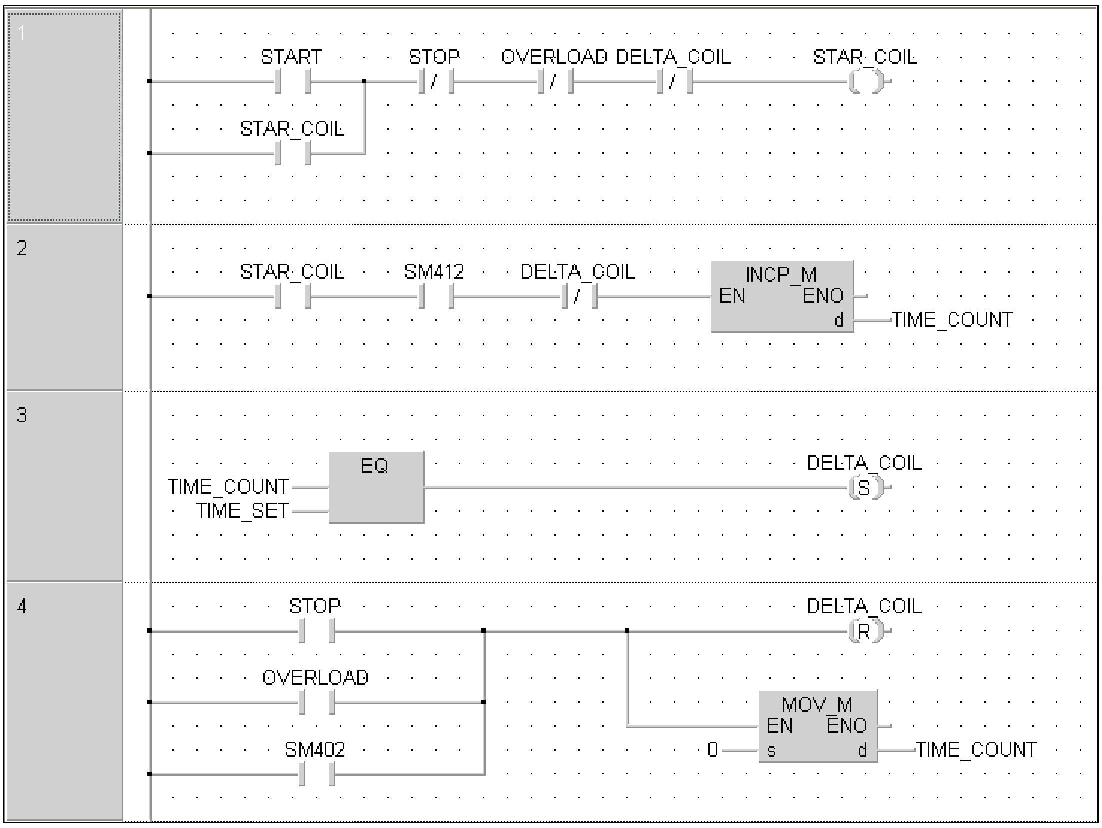
Declaring Local Variables

① Declare variables as shown below.

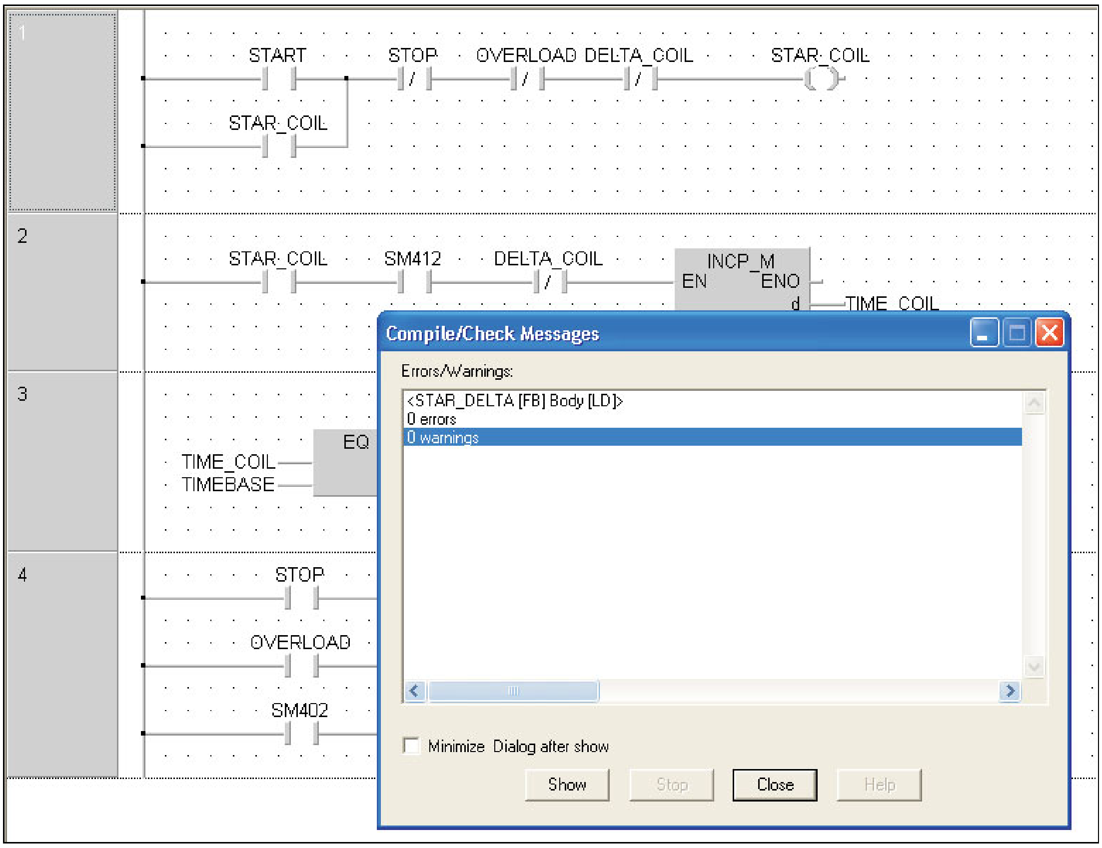
	Class	Identifier	Type	Initial	Comment
0	VAR_INPUT	START	BOOL	FALSE	
1	VAR_INPUT	STOP	BOOL	FALSE	
2	VAR_INPUT	OVERLOAD	BOOL	FALSE	
3	VAR_INPUT	TIME_SET	INT	0	
4	VAR_OUTPUT	DELTA_COIL	BOOL	FALSE	
5	VAR_OUTPUT	STAR_COIL	BOOL	FALSE	
6	VAR_OUTPUT	TIME_COUNT	INT	0	

② Check, save and then close the Header window.

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



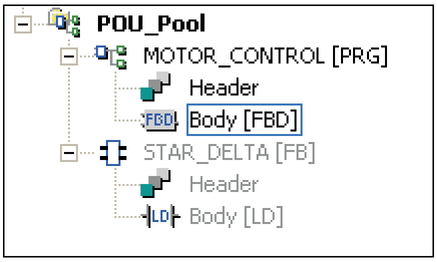
④ Check the Body, there should be no errors and no warnings!



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.



Creating new Global Variables List

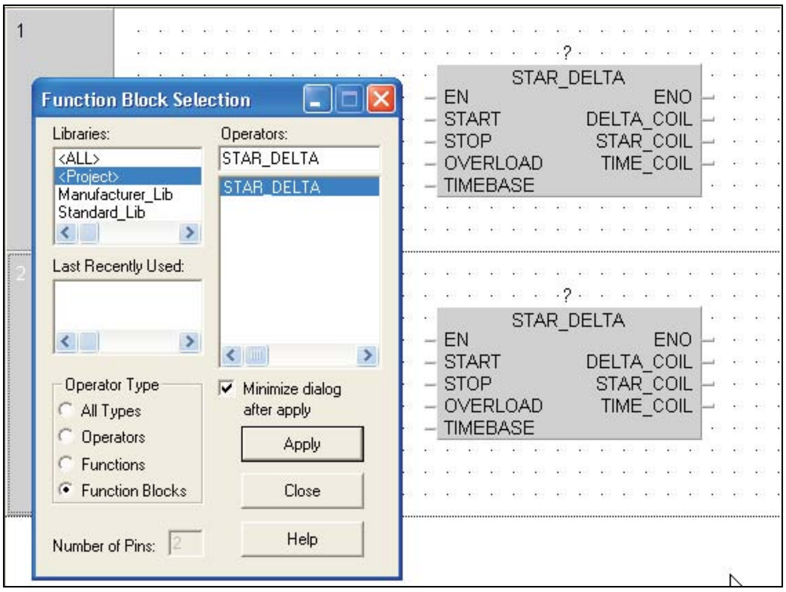
Open the GVL and enter the following I/O details:

Global Variable List

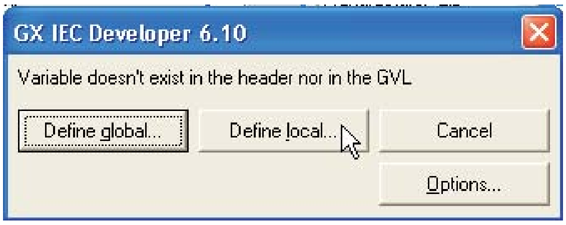
	Class	Identifier	MIT-Addr.	IEC-Addr.	Type	Initial
0	VAR_GLOBAL	START1	X0	%IX0	BOOL	FALSE
1	VAR_GLOBAL	STOP1	X1	%IX1	BOOL	FALSE
2	VAR_GLOBAL	OVERLOAD1	X2	%IX2	BOOL	FALSE
3	VAR_GLOBAL	STAR_COIL1	Y10	%QX16	BOOL	FALSE
4	VAR_GLOBAL	DELTA_COIL1	Y11	%QX17	BOOL	FALSE
5	VAR_GLOBAL	TIME_COIL1	D0	%MW0.0	INT	0
6	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	Y12	%QX18	BOOL	FALSE
10	VAR_GLOBAL	DELTA_COIL2	Y13	%QX19	BOOL	FALSE
11	VAR_GLOBAL	TIME_COIL2	D1	%MW0.1	INT	0

Assigning Instance Names

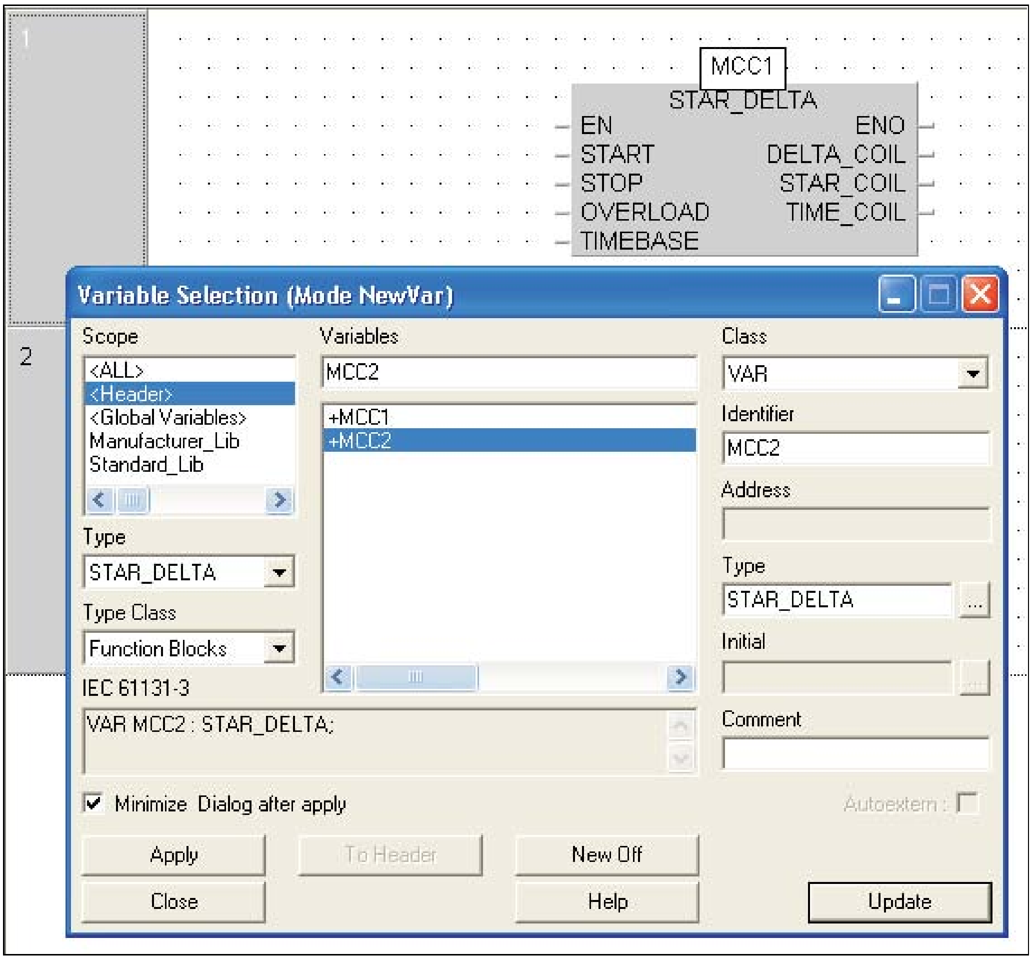
① Open the Body of MOTOR_CONTROL and enter create two networks. Place a Instance of the Function Block STAR_DELTA into each network as shown in the following figure:



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***.



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

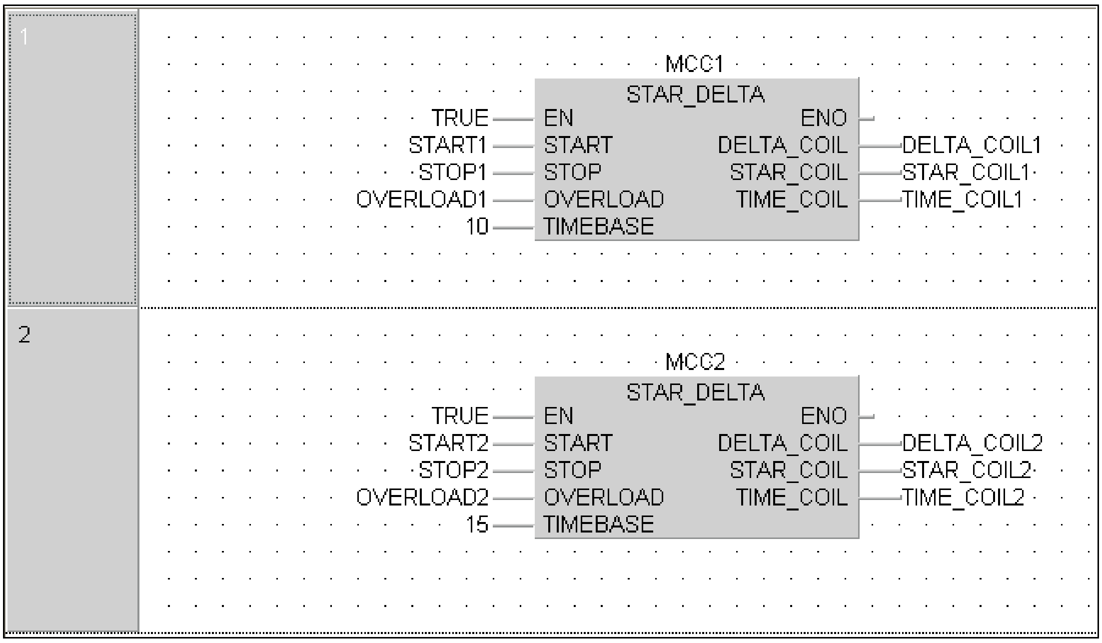


An Instance is the copy of the function block for this POU. For this example simply type MCC1 and MCC2. Notice that once entered, the instances are listed in the variable selection window as +MCC1 and +MCC2 as Type: STAR_DELTA.

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

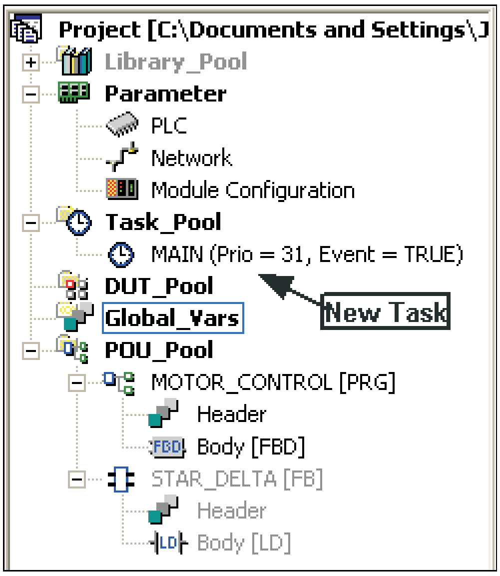
Mitsubishi addresses or symbolic declarations may be used. However, if Mitsubishi 'MELSEC' direct addresses are used then the program will no longer adhere to the IEC conventions.

Designating the variable "TRUE" as above, automatically assigns a 'normally on' contact (Q-Series SM400) 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:

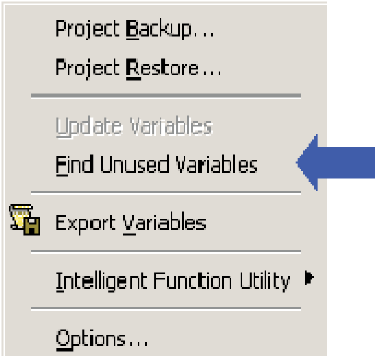


② Double click on the task and bind the POU “MOTOR_CONTROL” to the task “MAIN”:

	POU name		Comment
0	MOTOR_CONTROL	...	

③ Save the Program, close all windows and dialogues.

Find unused Variables



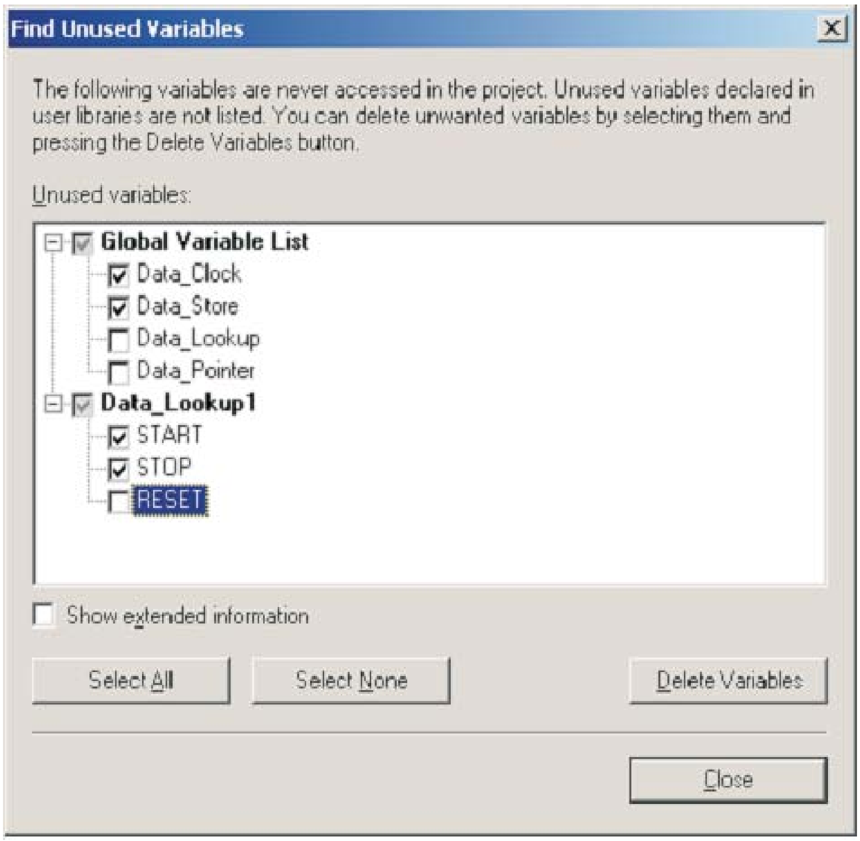
By using the function ***Extras*** → ***Find unused Variables*** you can find and delete all unused global and local variables that are declared but not used in a project.

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

NOTE

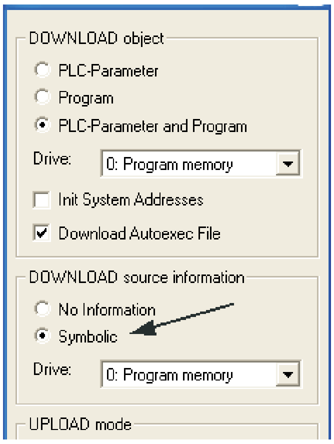
Finding unused variables can only be performed if the project has been build and was not changed since them. Otherwise a warning message will be displayed.

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.

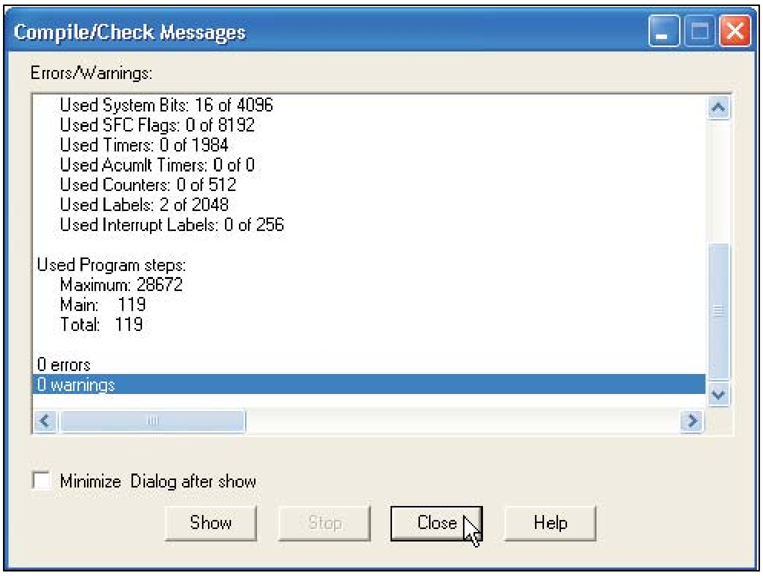


NOTE

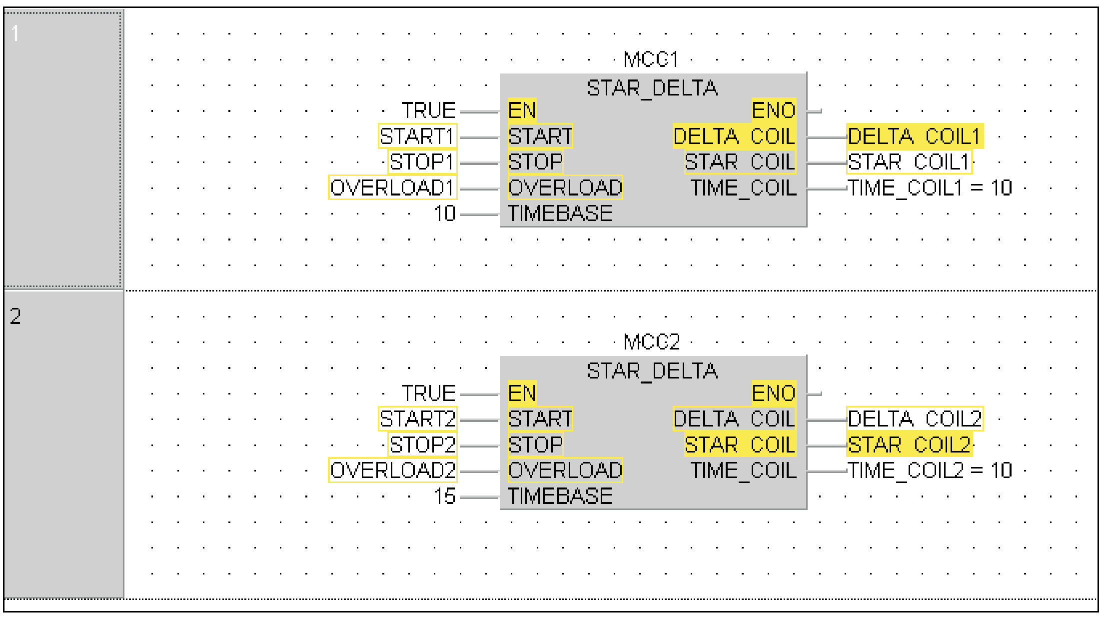
This can produce large reductions in the size of the source code. This is important particularly if the option to send all ***Symbolic*** (Source) Code to the PLC has been selected for download:



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



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

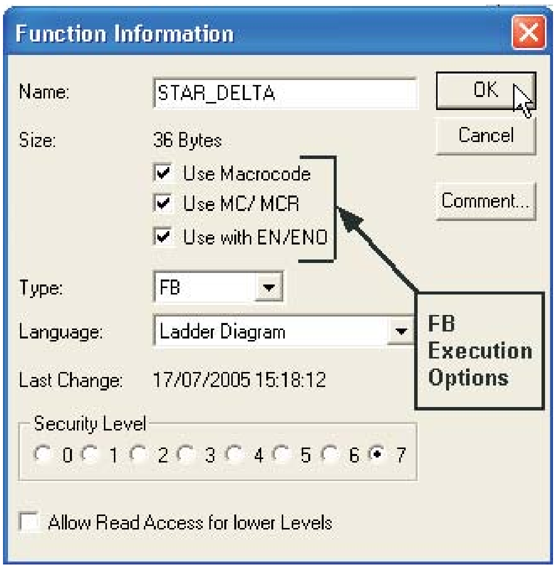


6.3 Execution options of Function Blocks

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:



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.

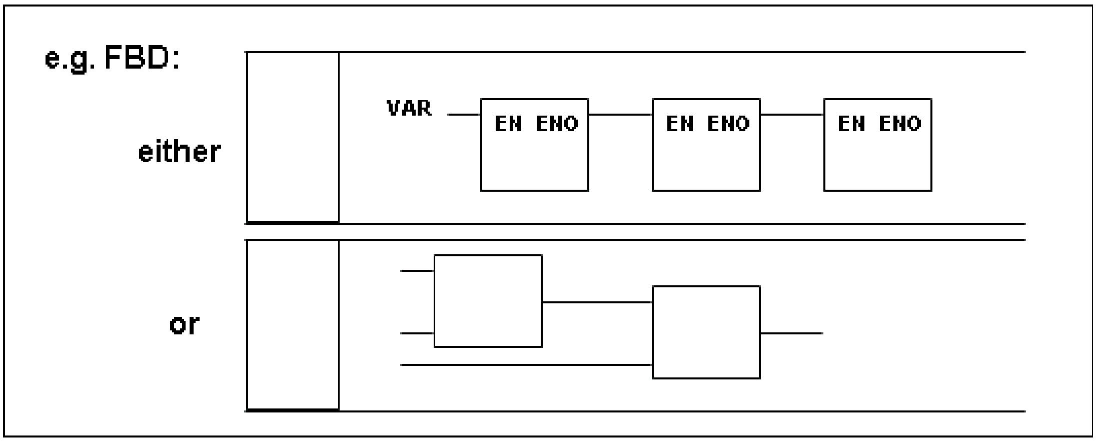
6.3.1 Macrocode execution

- 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:

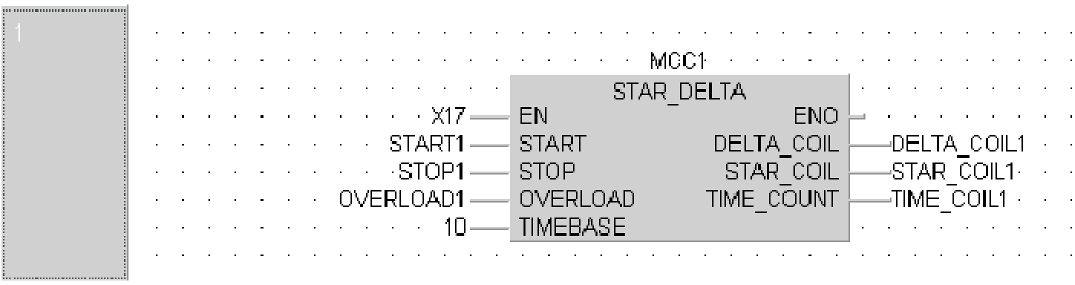


Function Definitions

- 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.
- Care should be taken, especially when using the FBD editor, not to disobey the Mitsubishi programming rules. When building circuits like the previous example, it is tempting to chain lots of instructions together to achieve, i.e. the calculation required. However, if the chosen Mitsubishi instruction, would normally sit at the end position on the rung, why should it suddenly become a series element, simply because you are using FBD?
- Choose the correct instruction for the job i.e. that may well be one from the IEC set.
- Also remember that a 16 bit Mitsubishi multiplication produces a 32 bit answer. If variables are used, then the result "type" should reflect this, i.e. the operands may be of type INT, the result of type DINT.

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 X17 contact:

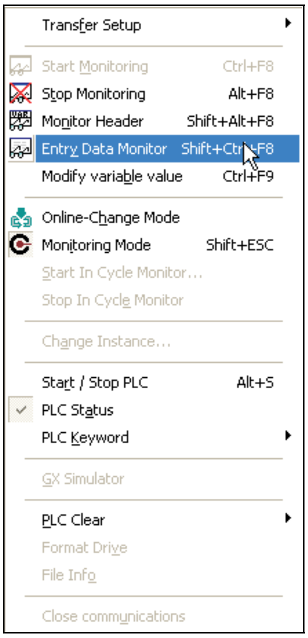


7 Advanced Monitoring Functions

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:

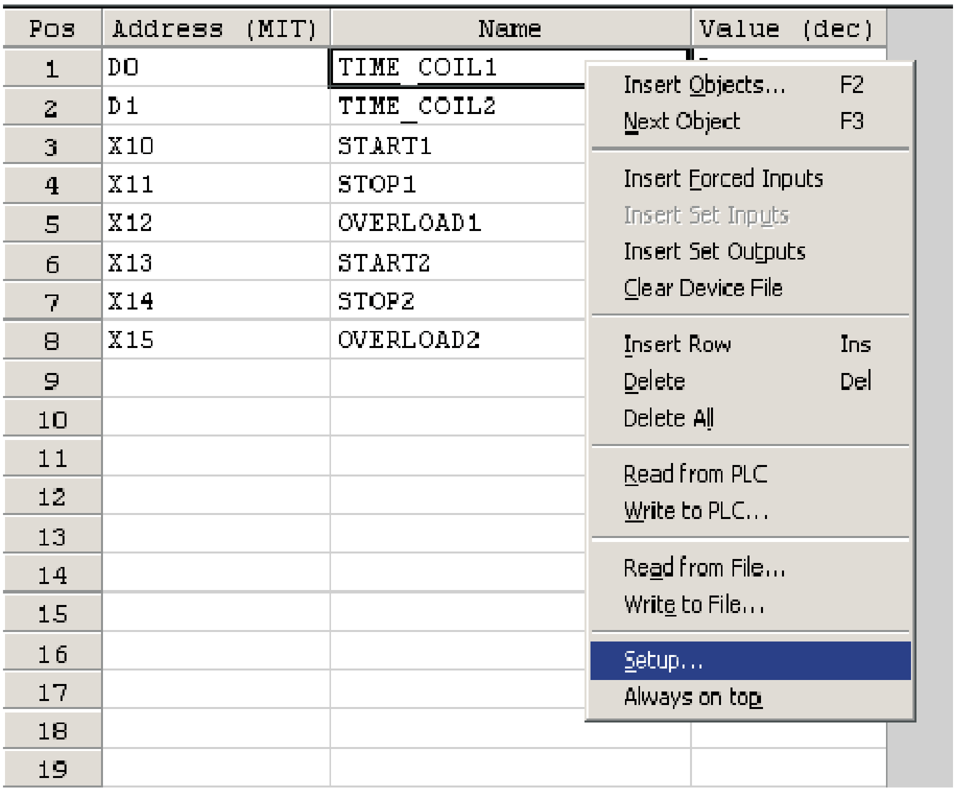
Pos	Address (MIT)	Name	Value (dec)
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			

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 as for Excel:

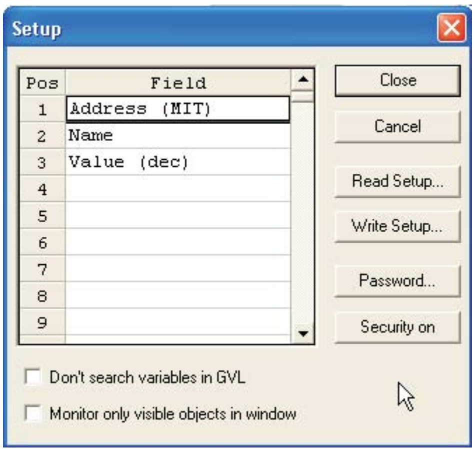
Pos	Address (MIT)	Name	Value (dec)
1	D0	TIME_COIL1	0
2	D1	TIME_COIL2	0
3	X10	START1	0
4	X11	STOP1	0
5	X12	OVERLOAD1	0
6	X13	START2	0
7	X14	STOP2	0
8	X15	OVERLOAD2	0

7.1.1 Customising the EDM

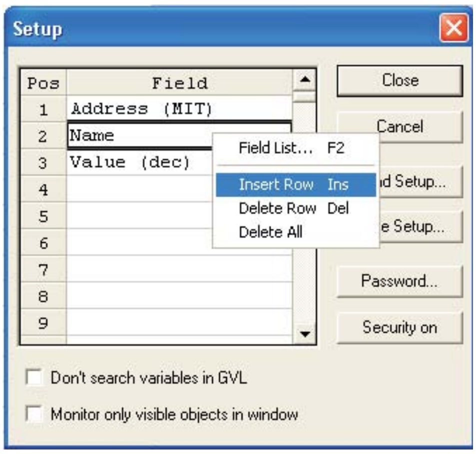
① Right Clicking the mouse button, displays the following window. Select ***Setup***.



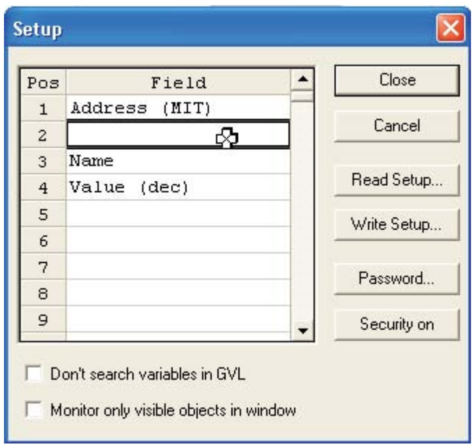
The ***Setup*** window allows the EDM to be user configurable; clicking the right mouse button, displays the configurator window. In this procedure Columns will be added to the EDM table for IEC Address and Hex Value Monitor.



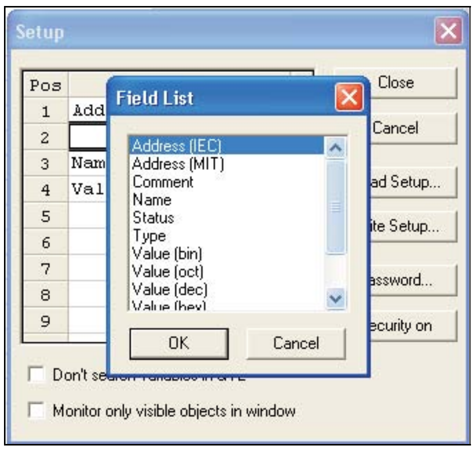
② Highlight or right click on the ***Name*** field and select ***Insert Row*** as shown.



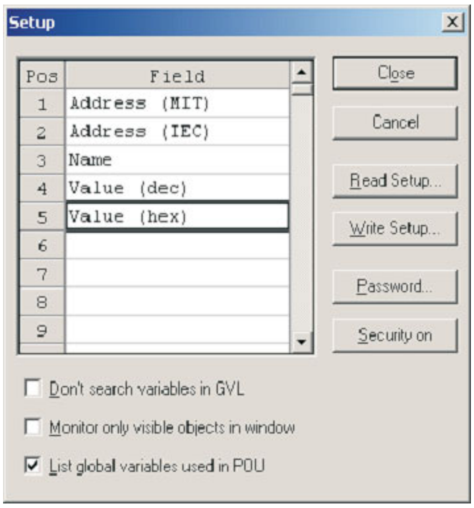
A second window appears, showing options for this row, select ***Value (hex)***, ***Value (bin)***. Repeat for ***Address (IEC)*** and ***Type***.



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.



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	X10	%IX16	START1	0	0
4	X11	%IX17	STOP1	0	0
5	X12	%IX18	OVERLOAD1	0	0
6	X13	%IX19	START2	0	0
7	X14	%IX20	STOP2	0	0
8	X15	%IX21	OVERLOAD2	0	0

In this way, the EDM table can be used to display multiple data on 1 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

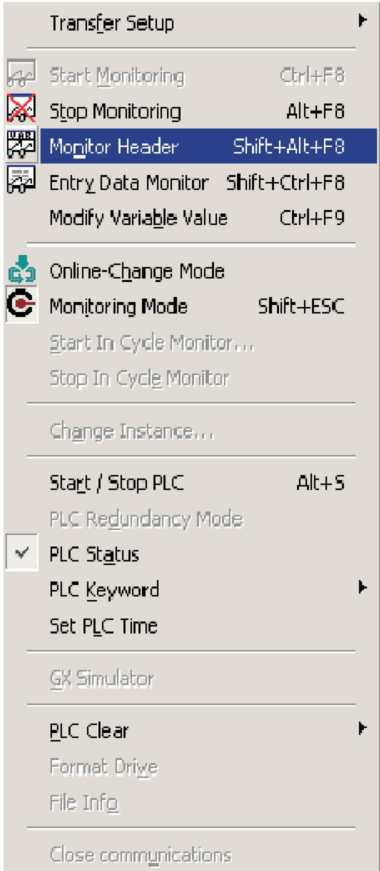
Providing the physical input to the PLC is not active, it is possible to toggle the input image in the CPU on and off by double clicking on the value field for that Boolean addresses as shown:

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	X10	%IX16	START1	1	1
4	X11	%IX17	STOP1	0	0
5	X12	%IX18	OVERLOAD1	0	0
6	X13	%IX19	START2	0	0
7	X14	%IX20	STOP2	0	0
8	X15	%IX21	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:

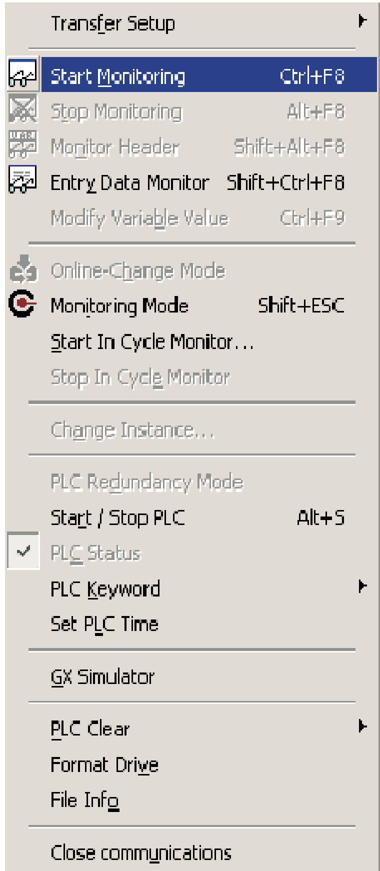
Pos	Address (MIT)	Address (IEC)	Name	Value (dec)	Value (hex)
1			-MOTOR_CONTROL		
2	X10	%IX16	START1	1	1
3	X11	%IX17	STOP1	0	0
4	X12	%IX18	OVERLOAD1	0	0
5	Y21	%QX33	DELTA_COIL1	1	1
6	Y20	%QX32	STAR_COIL1	0	0
7	D0	%MW0.0	TIME_COIL1	10	A
8	X13	%IX19	START2	0	0
9	X14	%IX20	STOP2	0	0
10	X15	%IX21	OVERLOAD2	1	1
11	Y23	%QX35	DELTA_COIL2	0	0
12	D1	%MW0.1	TIME_COIL2	0	0
13	Y22	%QX34	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:

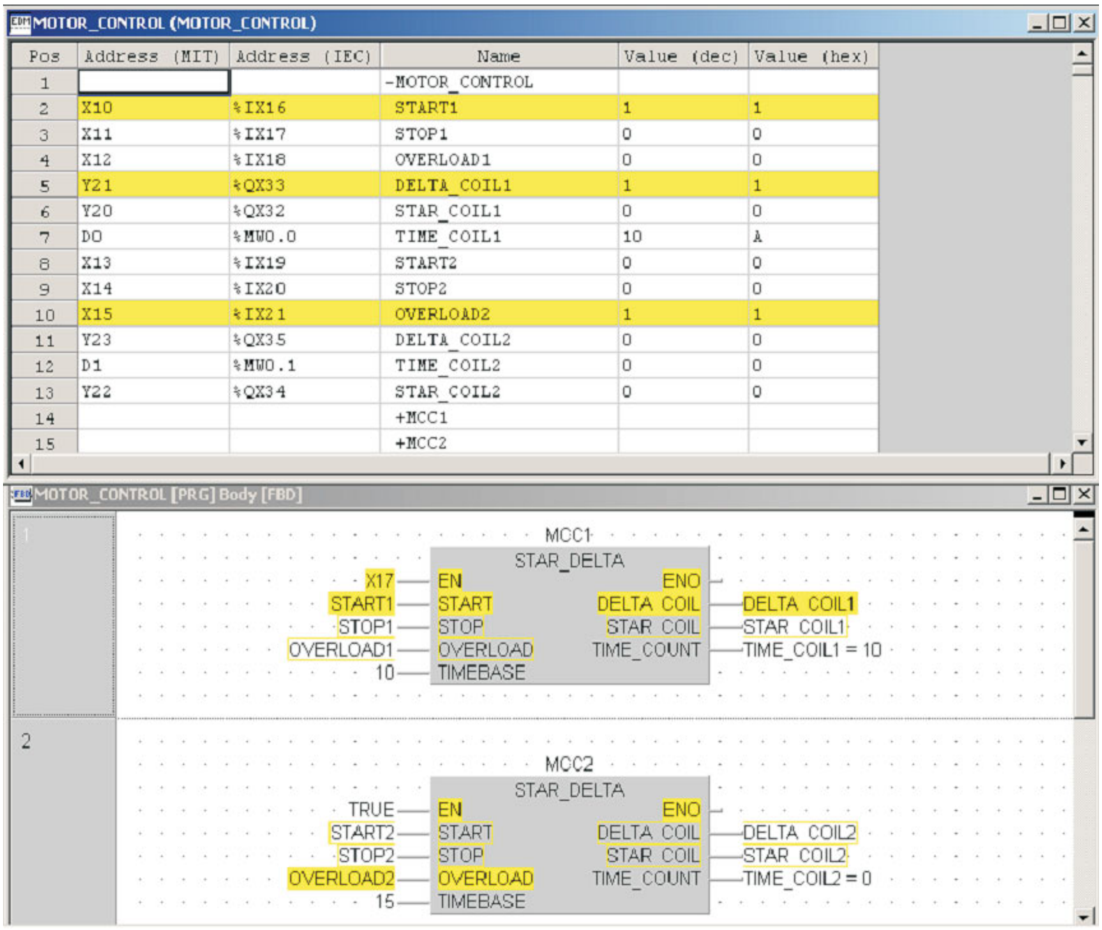


NOTE

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 A & FX PLC's or across slower serial links.

Simultaneous Monitoring of Header and Body

Here is an example of Monitoring a POU and its header simultaneously:



Pos	Address (MIT)	Address (IEC)	Name	Value (dec)	Value (hex)
1			-MOTOR_CONTROL		
2	X10	%IX16	START1	1	1
3	X11	%IX17	STOP1	0	0
4	X12	%IX18	OVERLOAD1	0	0
5	Y21	%QX33	DELTA_COIL1	1	1
6	Y20	%QX32	STAR_COIL1	0	0
7	D0	%MW0.0	TIME_COIL1	10	A
8	X13	%IX19	START2	0	0
9	X14	%IX20	STOP2	0	0
10	X15	%IX21	OVERLOAD2	1	1
11	Y23	%QX35	DELTA_COIL2	0	0
12	D1	%MW0.1	TIME_COIL2	0	0
13	Y22	%QX34	STAR_COIL2	0	0
14			+MCC1		
15			+MCC2		

7.4 Monitoring Mitsubishi “Transfer Form” Objects

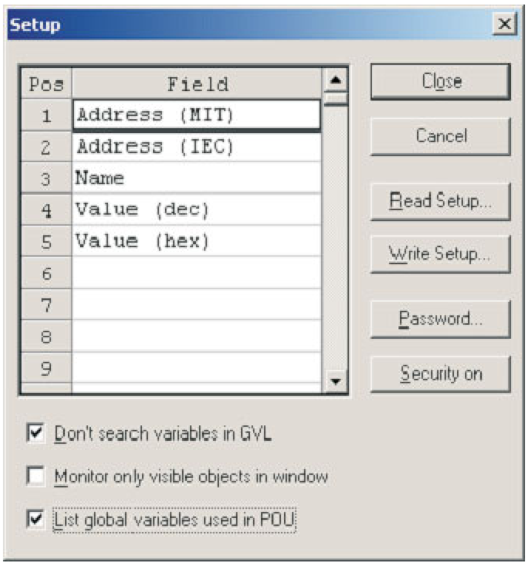
It is also possible to monitor using the Mitsubishi Kn (Official – ‘Transfer Form’) notation for Boolean objects. For example K1X0 monitors X0 - X3 as shown in the following example:

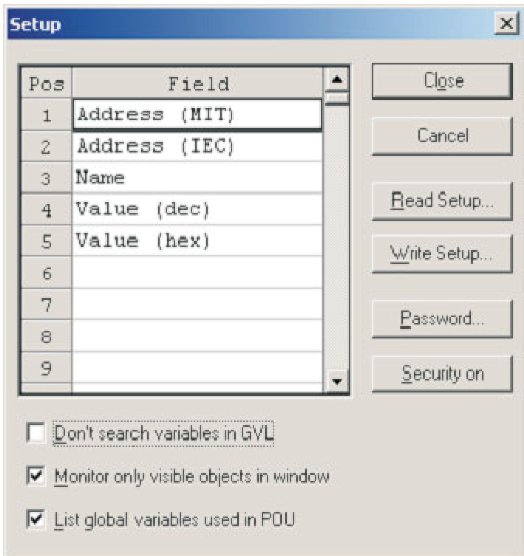
Pos	Address (MIT)	Address (IEC)	Name	Value (dec)	Value (hex)
1	D0	%MW0.0	TIME_COIL1	10	A
2	D1	%MW0.1	TIME_COIL2	0	0
3	X10	%IX16	START1	1	1
4	X11	%IX17	STOP1	0	0
5	X12	%IX18	OVERLOAD1	0	0
6	X13	%IX19	START2	0	0
7	X14	%IX20	STOP2	0	0
8	X15	%IX21	OVERLOAD2	1	1
9					
10	K1X10	%IW19.1.16	K1X10	1	1
11					
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.

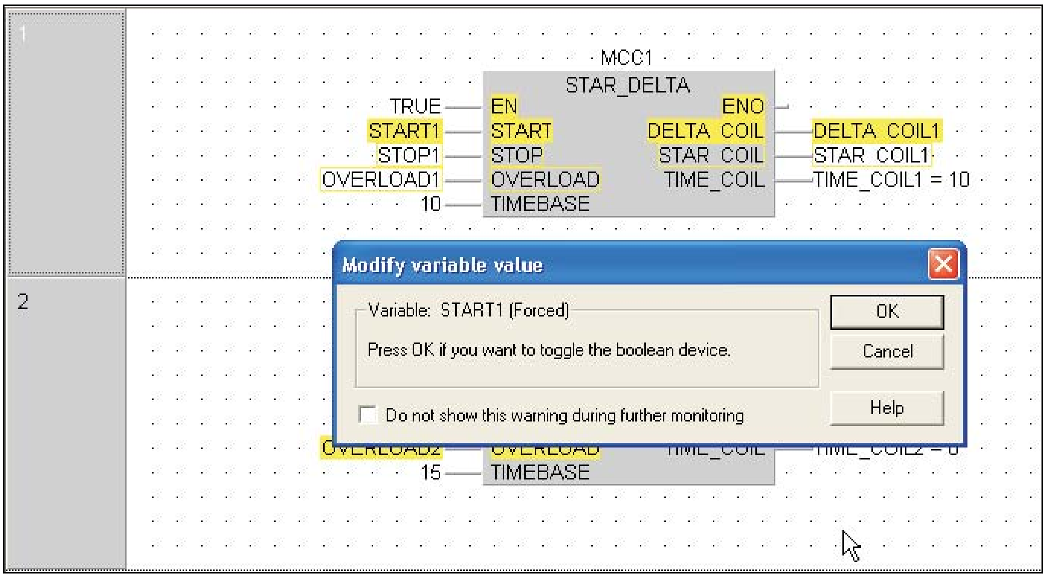




7.5 Modifying Variable Values from the POU Body

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.



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.

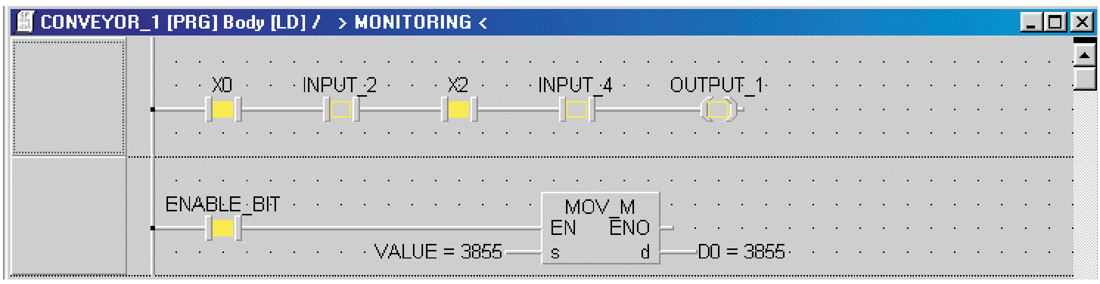
Again, if there is PLC code writing to this variable, then this will overwrite this action.

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.

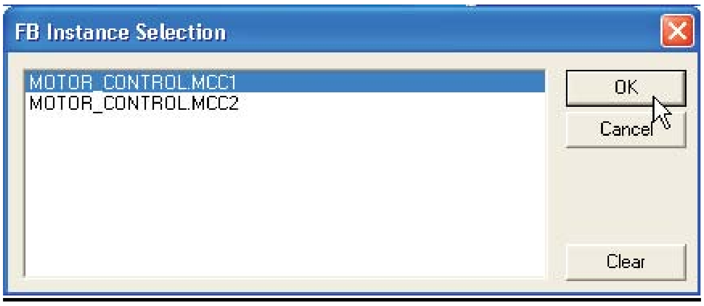
If Monitor Mode is stopped, then started again, identifiers are displayed.



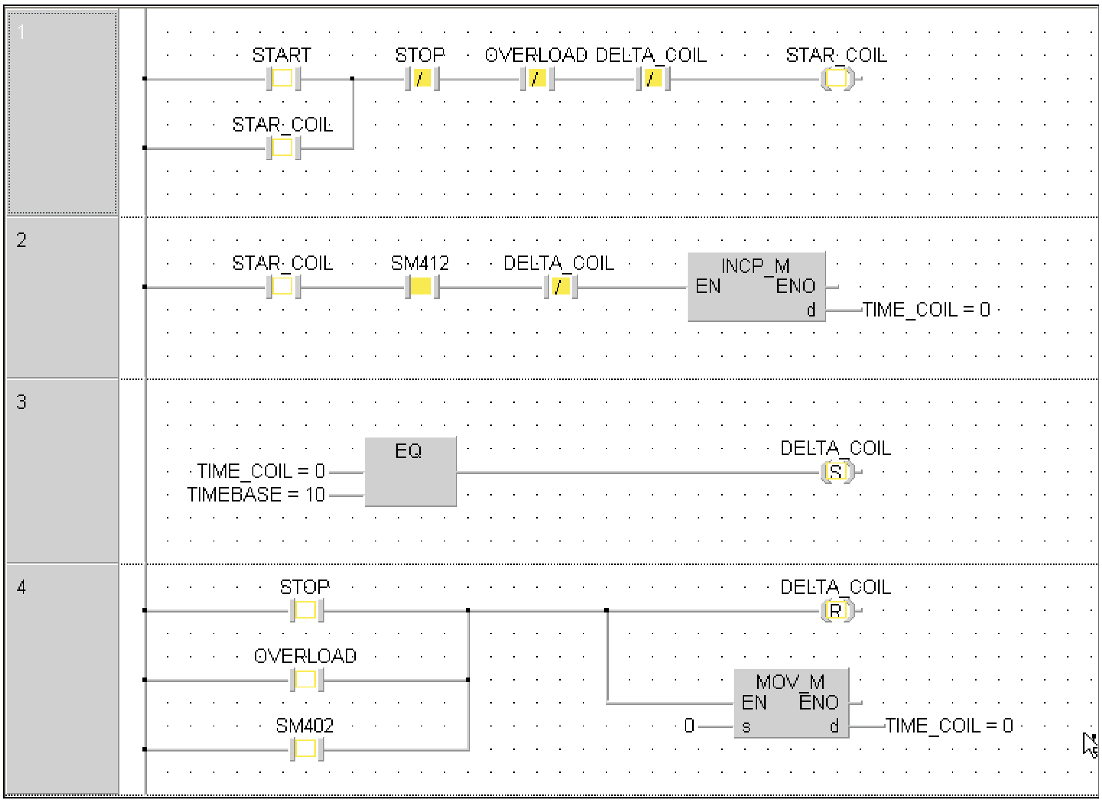
7.6 Monitoring “Instances” of Function Blocks

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:



② Select the instance of the Function Block MOTOR_CONTROL.MCC1 and observe the monitored page:



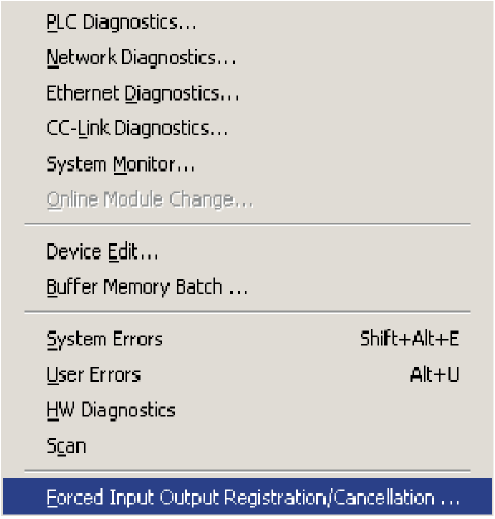
In this manner every instance of any Function Block may be monitored autonomously.

8 Forcing Inputs and Outputs

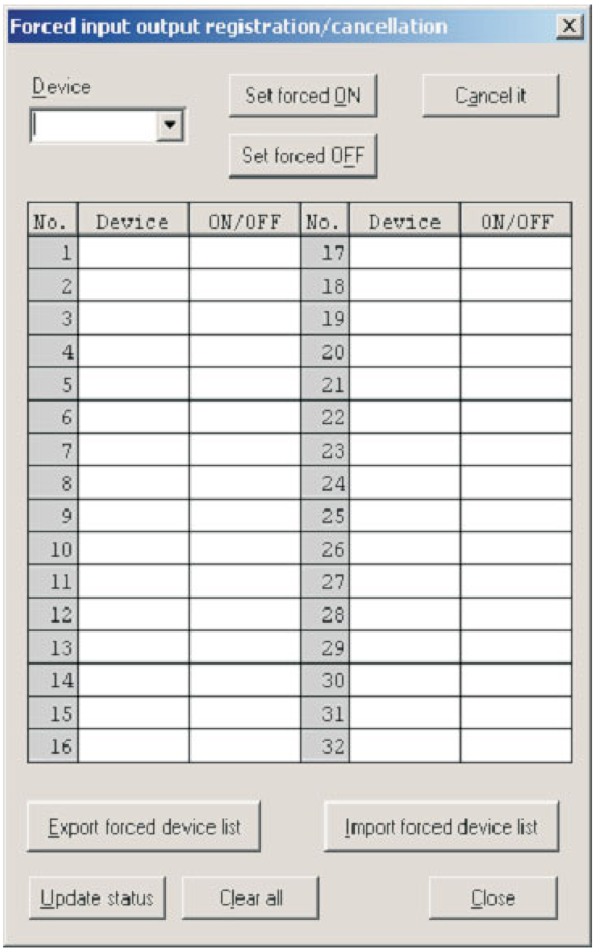
This GX-IEC Developer feature enables both the Physical Hardware Input and Output registers to be forced independently from the program scan.

Although great care must be exercised when operating this feature in live situations, it is particularly useful, as it enables the states of all physical Input and Output devices to be overridden.

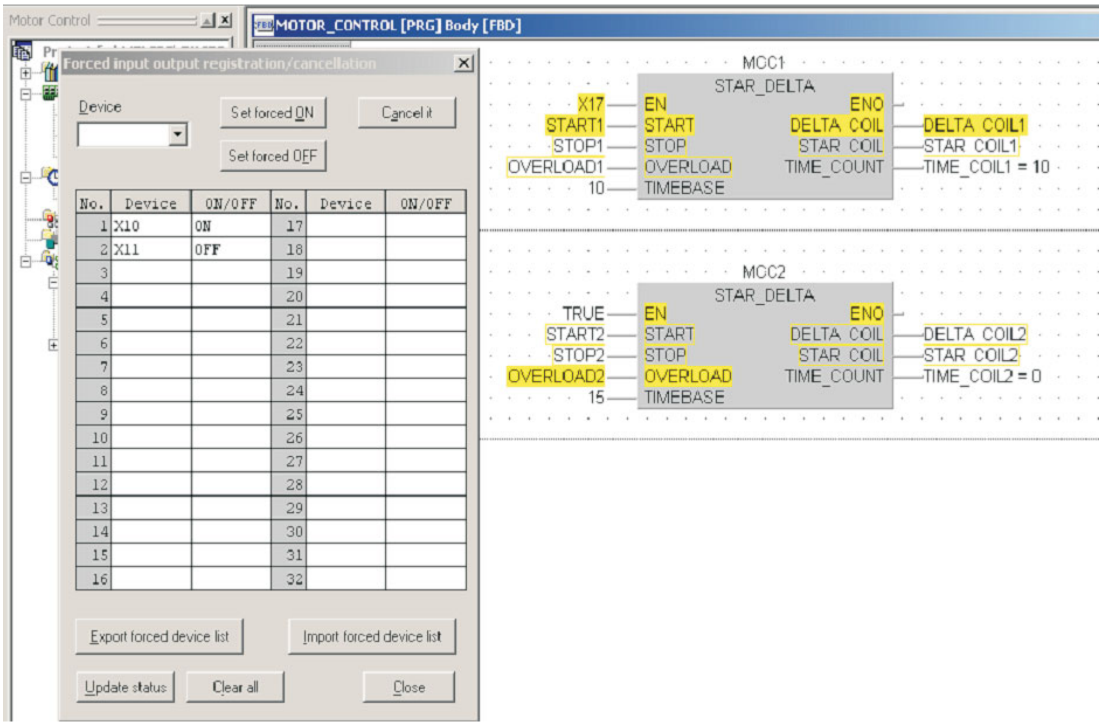
① To activate this function, and select the ***Forced input output registration/cancellation*** select it from the ***Debug*** menu thus:



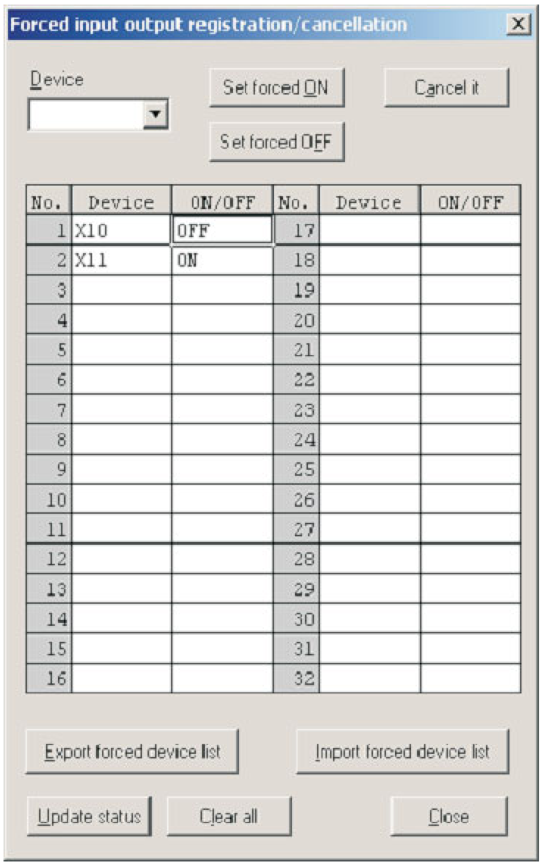
- The following window will be displayed:



② Enter X10 and X11 into the ***Device*** dialogue box and click on the ***Set Forced ON*** button for both variables:

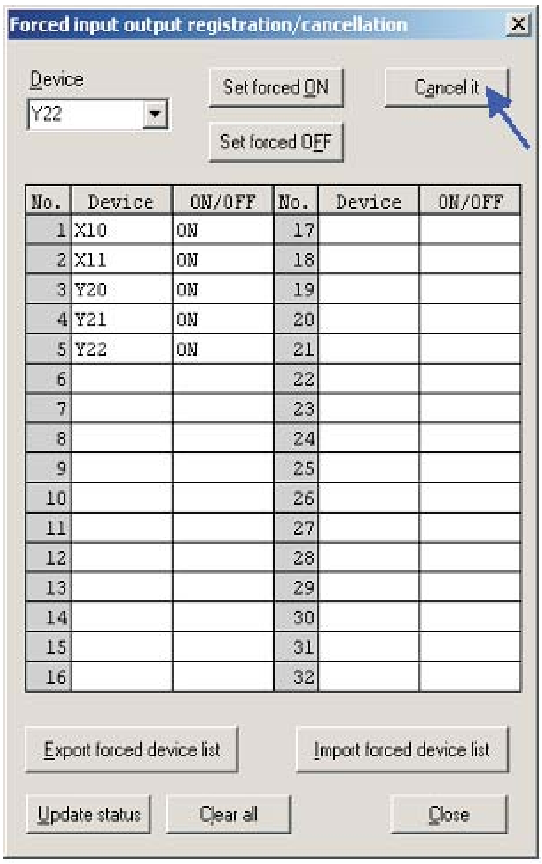


③ To toggle the status of X10 or X11, double click the left mouse button over the ***ON/OFF*** status cell.

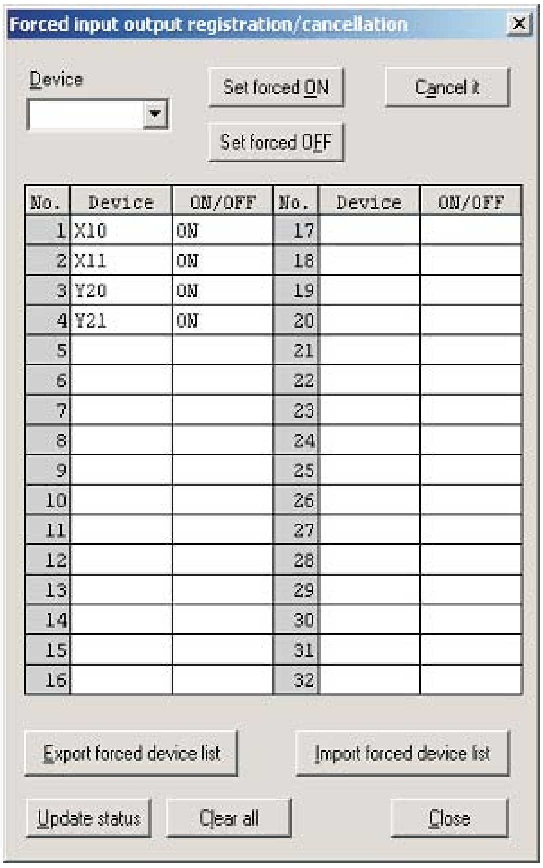


④ Carry out this method of forcing on Y20, Y21 and Y22, noting the effect on the devices.

⑤ To clear a force on an individual device, enter the device then click on the ***Cancel it*** button thus.

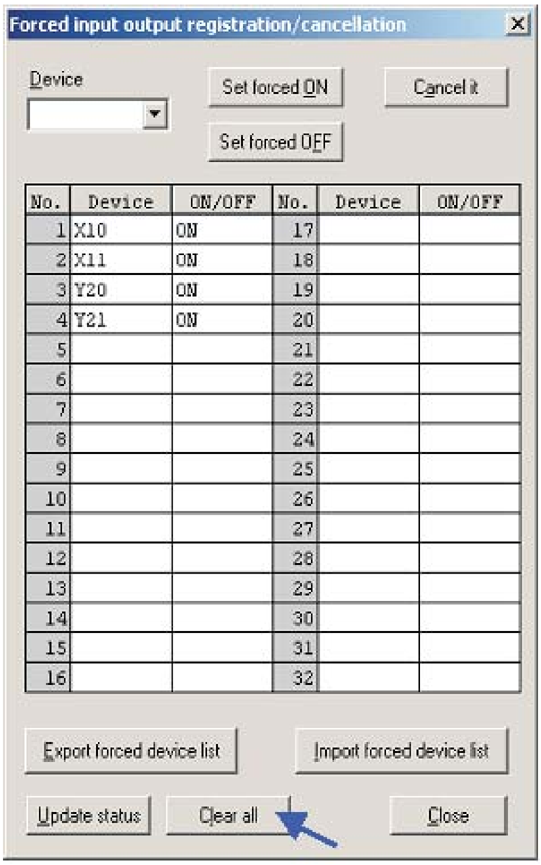


⑥ The following display will result:

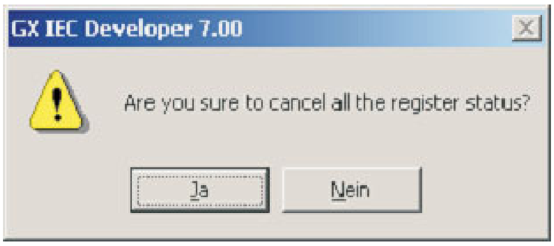


NOTE When any forces are registered within the PLC, the 'Mode' light on the CPU flashes at 2Hz.

⑧ To clear all forces registered in the CPU, click the ***Clear All*** button



⑨ Confirm the cancellation request using the following response:

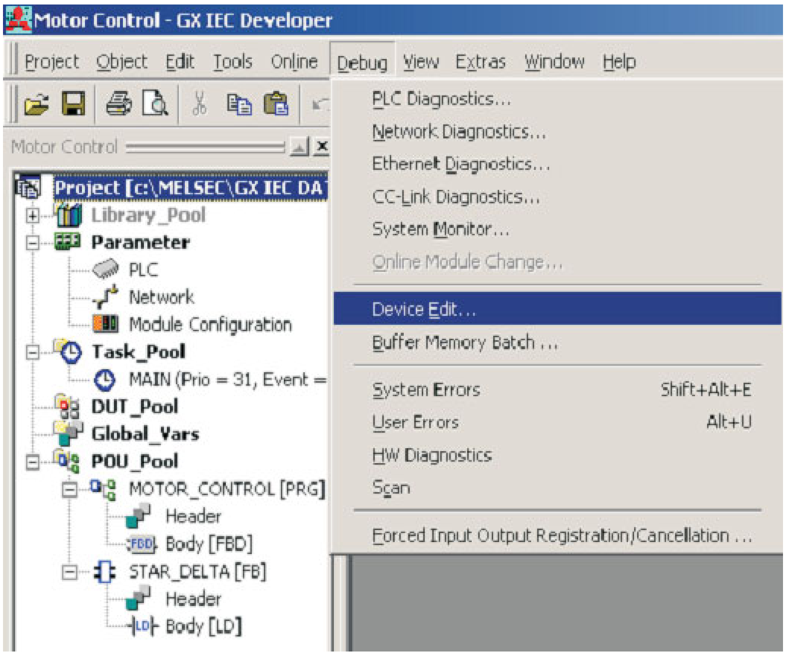


NOTE Individual forces may be removed from the active force table by clicking the ***Cancel it*** button for the appropriate entry.

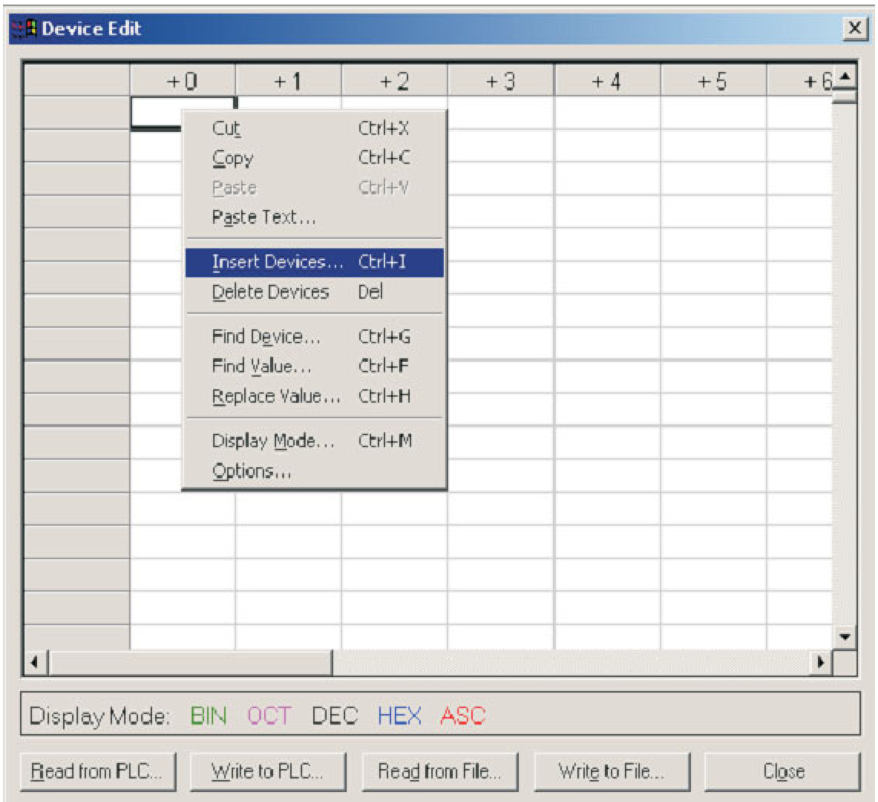
9 Device Edit

The ***Device Edit*** function is akin to the ***D,W,R set*** in MELSEC MEDOC and ***Device Memory*** feature in GX-Developer.

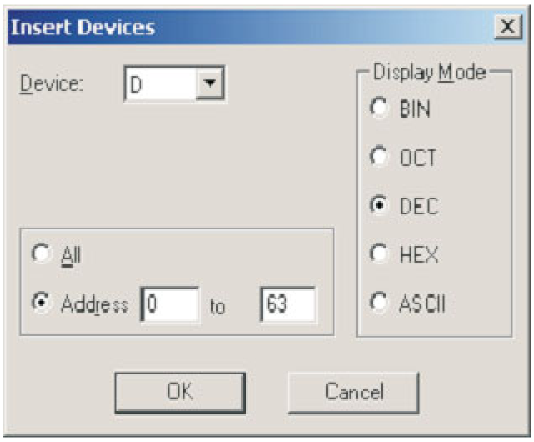
① Select ***Device Edit*** from the ***Debug*** menu.



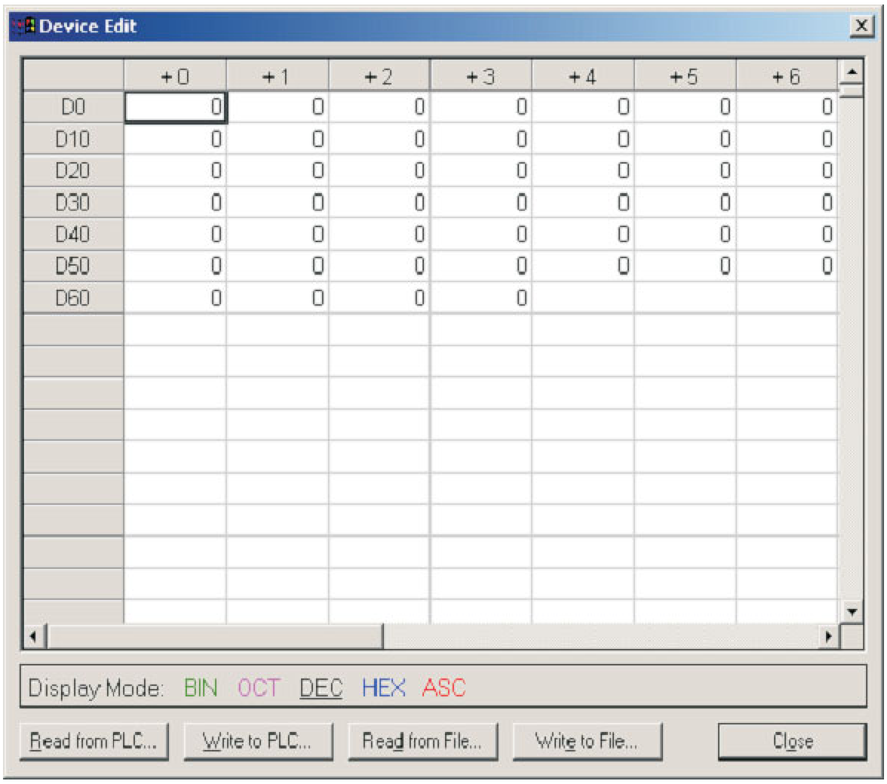
② Highlight the cell in the top left hand corner. Click the right mouse button and then select ***Insert Devices***:



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***.

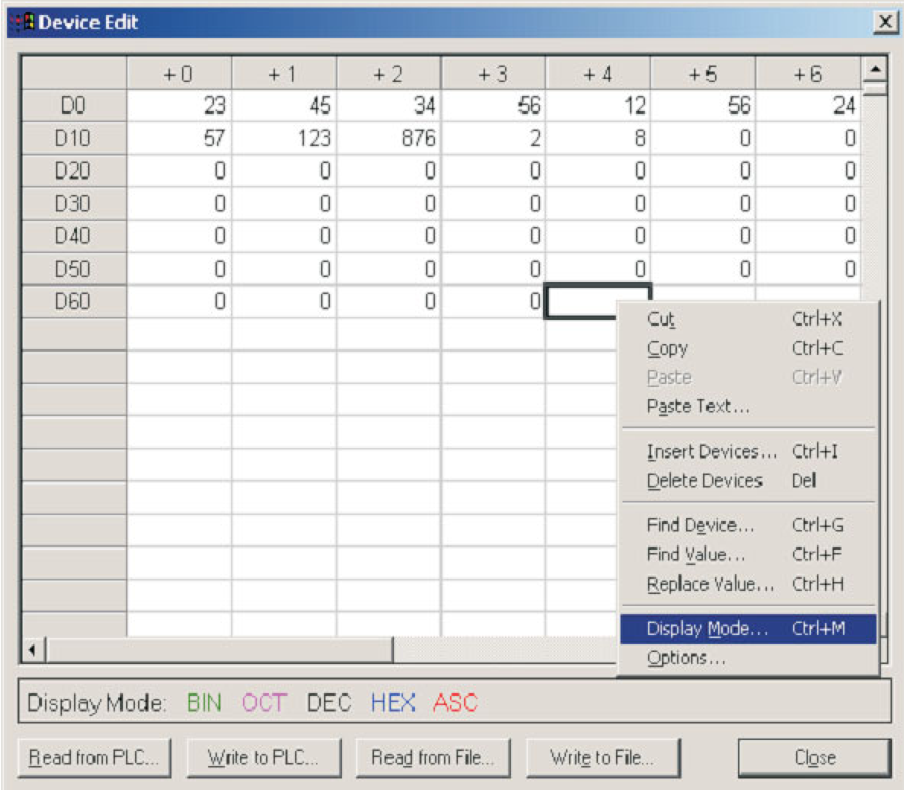


The device table can be configured as you wish and can be stored, as a file or written to the PLC. Information can also be uploaded from the PLC and displayed as below.

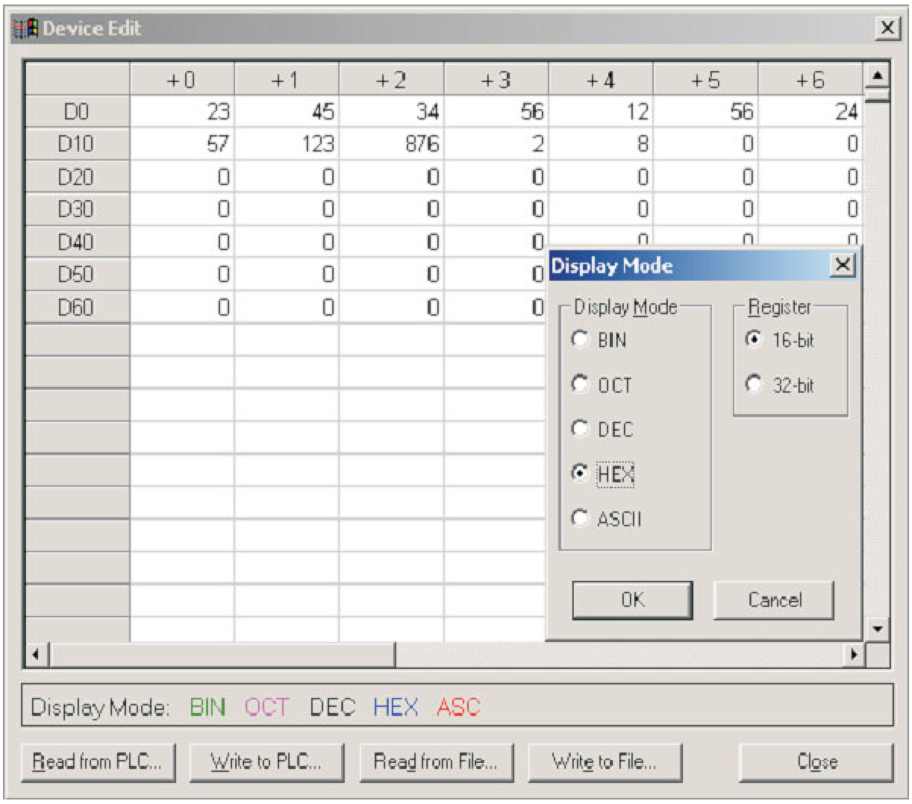


The right mouse button supports many editing functions, find and replace, copy / paste, etc.

④ Highlight a row by clicking on the left hand box, i.e. “D0” Select ***Display Mode***:



This window allows the display format to be changed - try ***HEX***.



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.

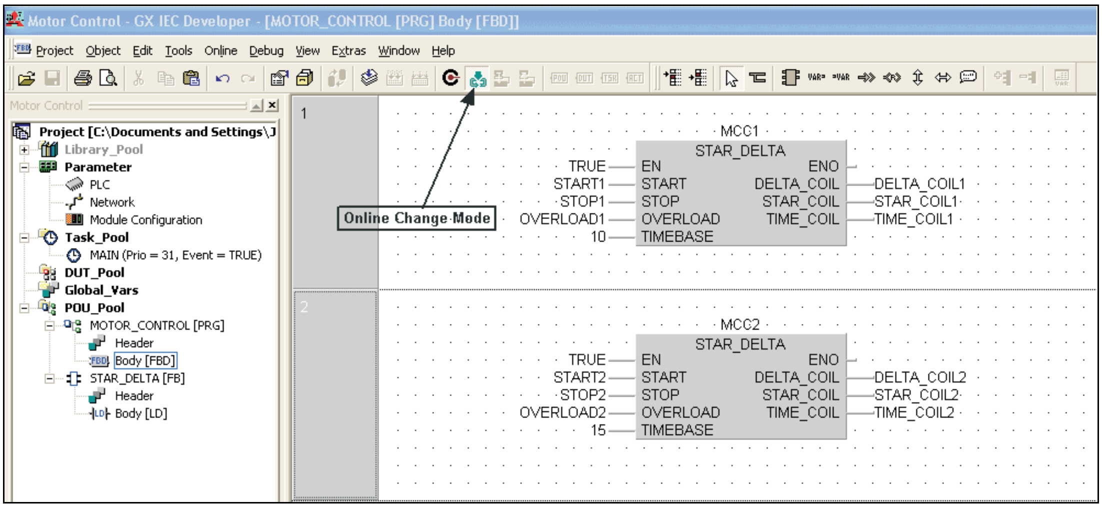
10 Online Mode

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.

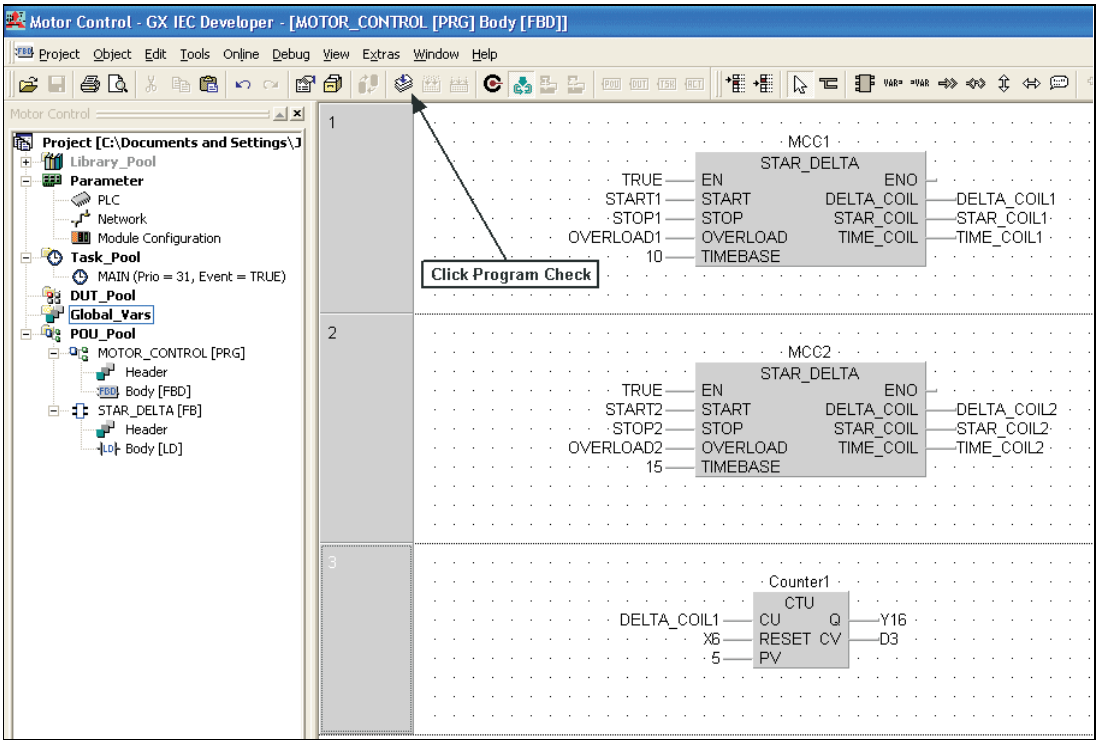
Rebuild the project and download it to the PLC.

10.1 Online Change Mode

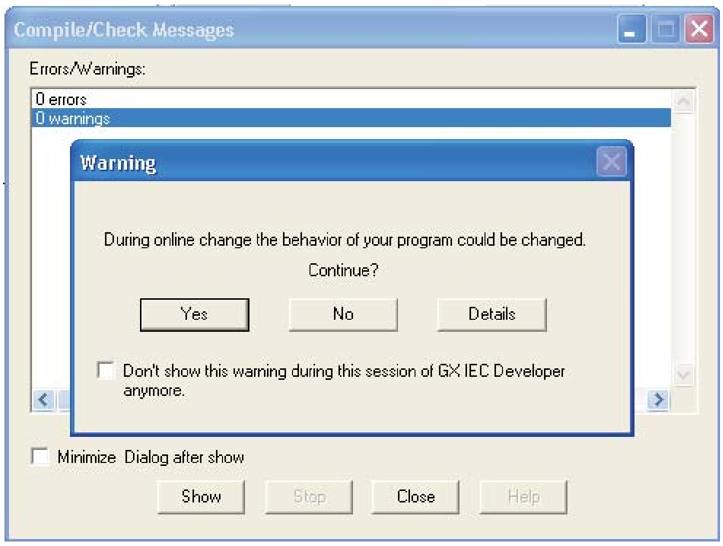
① Open the body of the ‘MOTOR_CONTROL’ POU and select ***Online change mode***:



② Add an additional network as shown below:

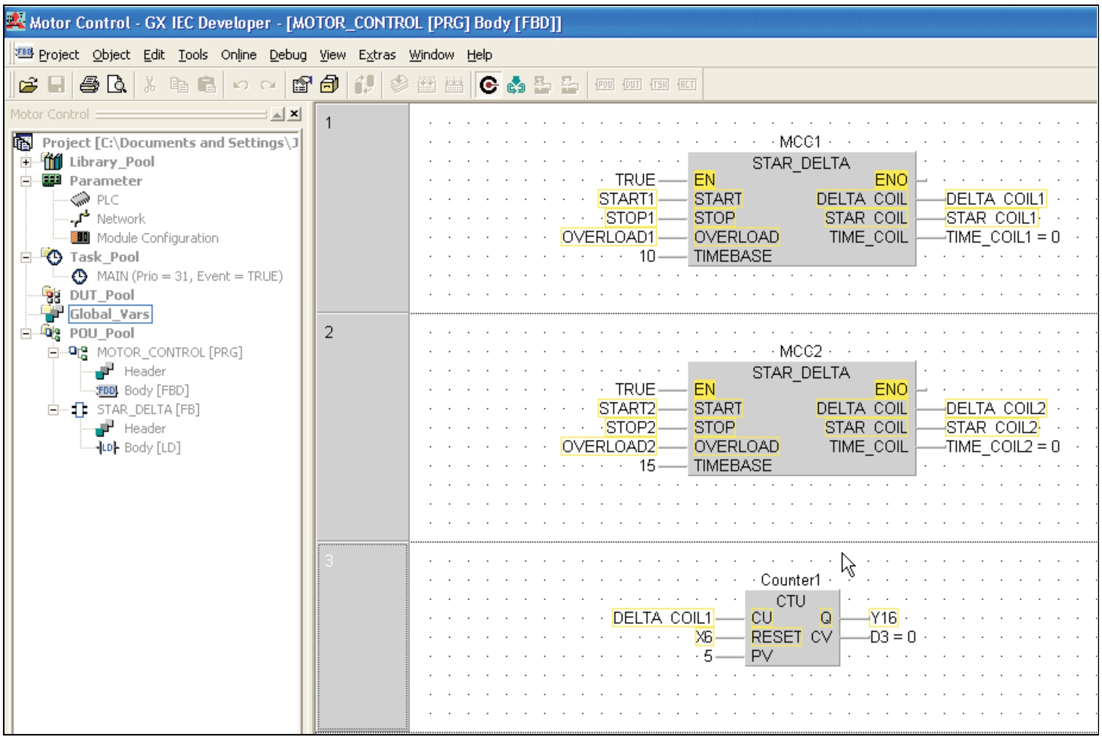


③ Then with the mouse, click away from this network or click on the check button and the changes are compiled and sent to the PLC automatically following a prompt to carry out or abort the action:



NOTE Online editing is only allowed if the code is identical in the resident project and PLC.

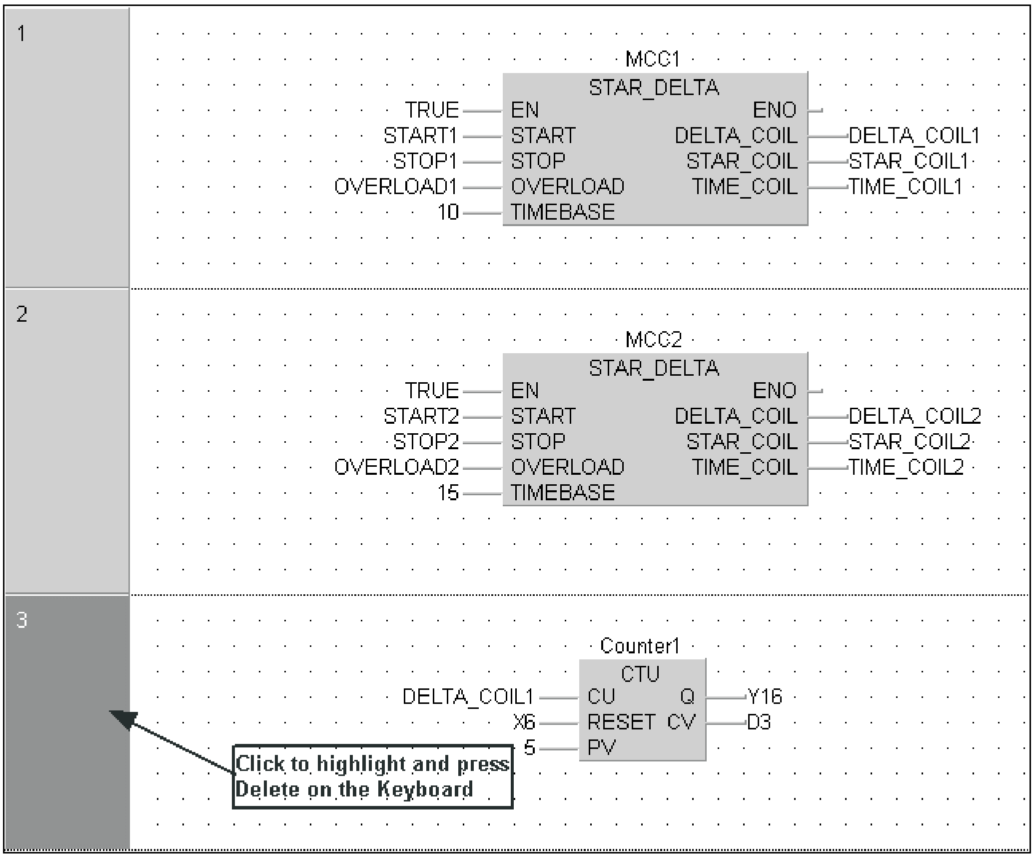
④ Enter Monitor mode and observe the operation of the modified block:



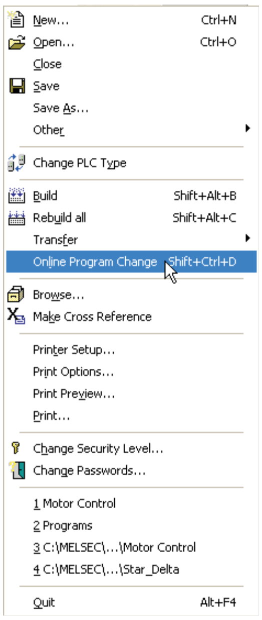
10.2 Online Program Change

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-Developer programs must be identical before proceeding).

① Highlight network 3 on the POU body “MOTOR_CONTROL” and press delete on the keyboard.

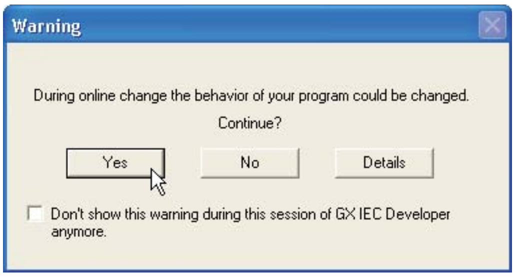


② Invoke the ***Online Program Change*** feature from the ***Project*** Menu. GX-IEC Developer will compile and write the online change automatically.

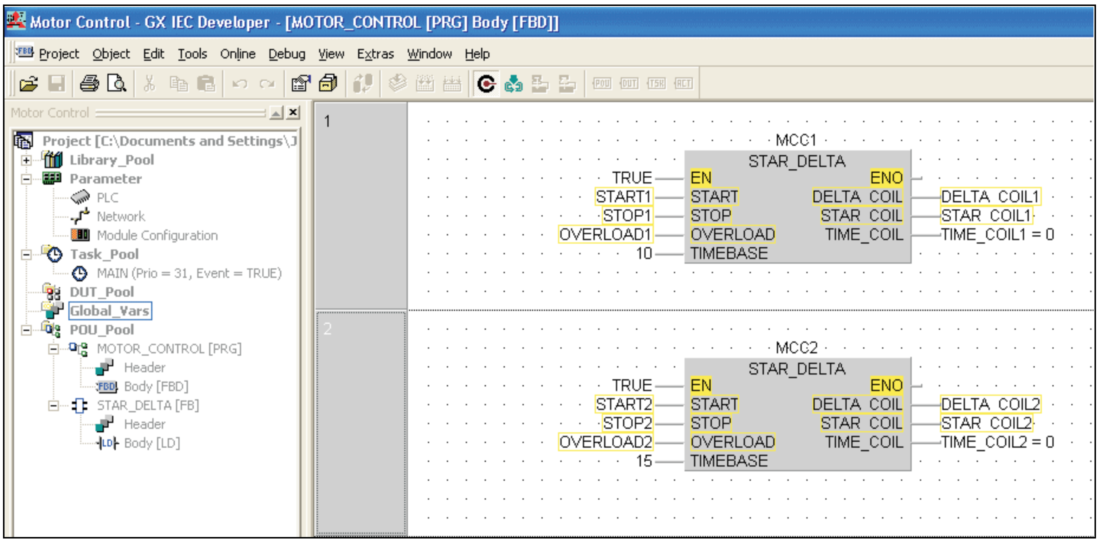


The system will prompt to continue or abort the process at this point.

③ Click ***Yes*** and wait for the download synchronisation process to complete:



④ Confirm correct operation by entering ***Monitor mode*** in the active POU.



11 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.

11.1 Example use of a DUT

The following example illustrates the use of a DUT.

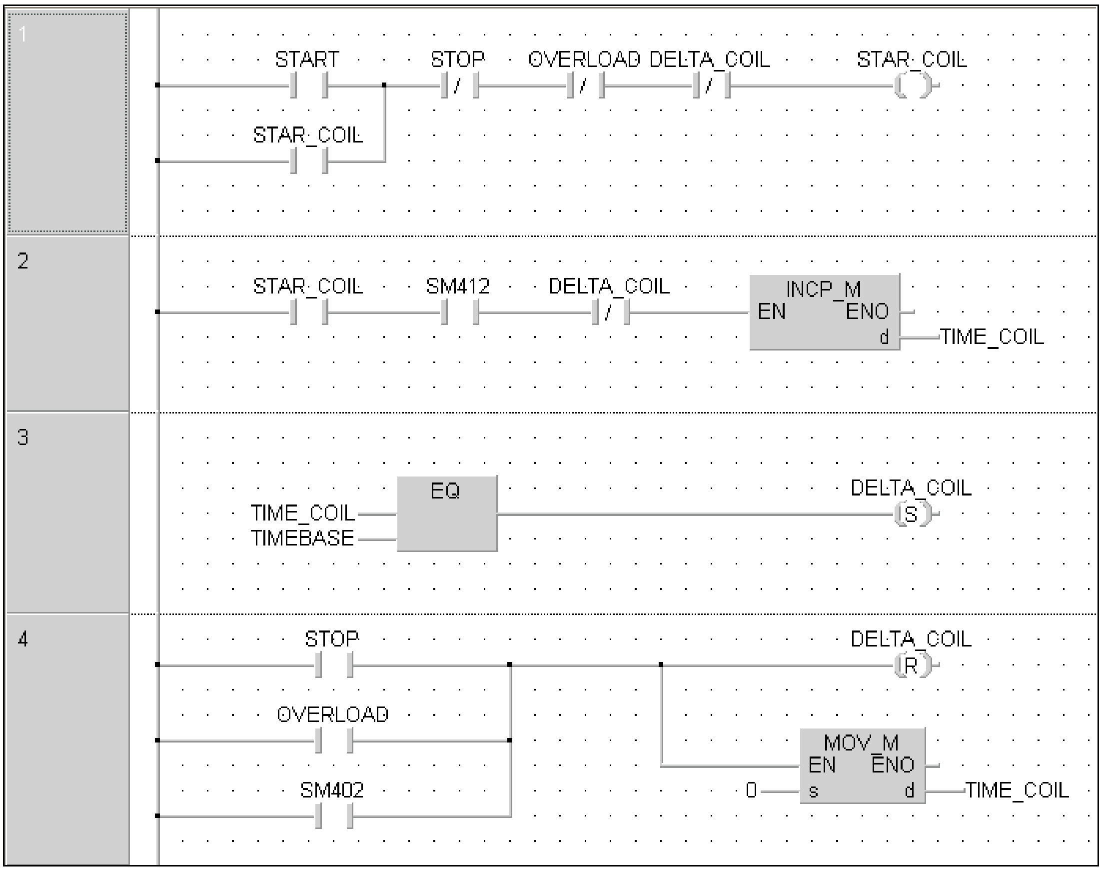
① Create a new project called “Motor Control DUT”:

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:

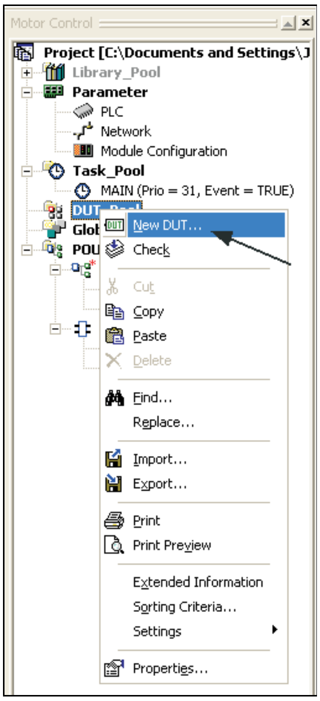
Body: STAR_DELTA



Header: STAR_DELTA

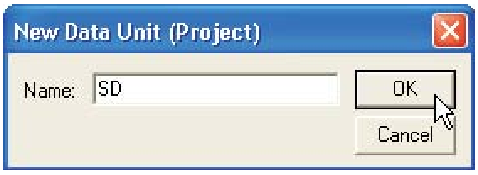
	Class	Identifier	Type	Initial	Comment
0	VAR_INPUT	START	BOOL	FALSE	
1	VAR_INPUT	STOP	BOOL	FALSE	
2	VAR_INPUT	OVERLOAD	BOOL	FALSE	
3	VAR_INPUT	TIMEBASE	INT	0	
4	VAR_OUTPUT	DELTA_COIL	BOOL	FALSE	
5	VAR_OUTPUT	STAR_COIL	BOOL	FALSE	
6	VAR_OUTPUT	TIME_COIL	INT	0	

The Header contains the definitions (Mask) of the data types that will be used when creating the DUT “SD”.

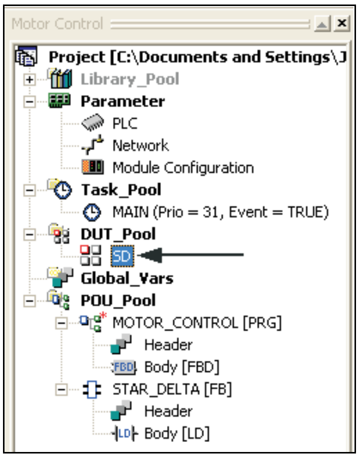


⑤ Create a new DUT by right clicking on the ***DUT Pool*** icon in the Program navigation window

or from the DUT icon DUT on the toolbar.



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			...		

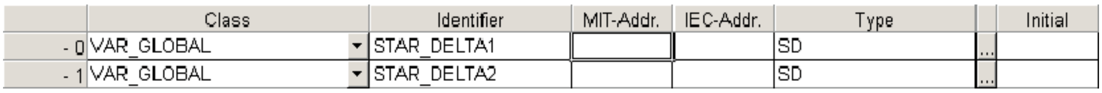
⑧ Enter the following data into the DUT "SD".

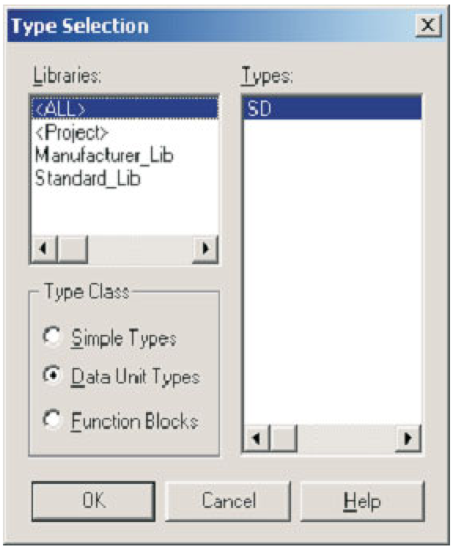
	Identifier	Type		Initial	Comment
0	DELTA	BOOL	...	FALSE	
1	O_L	BOOL	...	FALSE	
2	STAR	BOOL	...	FALSE	
3	START	BOOL	...	FALSE	
4	STOP	BOOL	...	FALSE	
5	TB	INT	...	0	
6	TV	INT	...	0	

⑨ Close the DUT and save the program.

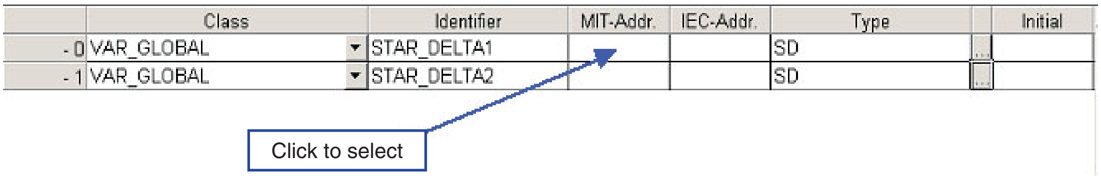
⑩ Open the GVL and create 2 new entries STAR_DELTA1 and STAR_DELTA2.

⑪ Click the 'ellipsis' ... to specify the ***Type*** as "Data Unit Types" SD for both entries:

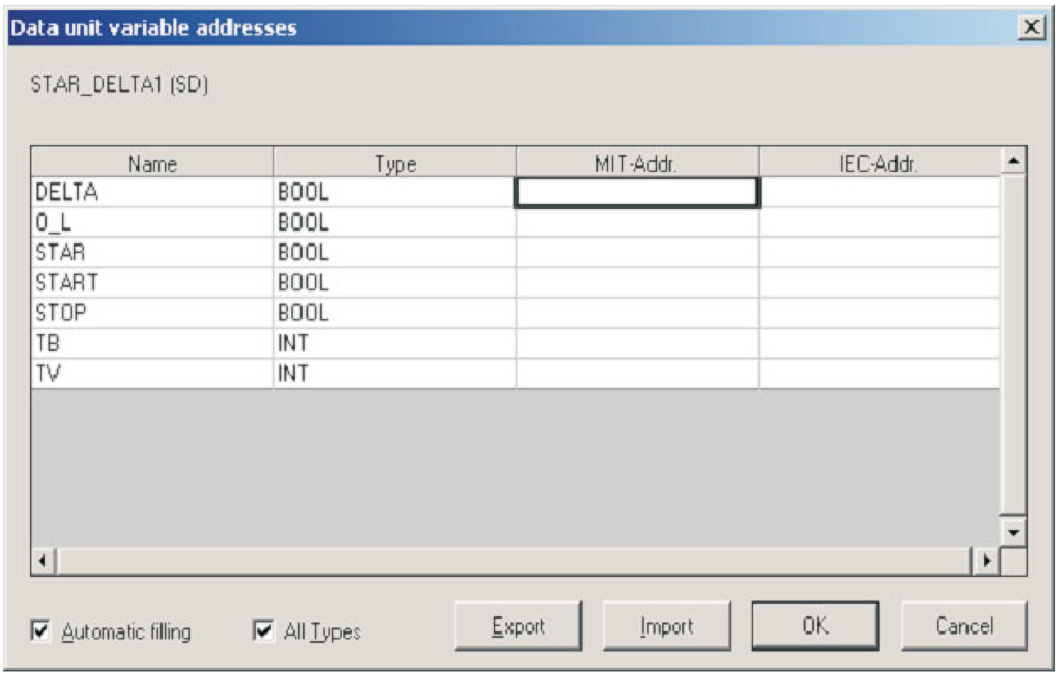




⑫ Next, click on the ***MIT-Addr.*** cell for STAR_DELTA1 to enter the variable data for the selected DUT entry:



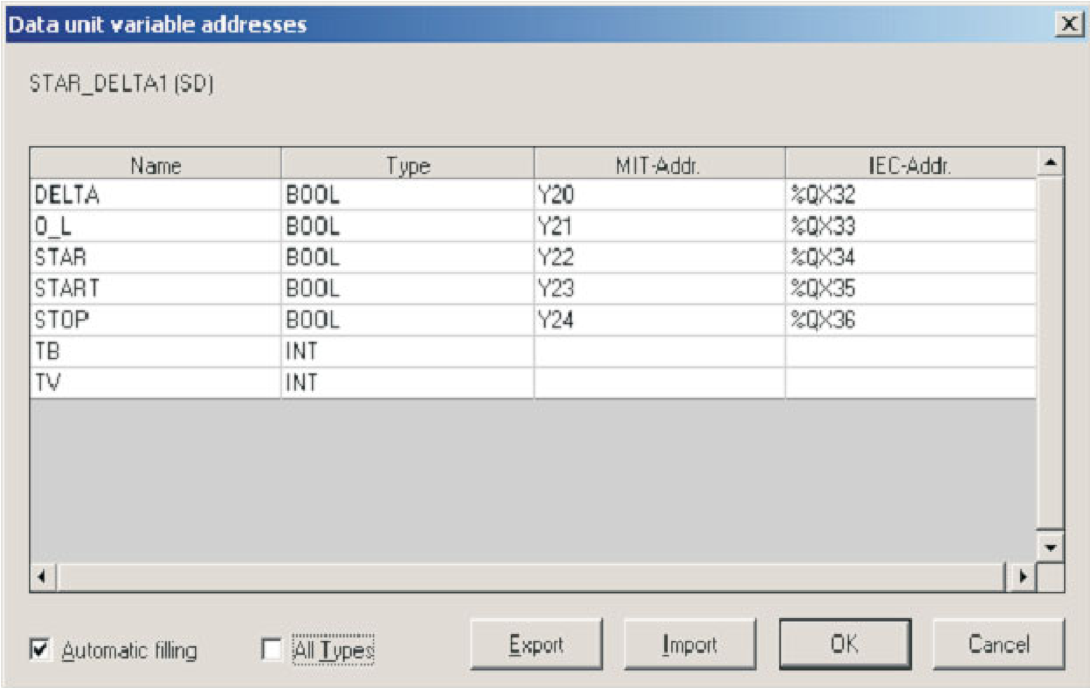
Resulting window:



11.2 Automatic Filling, Variables

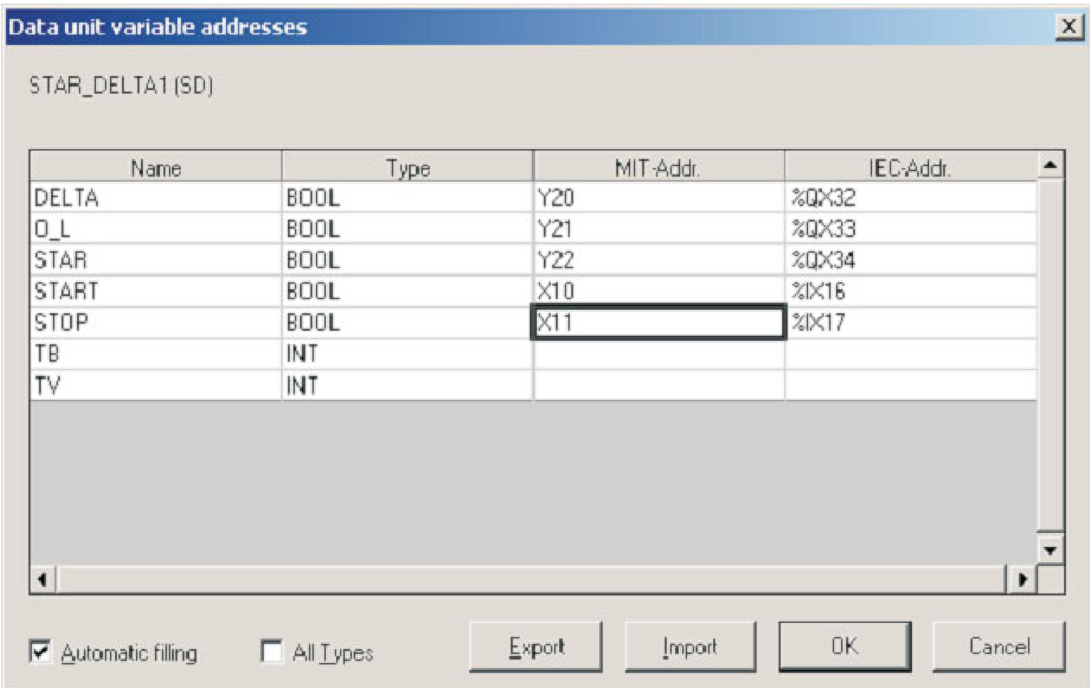
① Deselect ***All types*** as this operation is illegal when using mixed variable types.

② Enter Y20 in the ***MIT-Addr.*** position for the variable: ‘DELTA’:

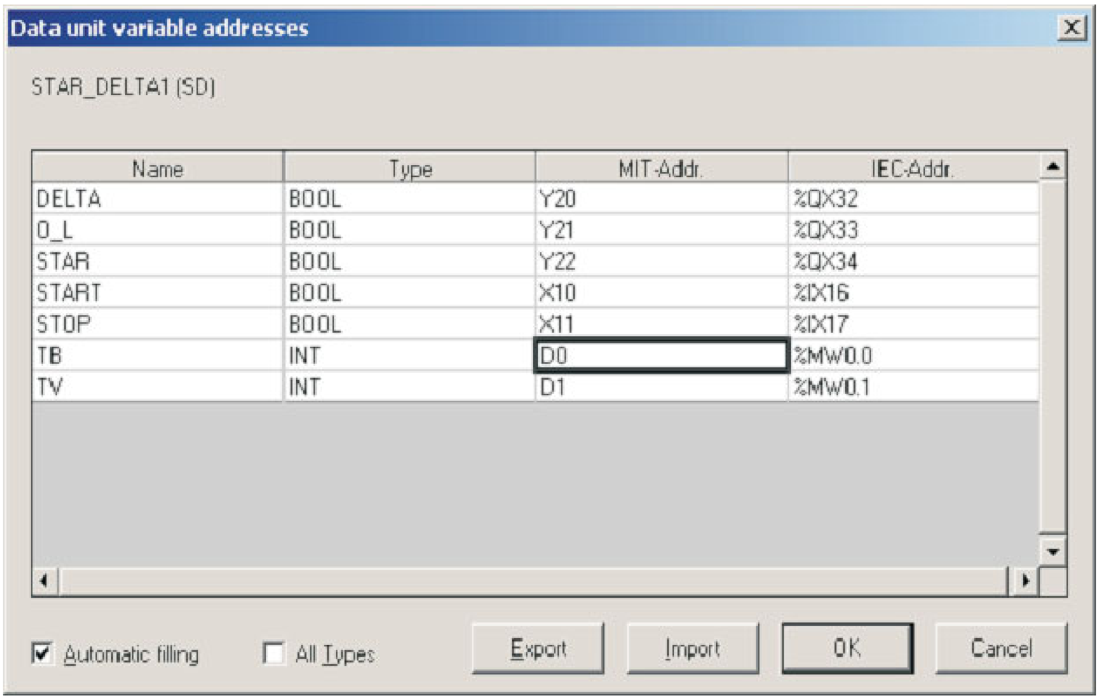


The system will try to sequentially ‘Auto Fill’ the variables of type BOOL. Although in many situations this is recommended, in this case it is only partially successful.

③ Therefore overtype “START and STOP” variables with X10 and X11 thus:



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

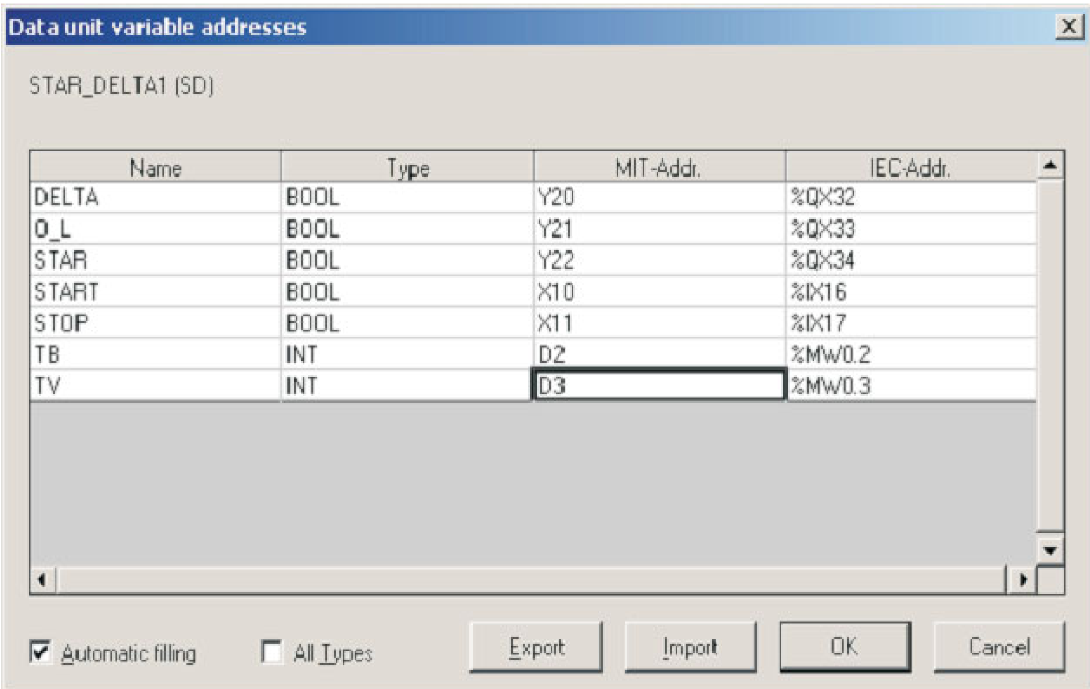
STAR_DELTA1 (SD)

Name	Type	MIT-Addr.	IEC-Addr.
DELTA	BOOL	Y20	%QX32
O_L	BOOL	Y21	%QX33
STAR	BOOL	Y22	%QX34
START	BOOL	X10	%IX16
STOP	BOOL	X11	%IX17
TB	INT	D0	%MW0.0
TV	INT	D1	%MW0.1

Automatic filling · All Types · Export · Import · OK · Cancel

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)

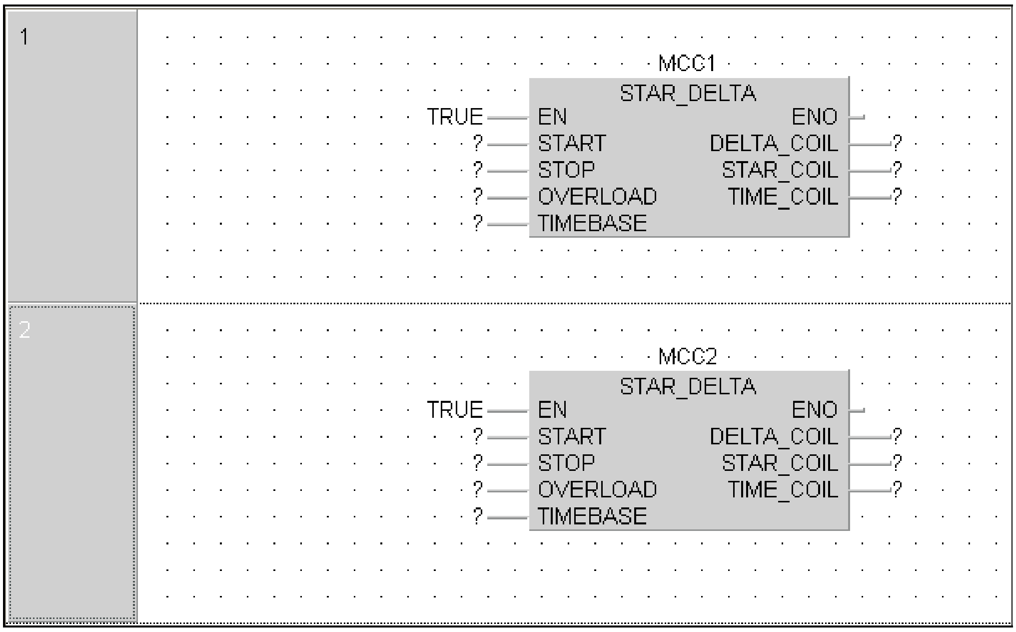
Name	Type	MIT-Addr.	IEC-Addr.
DELTA	BOOL	Y20	%QX32
O_L	BOOL	Y21	%QX33
STAR	BOOL	Y22	%QX34
START	BOOL	X10	%IX16
STOP	BOOL	X11	%IX17
TB	INT	D2	%MW0.2
TV	INT	D3	%MW0.3

Automatic filling · All Types · Export · Import · OK · Cancel

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	

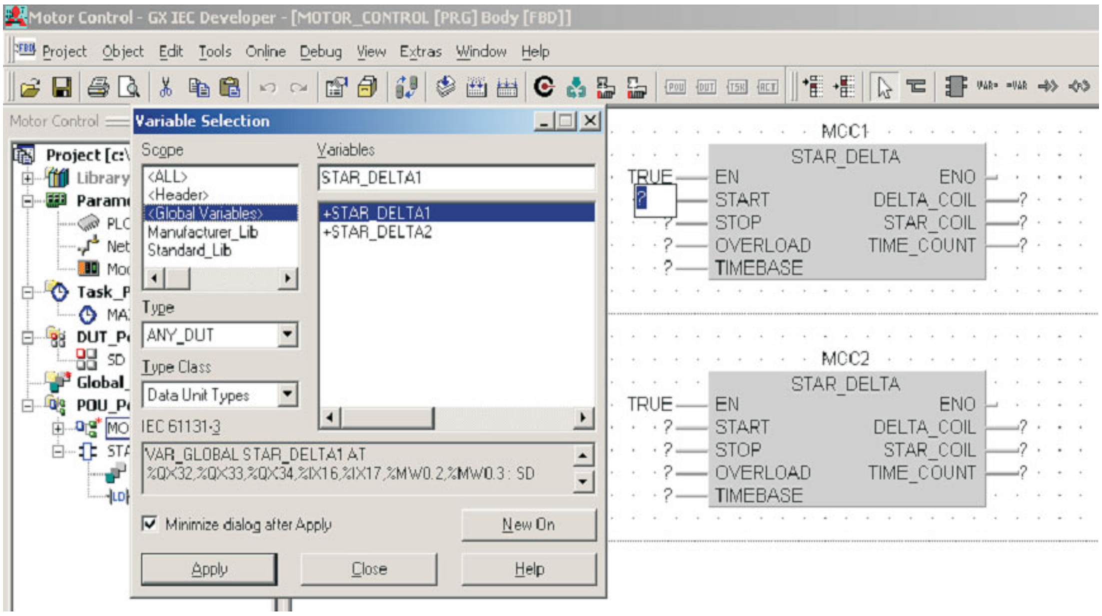
Open the MOTOR_CONTROL program POU and place 2 instances of the user created Function Block STAR_DELTA as shown:



11.3 Assigning DUT Variables to Function Blocks

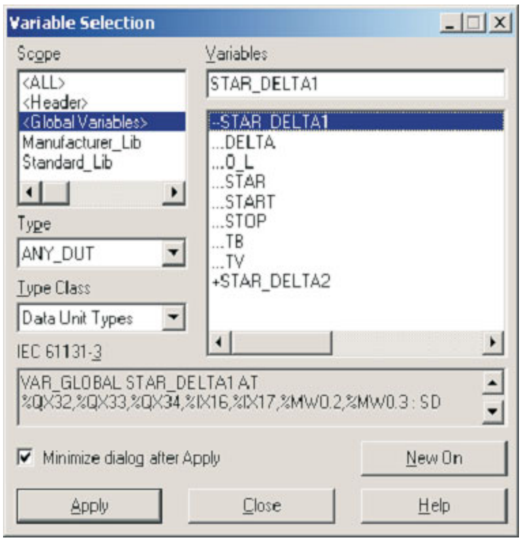
To assign variables to the Function blocks...

① ...right Click on a variable (or F2). The following variable selection window appears:

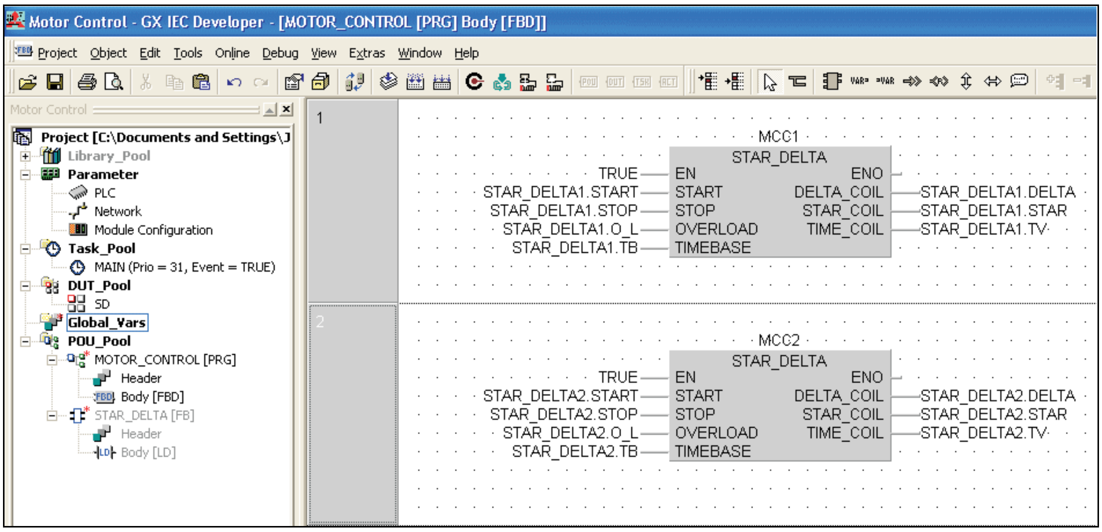


② Set the ***Scope*** to ***Header***, ***Type Class*** to ***Data Unit Types*** and ***Type*** to ***ANY_DUT***.

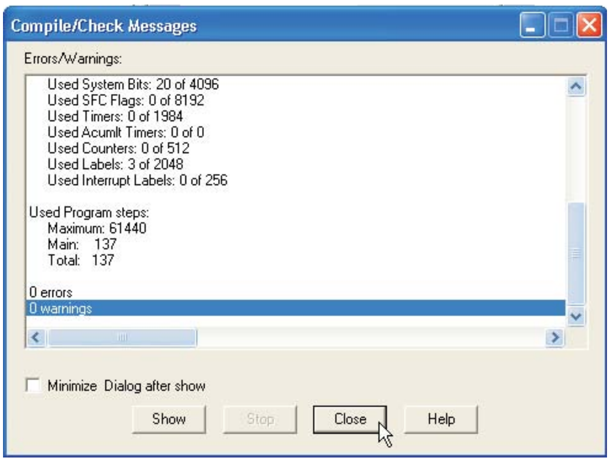
③ Double Click on +STAR_DELTA1 and the following expanded DUT variable list appears:



④ Pick and assign the variables to the two STAR_DELTA Function Blocks on the MOTOR_CONTROL Program POU as shown:

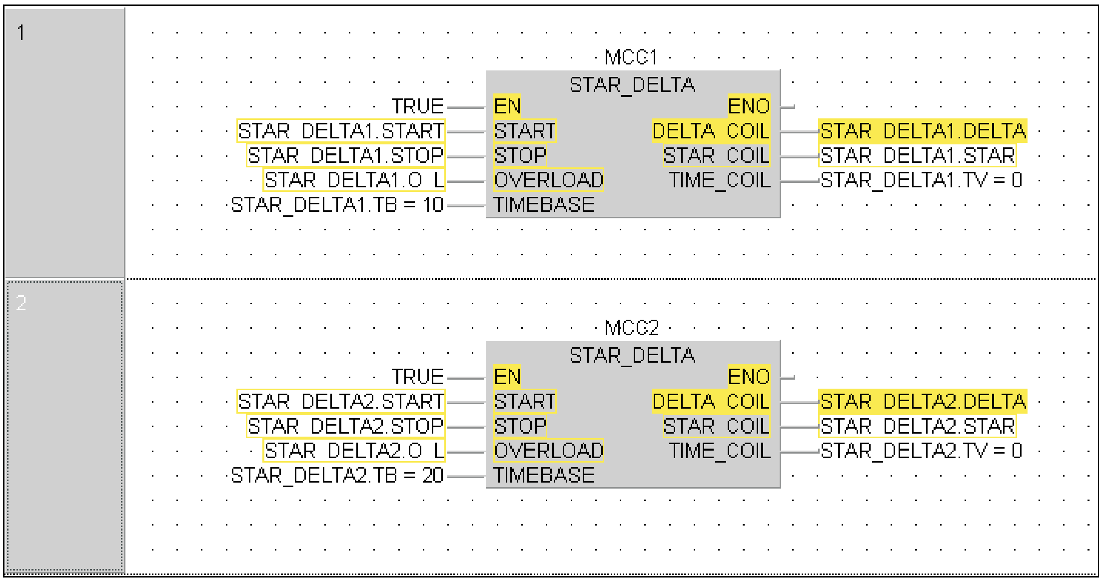


Save the project and ***Rebuild All*** to compile the code:



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

12 Arrays

12.1 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

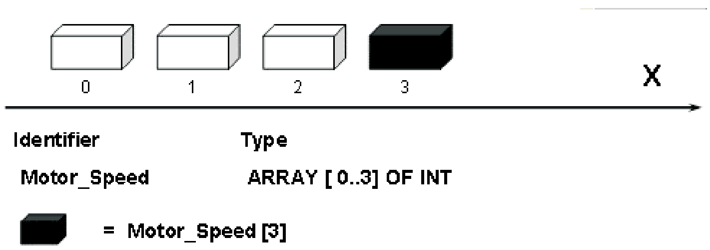
In software, program elements can use: Motor_Volts[1] and Motor_Volts[2], as declarations, which in this example mean that D1 and D2 are addressed.

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.

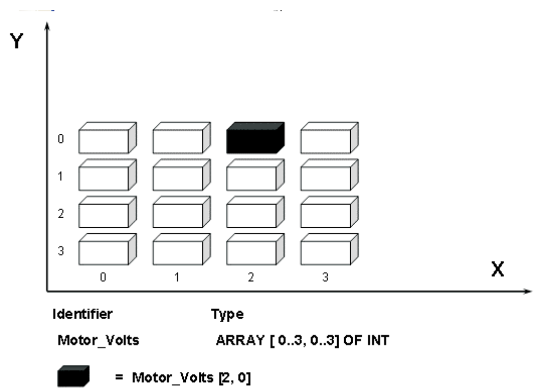
Arrays can provide a convenient way of 'indexing' tag names, i.e. one declaration in the Local or Global Variable Table can access many elements.

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

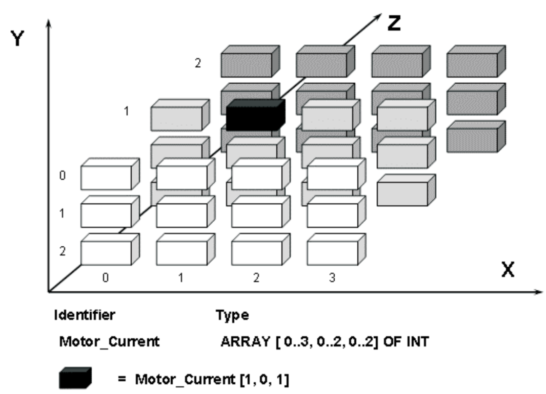
Single Dimensional Array



Two Dimensional Array



Three Dimensional Array

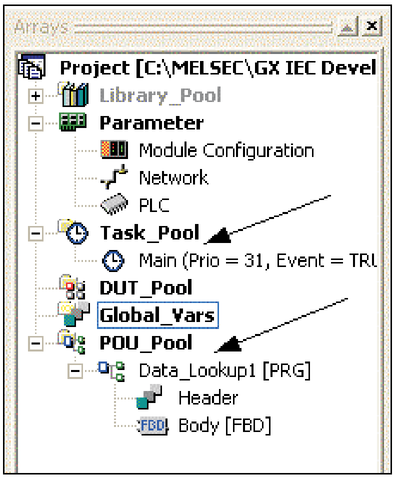


12.2 Array Example: Single Dimension Array

The following example is used to illustrate a single dimension array. The array is 10 words long and uses Global MELSEC addresses D100-D109. This example uses only “Standard IEC” Operators, Functions and Function Blocks.

① Create a new project and define 1 new POU of Class “Program” using a body of Language ***FBD*** and named “Data_Lookup1”

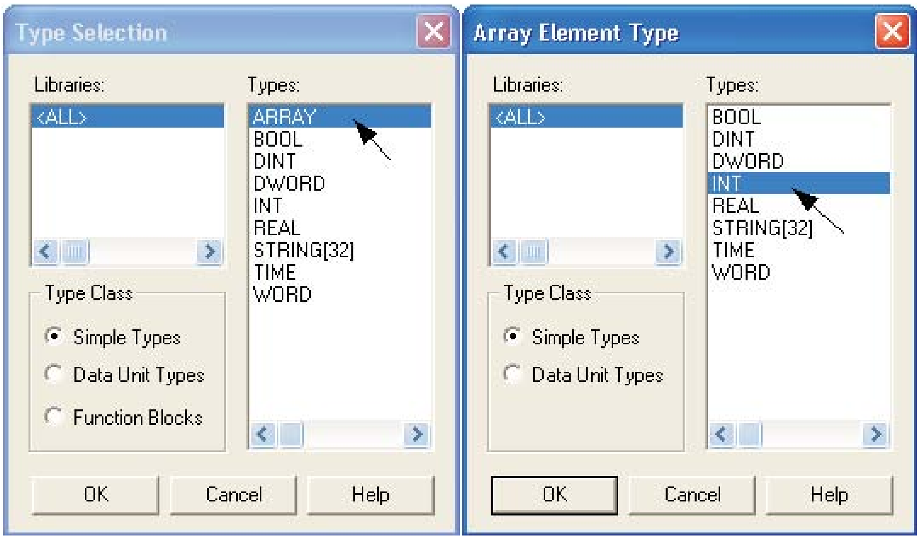
② Create a new Task in the task pool named “Main” and bind the program POU “Data_Lookup1” to it:



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

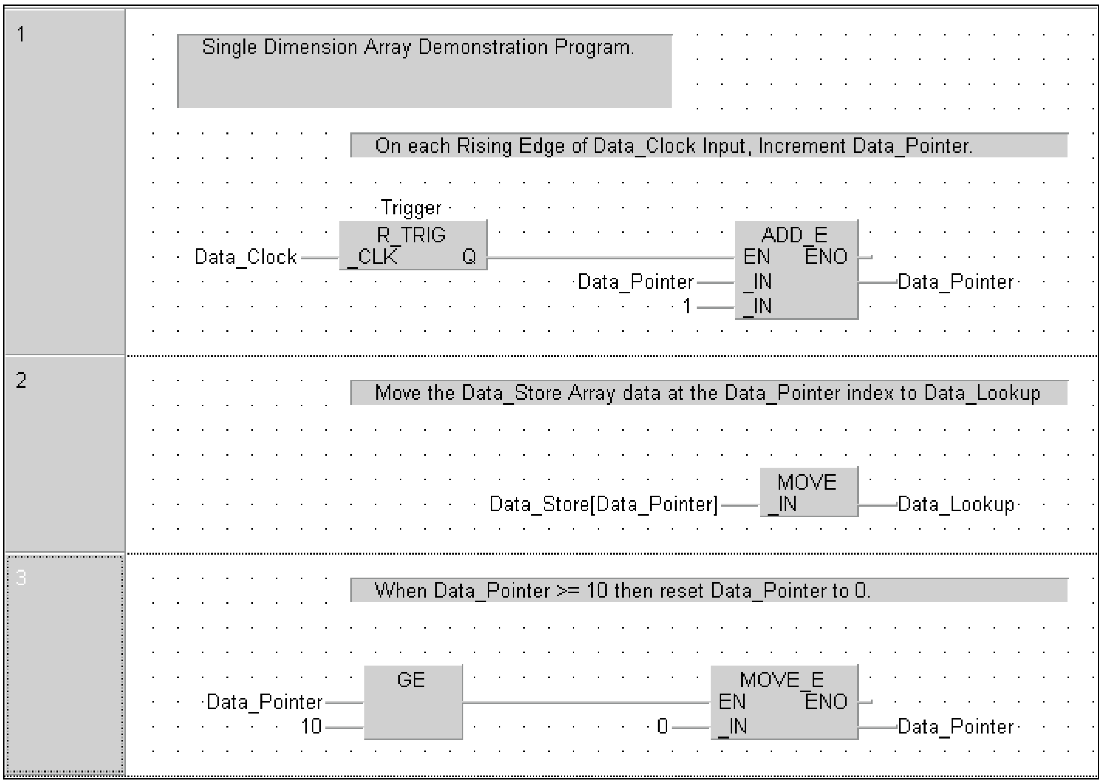
Note: The variable type “Array” in entered as follows:



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:



Note: Define the 'R_Trig' Function block with instance name "Trigger".

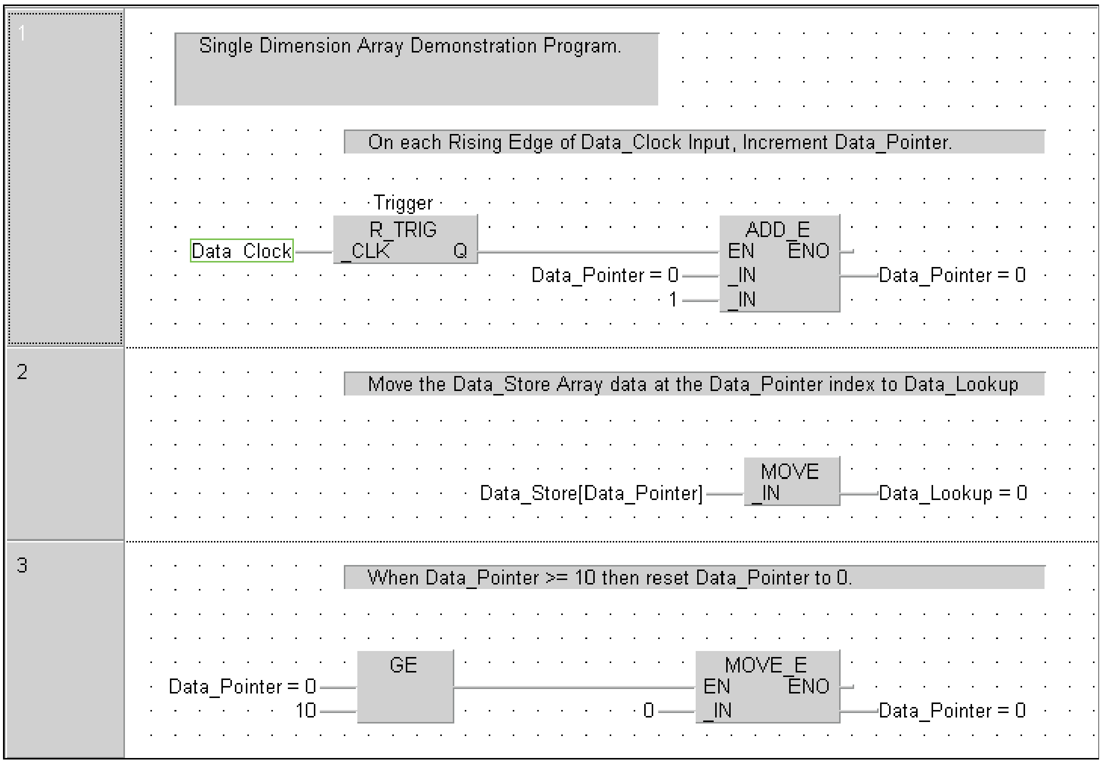
⑤ Check the Header reads as shown below:

	Class	Identifier	Type	Initial	Comment
0	VAR	Trigger	R_TRIG		

⑥ Save the program and use ***Rebuild All*** to compile the program.

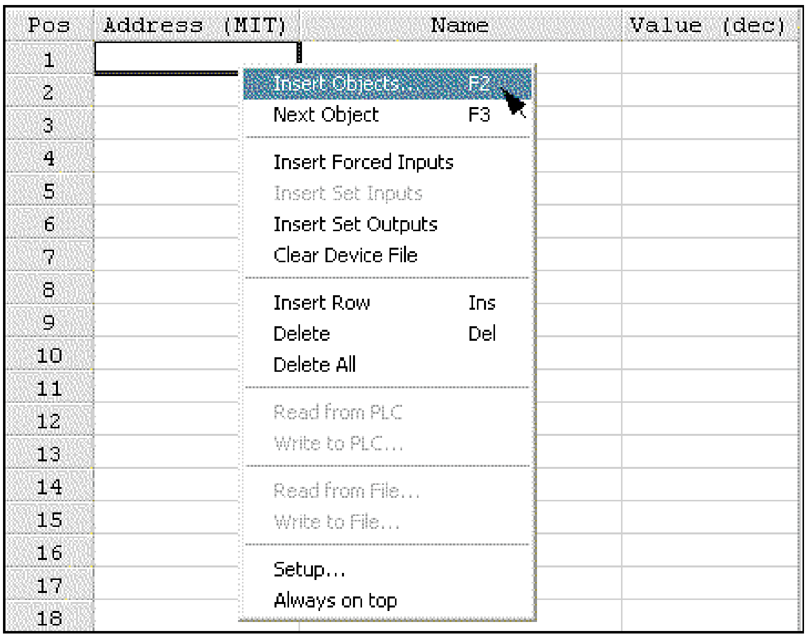
⑦ Transfer the program to the PLC.

⑧ Monitor the POU body (see next page)

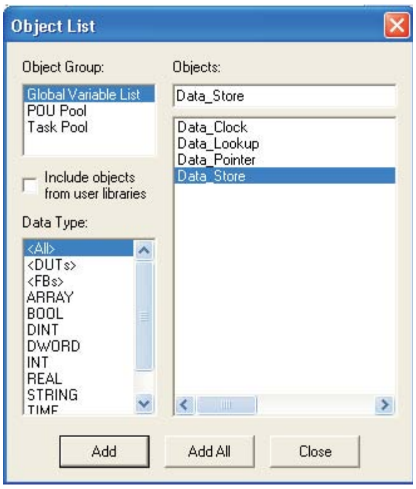


Before the program is able to function as intended it is necessary to input data into the physical MELSEC addresses occupied by the array variables. There are two ways in which this may be achieved:

- 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***:



– 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:

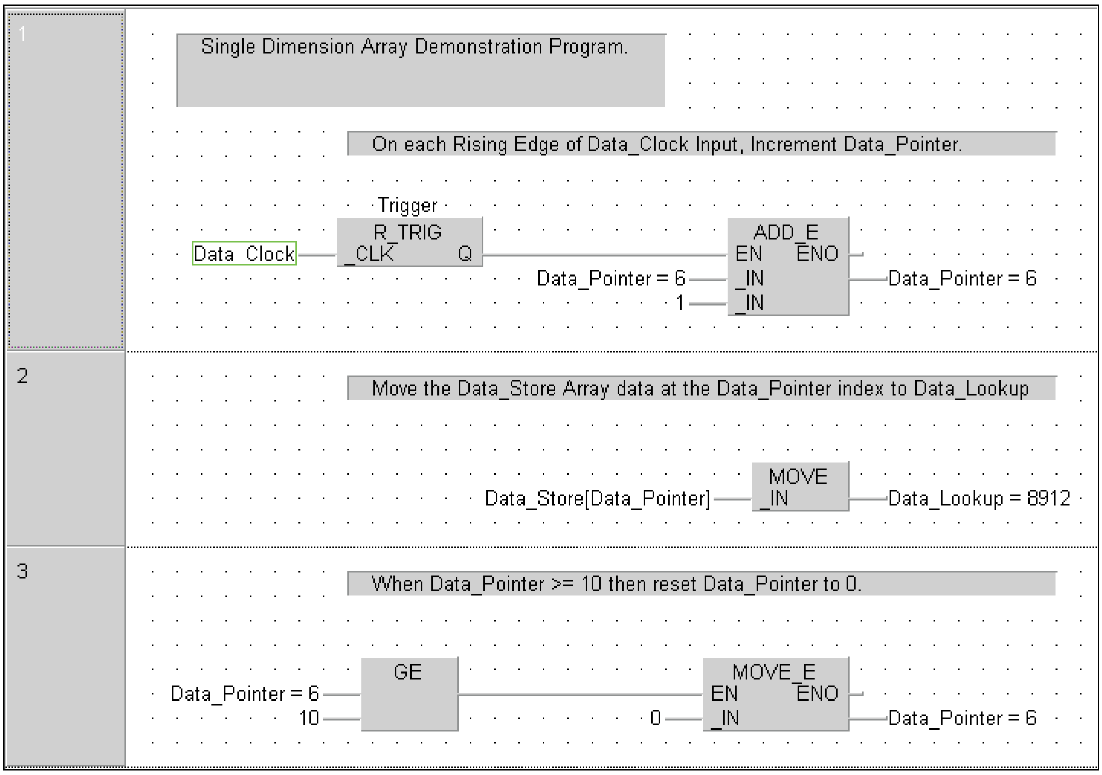
Pos	Address (MIT)	Name	Value (dec)
1		-Data Store	
2	D100	[0]	0
3	D101	[1]	0
4	D102	[2]	0
5	D103	[3]	0
6	D104	[4]	0
7	D105	[5]	0
8	D106	[6]	0
9	D107	[7]	0
10	D108	[8]	0
11	D109	[9]	0

– 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:



- 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).

13 Working with Libraries

13.1 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.

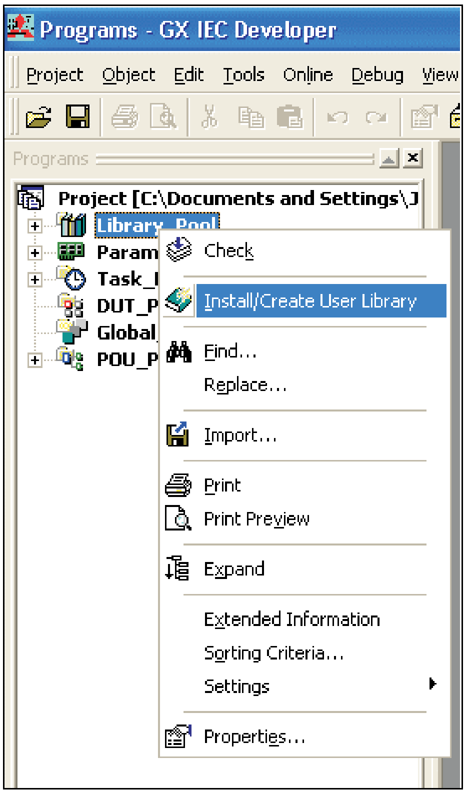
As already seen, when called program functions, the ***Standard Library*** contains IEC functions. The ***Manufacturer Library*** contains Mitsubishi functions (denoted by *_M) – M meaning manufacturer not Mitsubishi!

Any user defined libraries will also appear on this list.

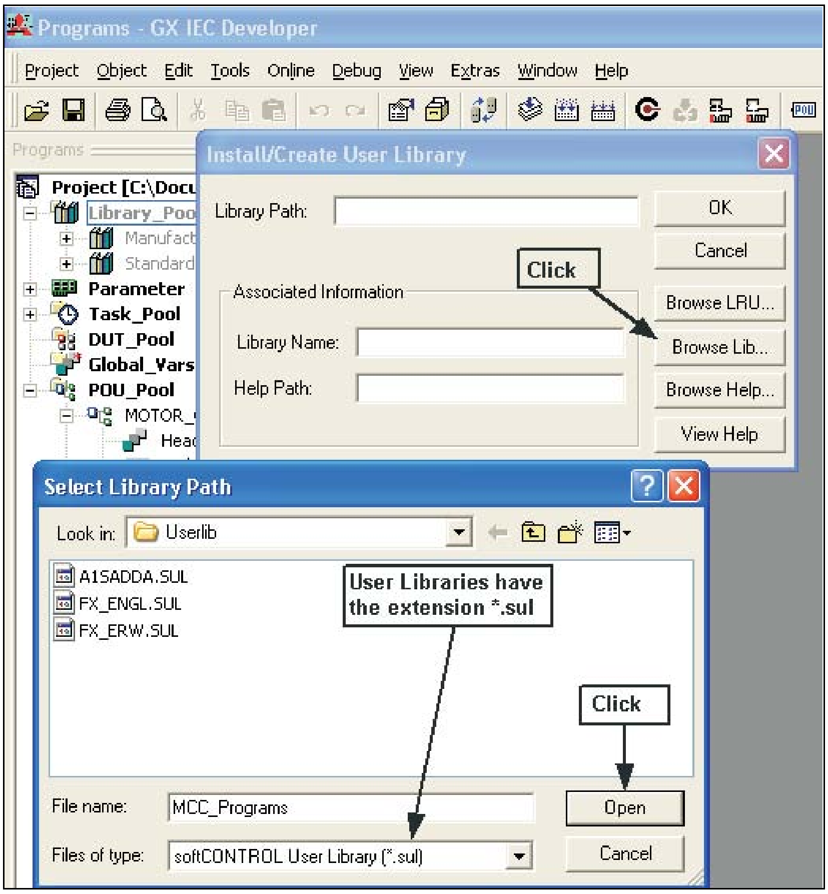
13.1.1 Example – Creating a new Library

① Assign the function block STAR_DELTA to a new library.

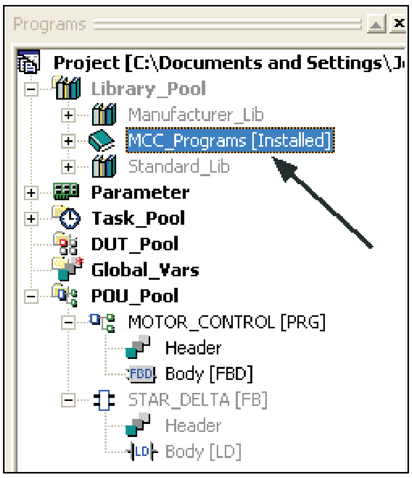
② Right Click the Library Pool, in the Project Navigator window and from the displayed menu select ***User Library*** and ***Install/Create Library***.



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”.



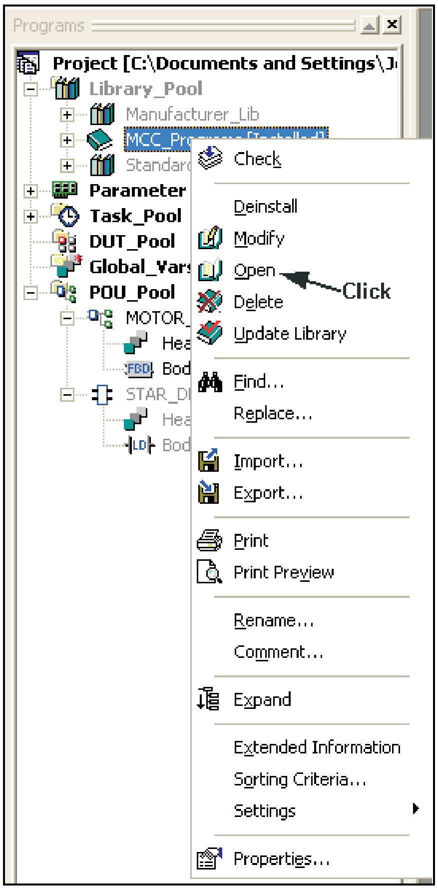
④ Click ***Open*** when done:



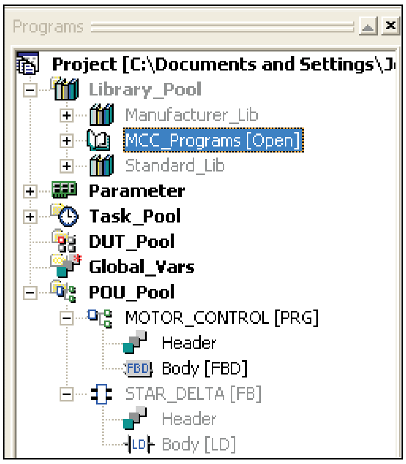
Notice the new Library “MCC_Programs” that is now present in the project Library Pool.

13.1.2 Opening the Library

① Open the Library by right clicking on the icon 'MCC_Programs' and click on ***Open*** from the menu:



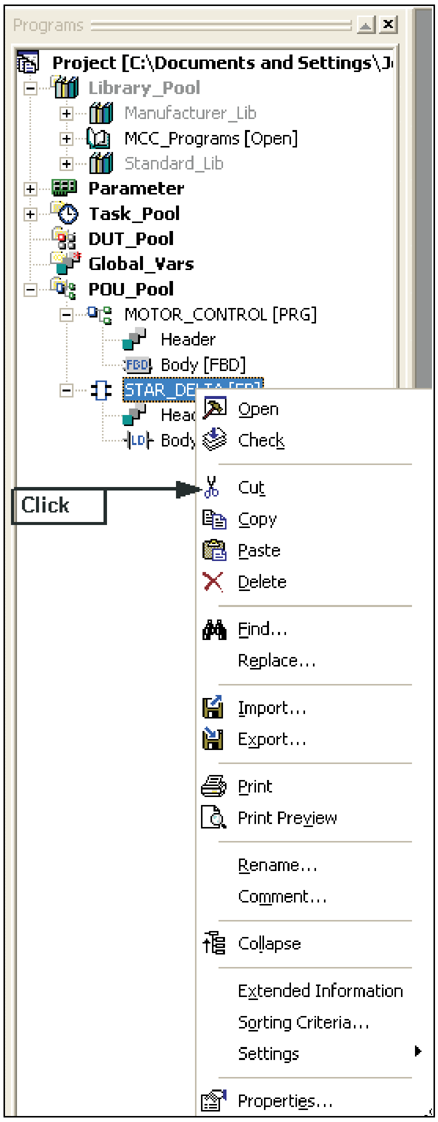
The Library is now open and may be accessed and edited:



13.1.3 Moving a POU "Function Block" to an open Library

The Function Block STAR_DELTA will now be moved into the Library 'MCC_Programs'.

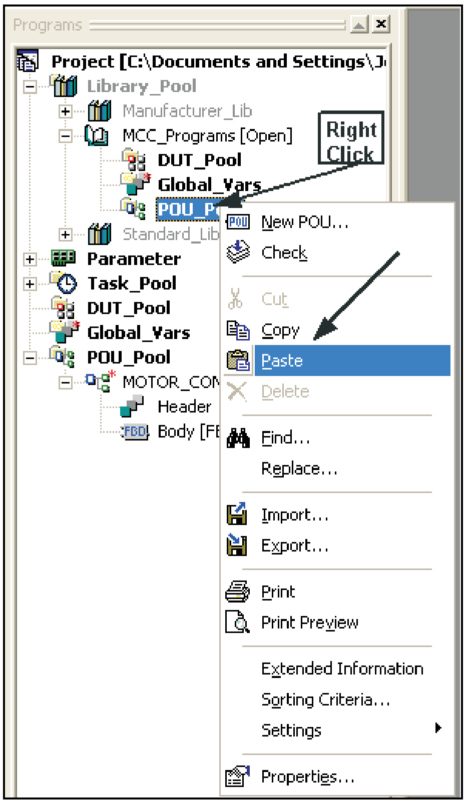
① Right Click on the STAR_DELTA icon in the Project navigation window and click on "Cut":



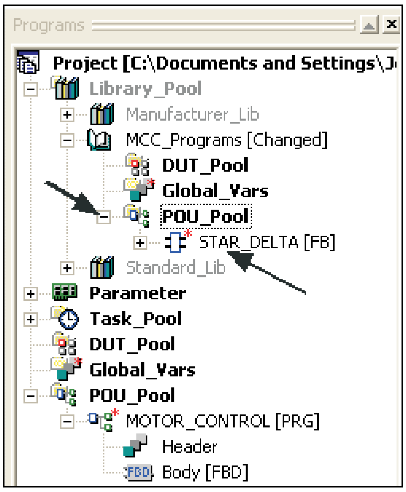
The following dialogue will be displayed:

② Select ***Yes***

③ Right Click on the User Library icon and select ***Paste*** from the menu:



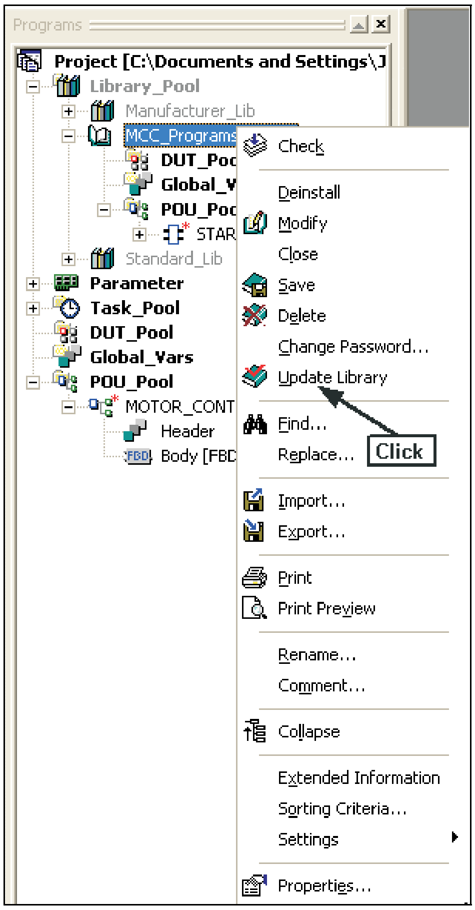
④ Click on the '+' on the new entry in the Library POU Pool to expand the 'STAR_DELTA' Function Block:



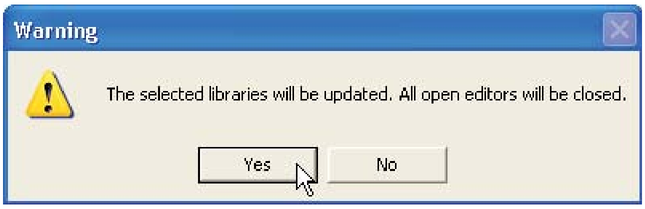
The Function Block POU, "STAR_DELTA" is now present in the Library "MCC_Programs" and no longer in the Project POU Pool.

Any POU; Function, Function Block, PRG or DUT can be added to the Library in this way.

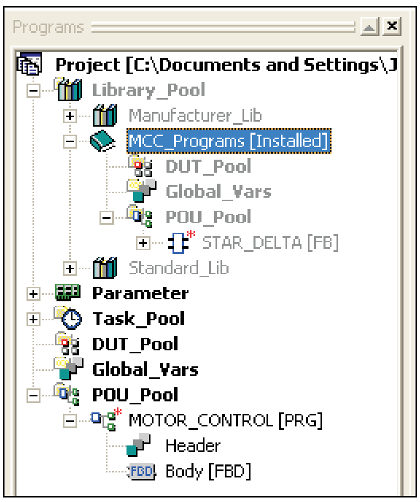
⑤ When editing of the library is complete, click ***Update Library***. This will update and close the library.



The following message will be displayed:

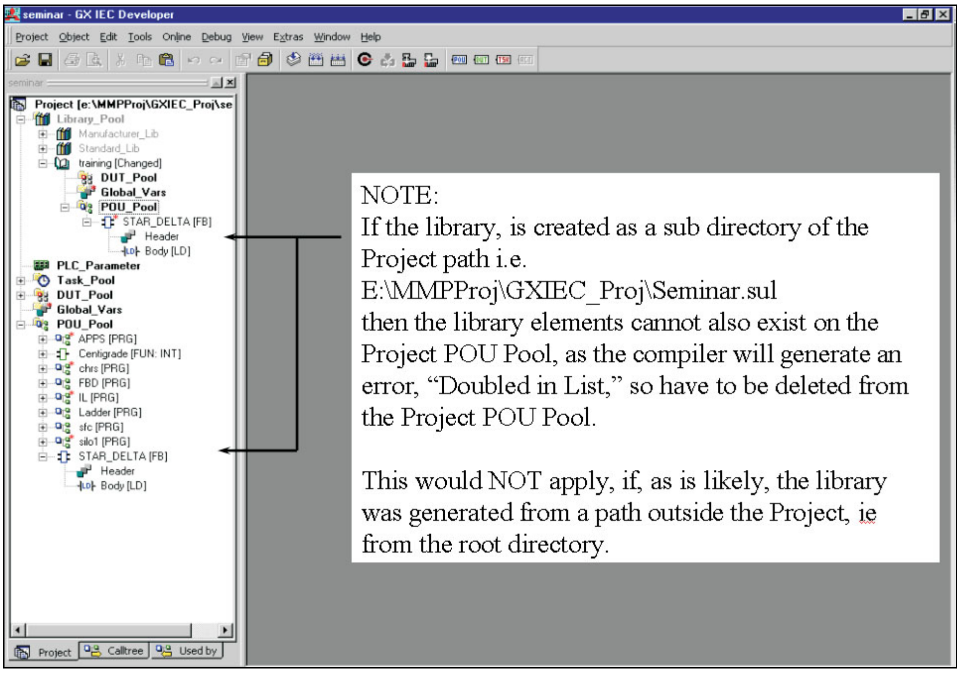


⑥ Click ***Yes*** and the library will be updated, saved and closed.



The library is now stored in the default location of “C:\MELSEC\GX IEC DEVELOPER 7.00\Userlib” as set when creating the library.

13.2 Special Note about Libraries



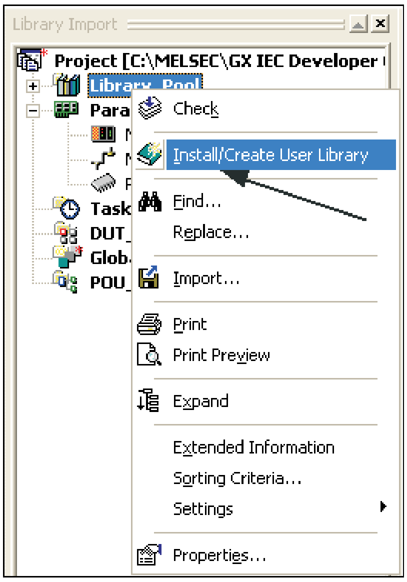
13.3 Importing Libraries into projects

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.

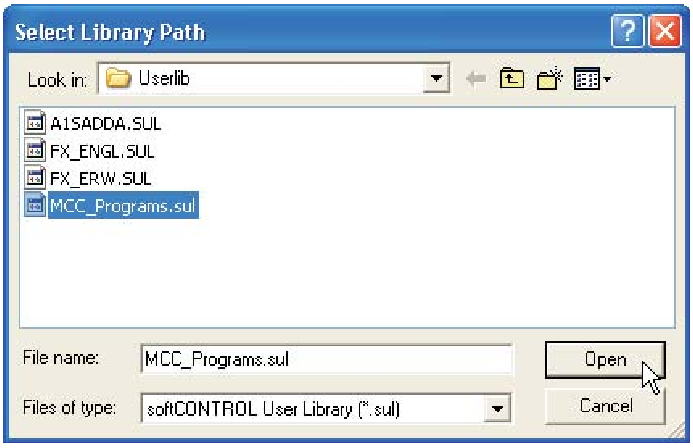
The following two examples describe the methods used to import Libraries into working applications:

The previously saved Library "MCC_Programs" will be imported into the current project and the Function Block contained therein will be re-used.

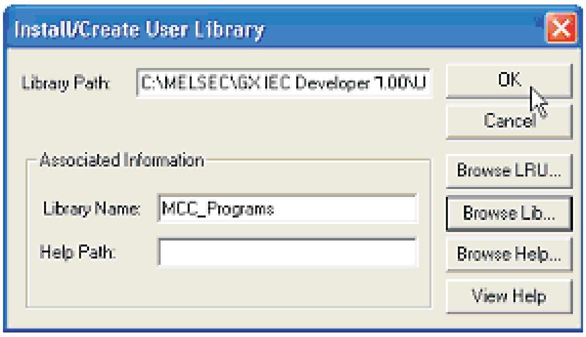
① Create a new empty project with no POU's called "Library Import".



② Enter the following details into the prompt:



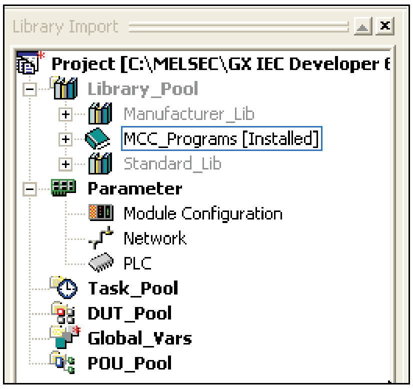
③ Next click ***OK*** to accept the entries.



NOTE

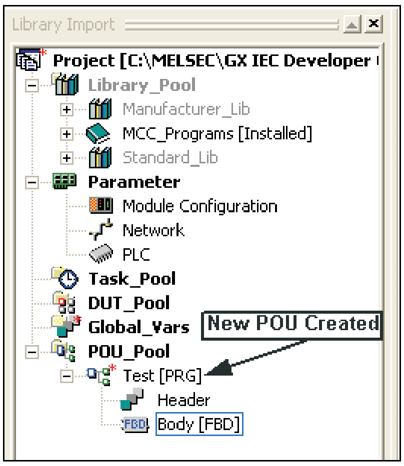
The help path is used for user help files that can be created in order to describe the operation of routines held in the library. These files can be created in MS-Word, for example in HTML format and manually saved with the reserved extension *.CHM. These files can be bound to the library by clicking ***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:

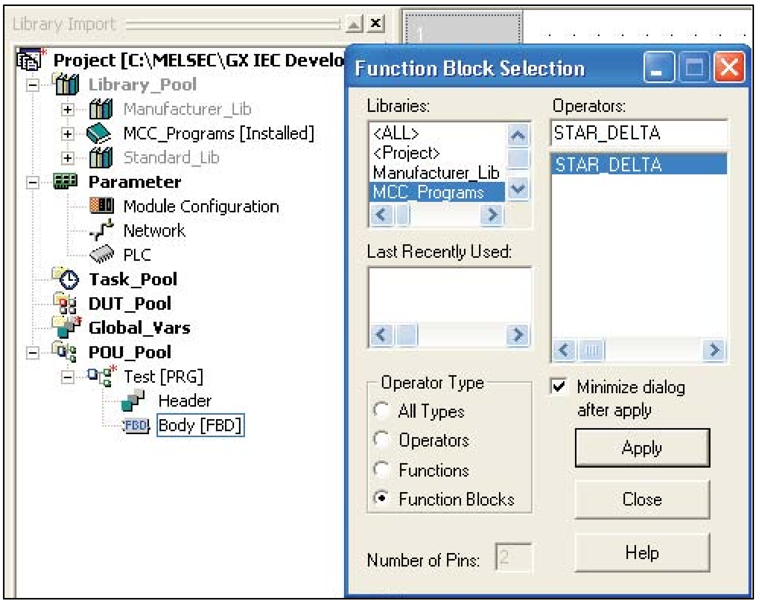


Items stored in libraries can be easily recalled and selected into a project, as shown in the following illustrations:

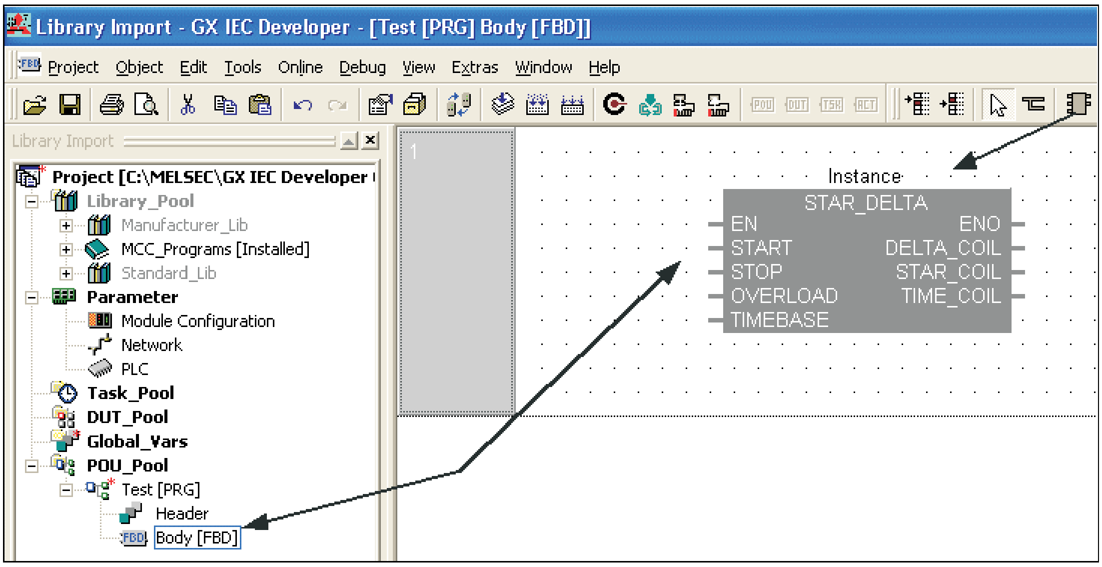
① Create a new POU, type: ***FBD*** and named “Test”:



② Open the new POU and select the Function Block as shown:



As can be seen the new library appears in the domain and may be selected as shown:



13.3.1 Example: Importing a Mitsubishi Library Function Block

The following illustrations demonstrate the procedures required to import a Mitsubishi Function Block for Analogue Input using a Q-Series Module Q64AD.

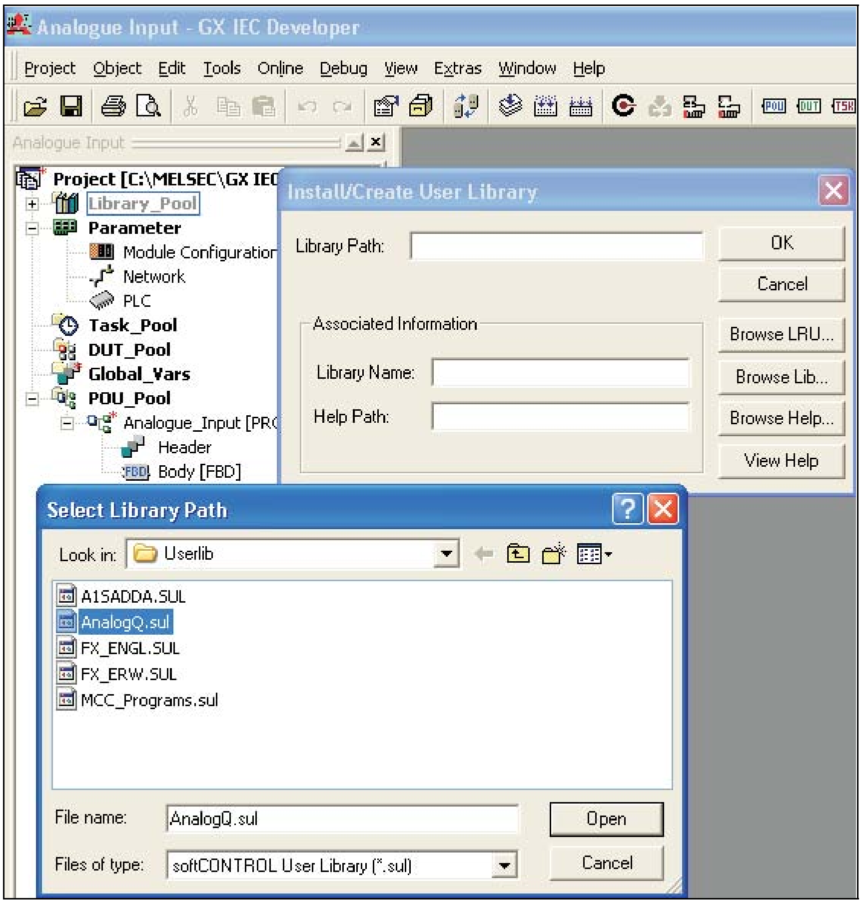
In order for the following example to function correctly, it is necessary to install the Mitsubishi Q-Series Analogue Library into the project.

The Analogue Function Block library “AnalogQ” is to be found on the Mitsubishi Website or can be installed directly from the GX-IEC Developer disk from the Function Block selection on the installer program. The Library can now be accessed from the “Userlib” directory.

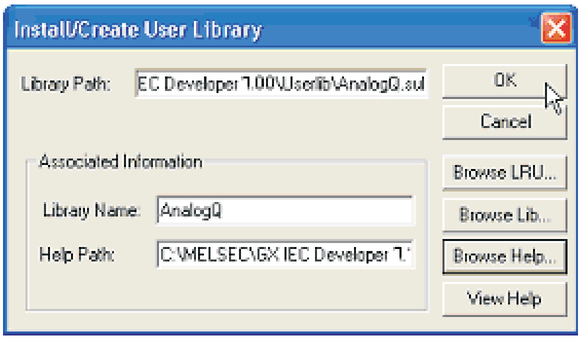
① Create a new empty project with no POU’s called “Analogue_Demo”.

② Create a new POU Type: ***FBD***, Class: ***PRG***, and name it “Analogue_Input”

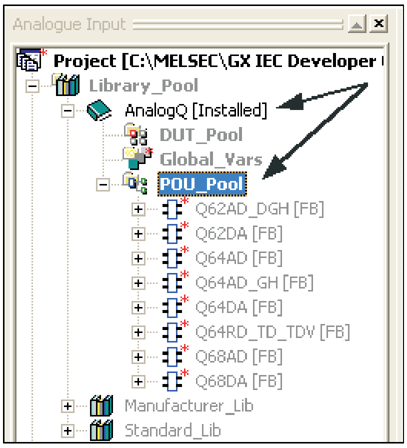
③ Right Click on the Library_Pool Icon and select ***Browse Lib***. Select The AnalogQ.sul library file and click ***Open***.



④ Click ***OK*** on the ***Install/Create User Library*** prompt:

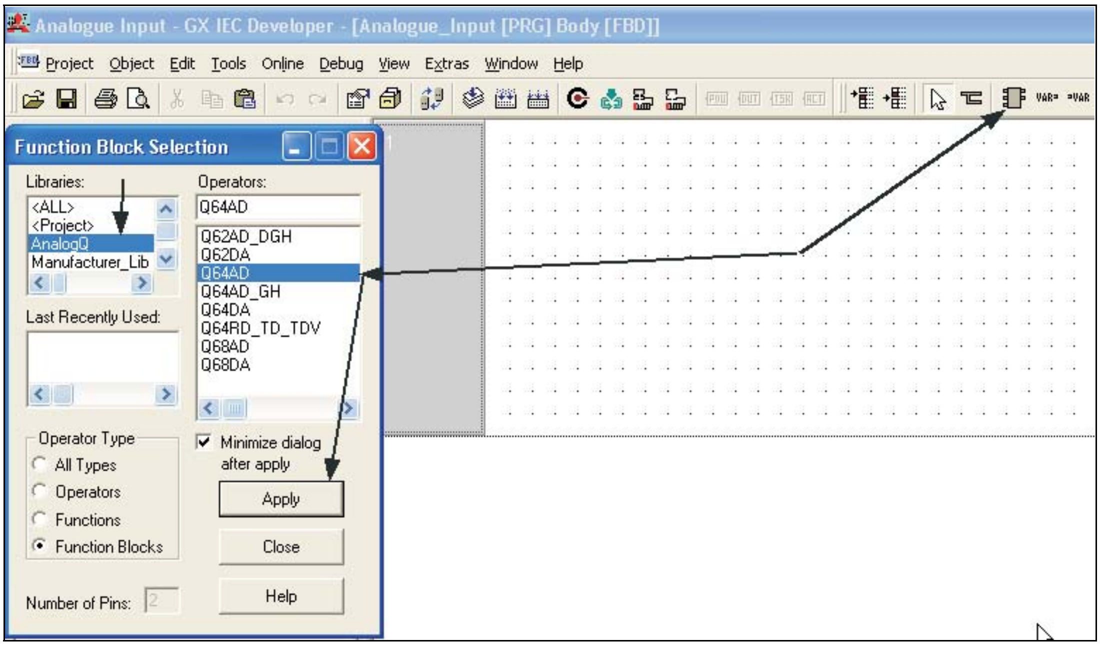


Note the new “AnalogQ” Library in the Project Navigation Window.

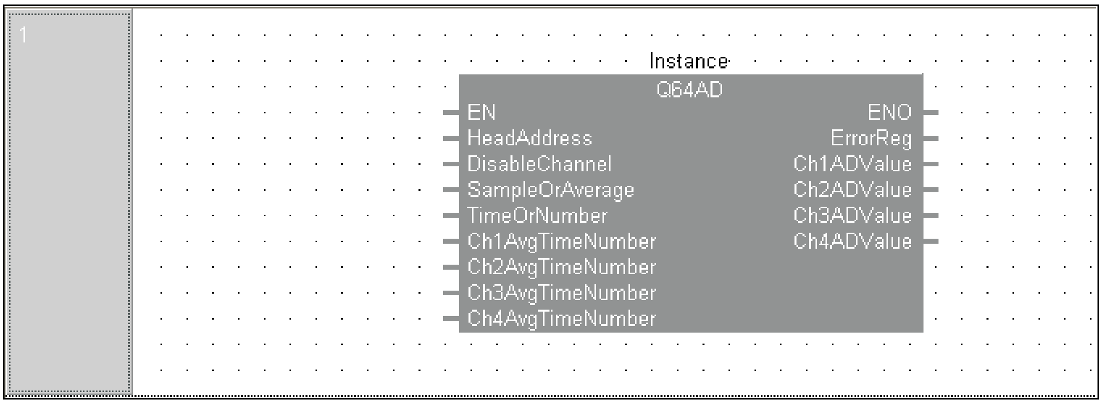


⑤ Create a new task in the task pool: “MAIN” and bind the POU “Analogue_Input” to it.

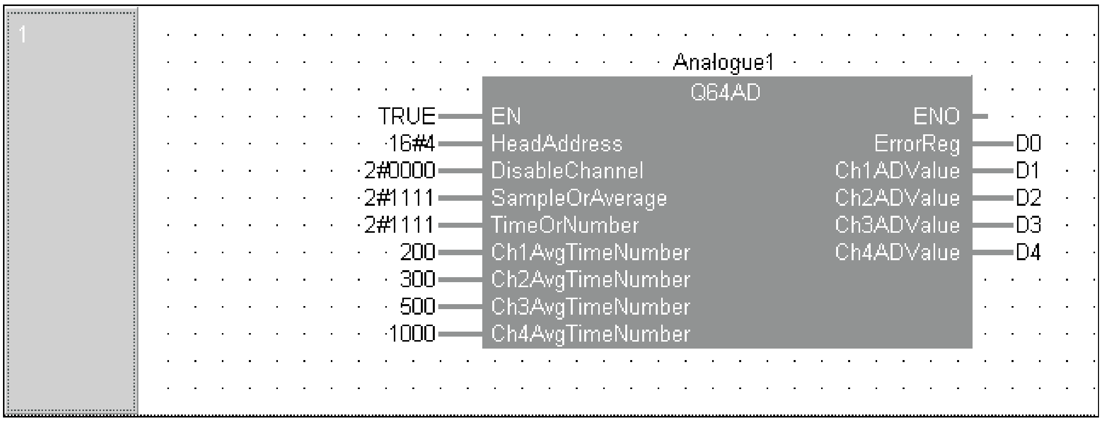
⑥ Place the Q64AD Function Block into the POU as shown below:



The Function Block will appear thus:



⑦ Define all variables as below:



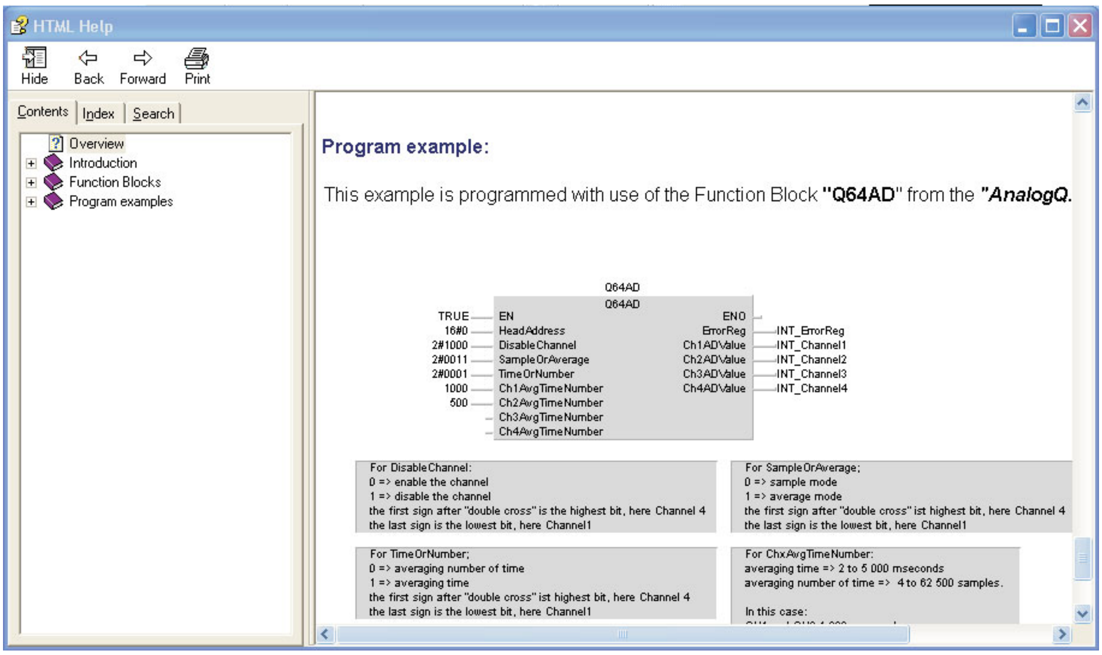
⑧ Compile and download the program to the PLC.

⑨ Monitor and test for correct operation. Observe the behaviour of the analogue outputs due to the “sampling settings”

13.3.2 Library Function Block Help

Providing the accompanying Library Help file has been imported, for a full explanation with examples of all Analogue Q Library Function Blocks, click to highlight the Function Block and press the “F1” Key.

The following HTML Help Screen will be displayed:



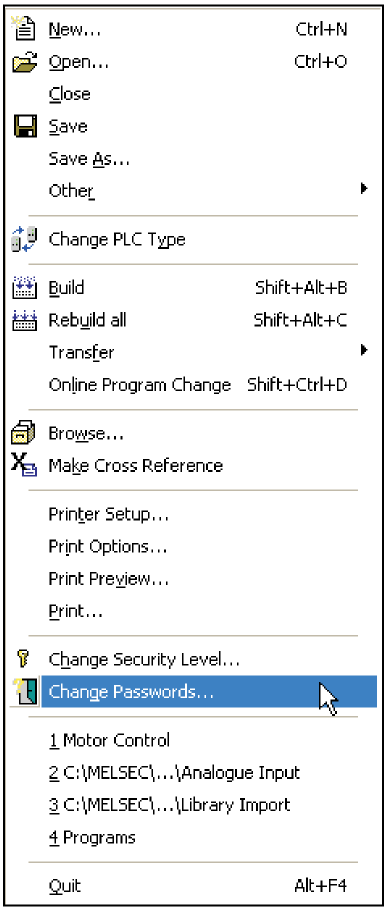
The Help files cover every aspect from the setup of the Q-Series analogue hardware modules to use of the library function Blocks.

14 Security

14.1 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.

14.1.1 Setting the Password

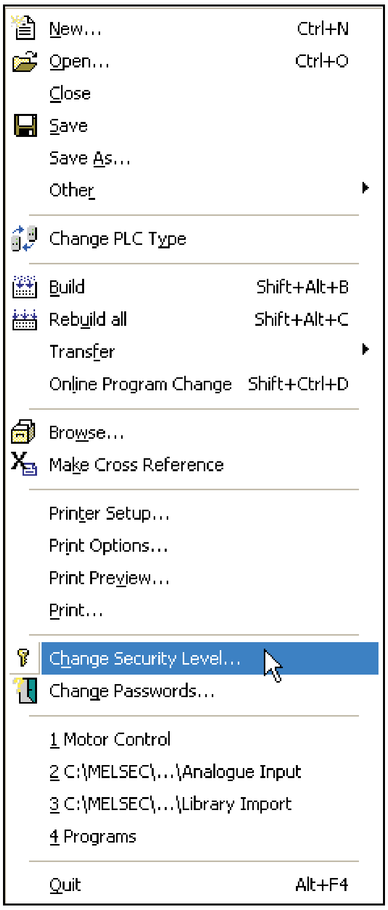


Passwords can be entered and security levels can be changed, using these windows, via the ***Project*** menu.

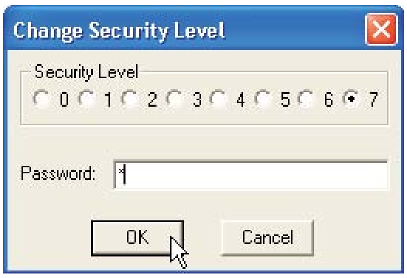
To illustrate the operation of passwords, select ***Security Level*** 7 and enter a new password for this level (For simplicity here, press 7). Re-enter the password and click ***Change***.

14.1.2 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.



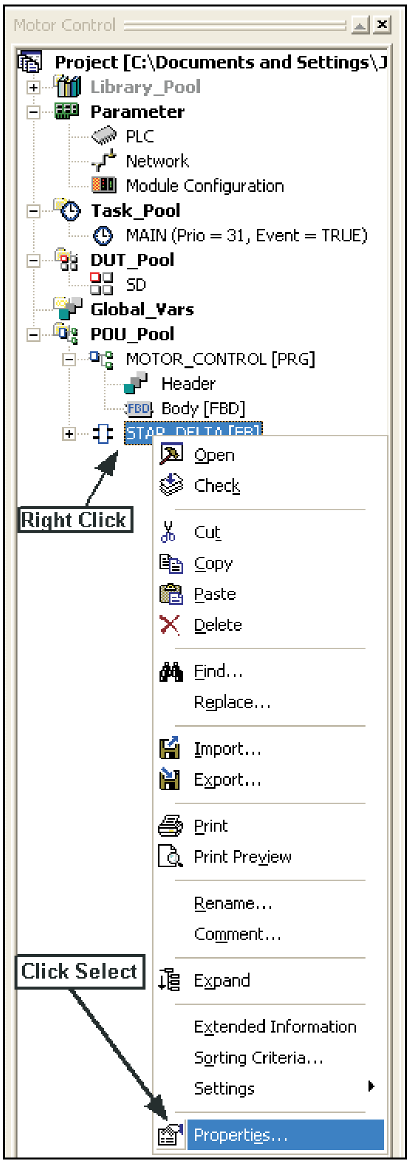
Once logged on, the security attributes for many items may be altered. For example one of the most common security options is to change access to POU’s, i.e. User Functions and Function Blocks.

14.1.3 Modifying POU Password Access

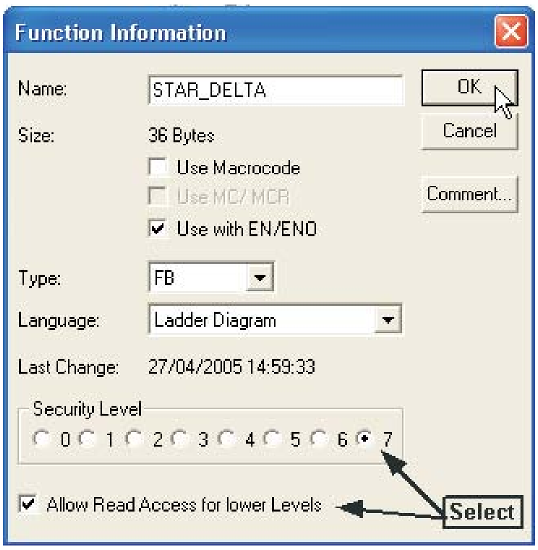
In order to protect the content or control access to User POU's the security attributes may be adjusted, whilst being logged into the security current level, as follows:

Setting Security Level

① Open the project "Motor Control" and open the header of the Function Block "STAR_DELTA":



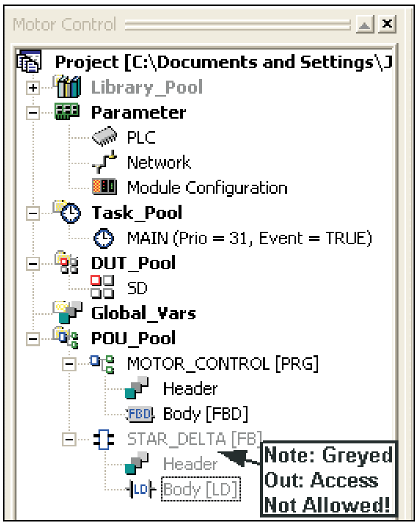
② Adjust the Security to Level ‘7’ and click ***Allow Read Access for lower Levels***. This will allow subordinate users “Read access” only to the Header and body of the function Block:



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.

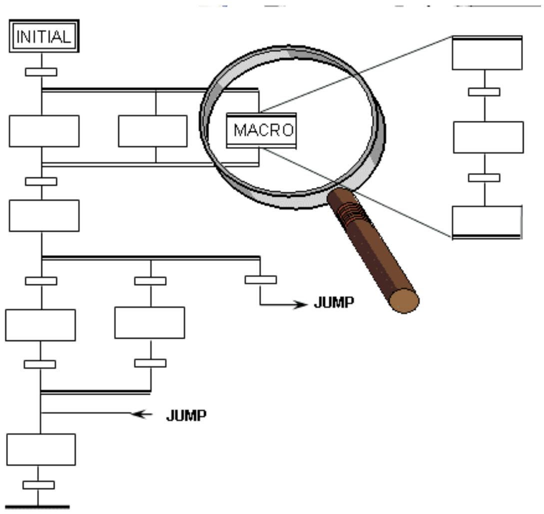
⑤ Change the security level to ‘0’ and try to access the body of the Function Block “STAR_DELTA”. The Header and Body of the POU will be greyed out with access to the POU completely blocked:



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.

15 Sequential Function Chart - SFC

15.1 What is SFC?

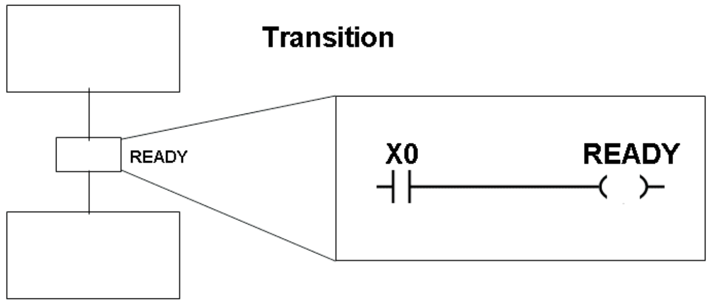


- The “Sequential Function Chart” editor is a guided editor.
- Graphical Flowchart representation.
- Based on the French Grafcet (IEC 848)
- SFC is a structural language which divides the process into steps and transitions.
- The steps “hide” actions (no POUs) and / or directly switched bit operands.
- 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).

15.2 SFC Elements

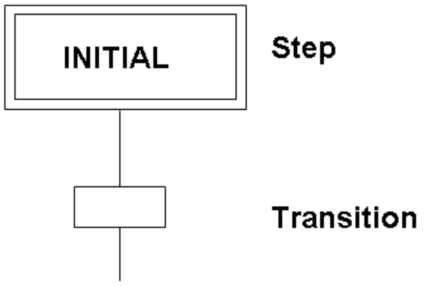
15.2.1 SFC Transitions



- 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.

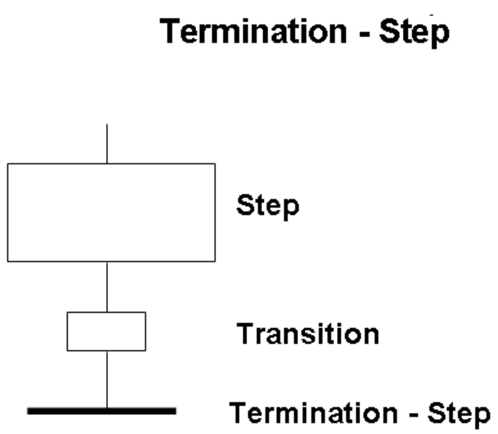
15.2.2 Initial Step

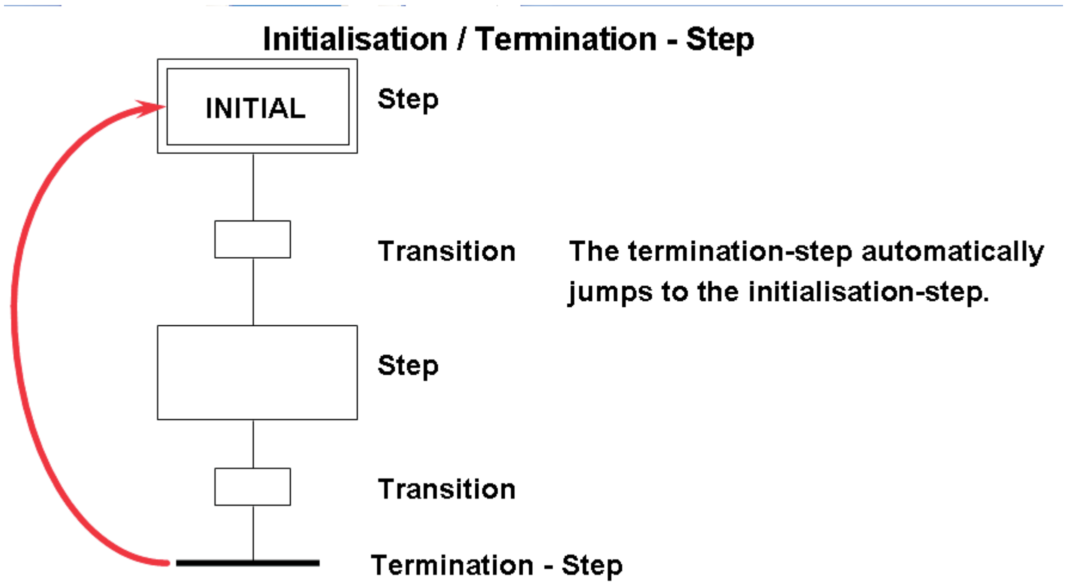
SFC programs begin with an Initial Step function which indicates the start of a sequence:



15.2.3 Termination Step

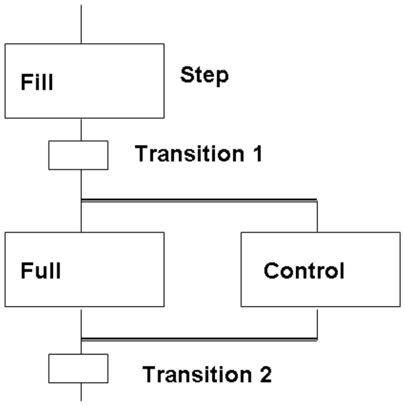
All Sequences finish with a Termination Step:



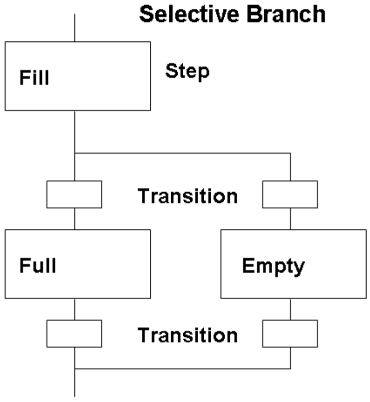

Initialisation / Termination - Step
INITIAL
Step
Transition
The termination-step automatically jumps to the initialisation-step.
Step
Transition
Termination - Step

15.3 SFC configuration examples

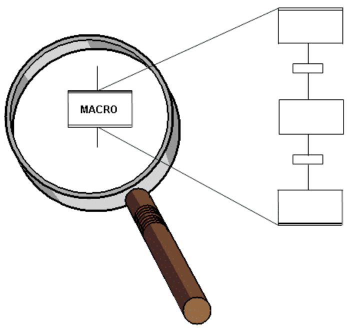
Parallel Branch



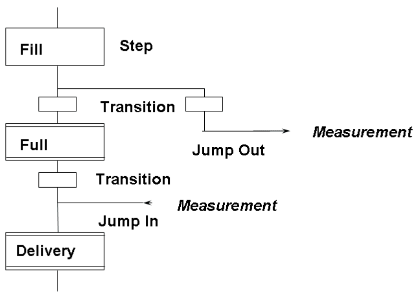
Selective Branch



Macro - Step

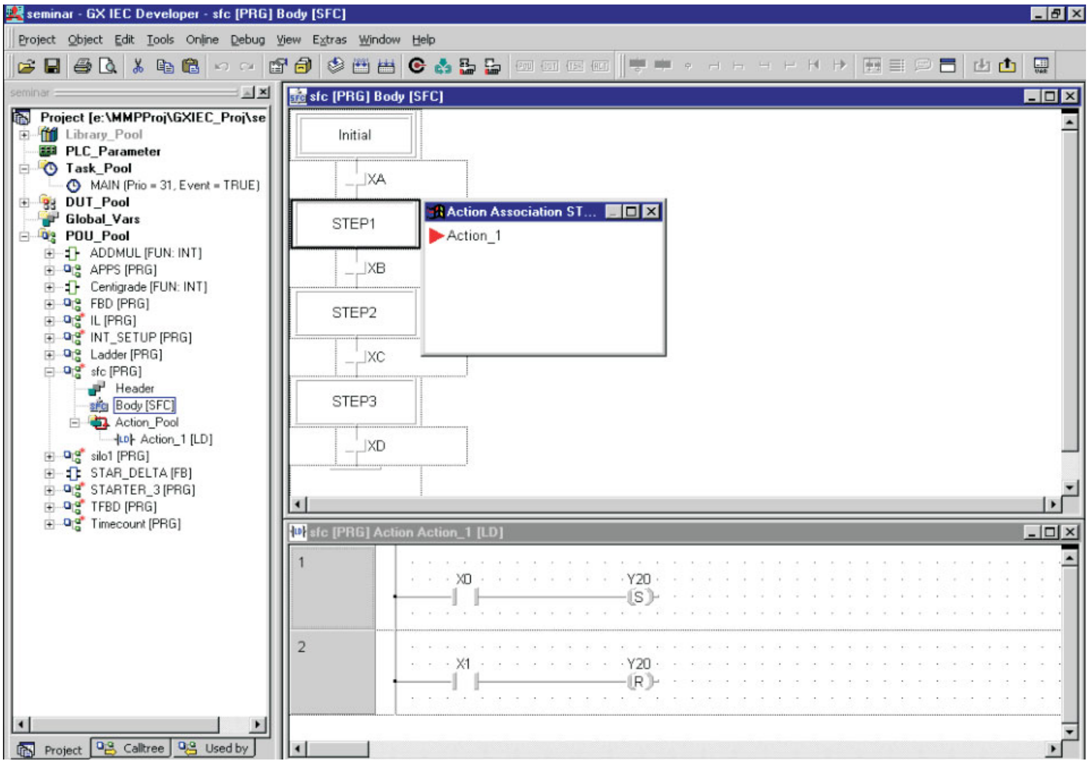


SFC Jump

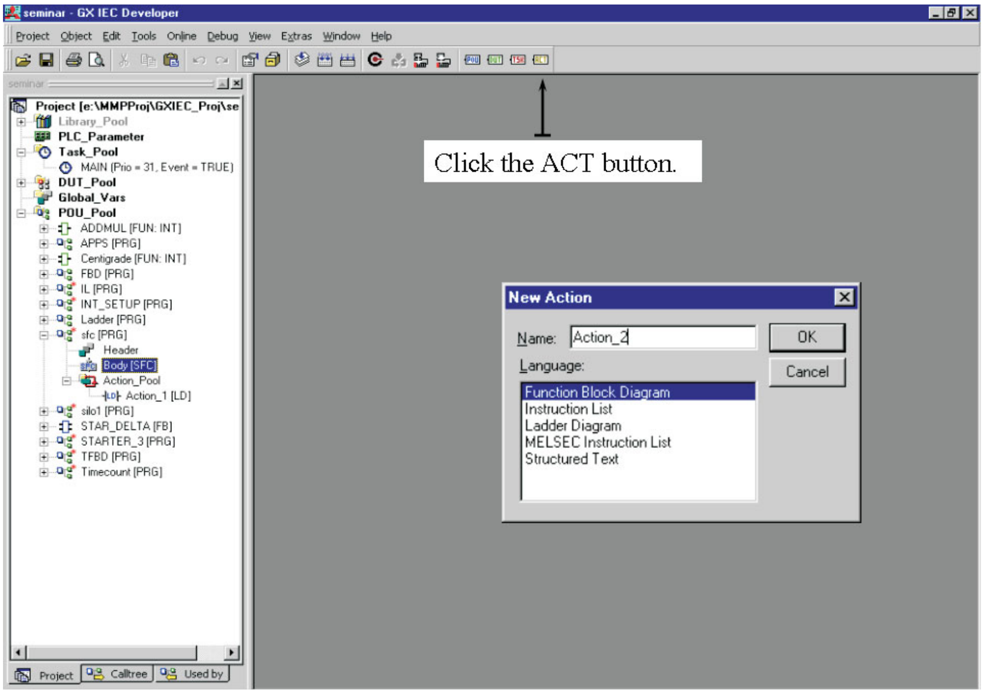


15.4 SFC Actions

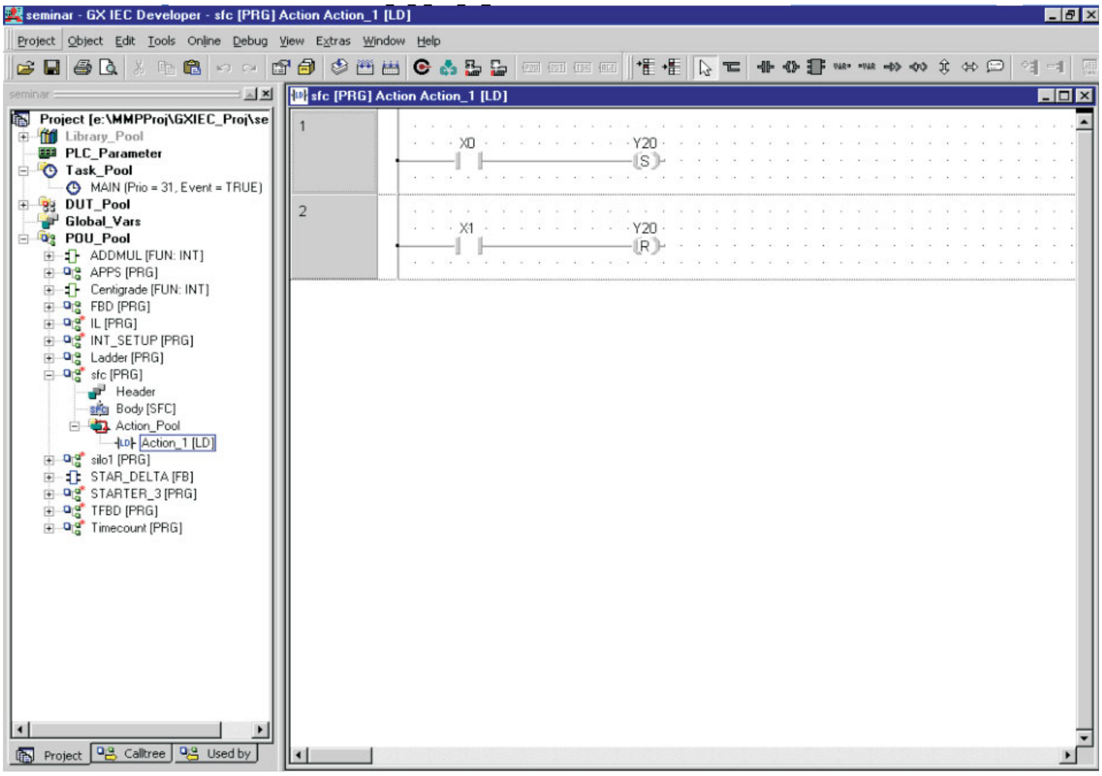
Each Step has associated Actions. An Action is simply a program, as for a POU. Each Action has associated logic written in either, IEC LD, IL, FBD or ST:



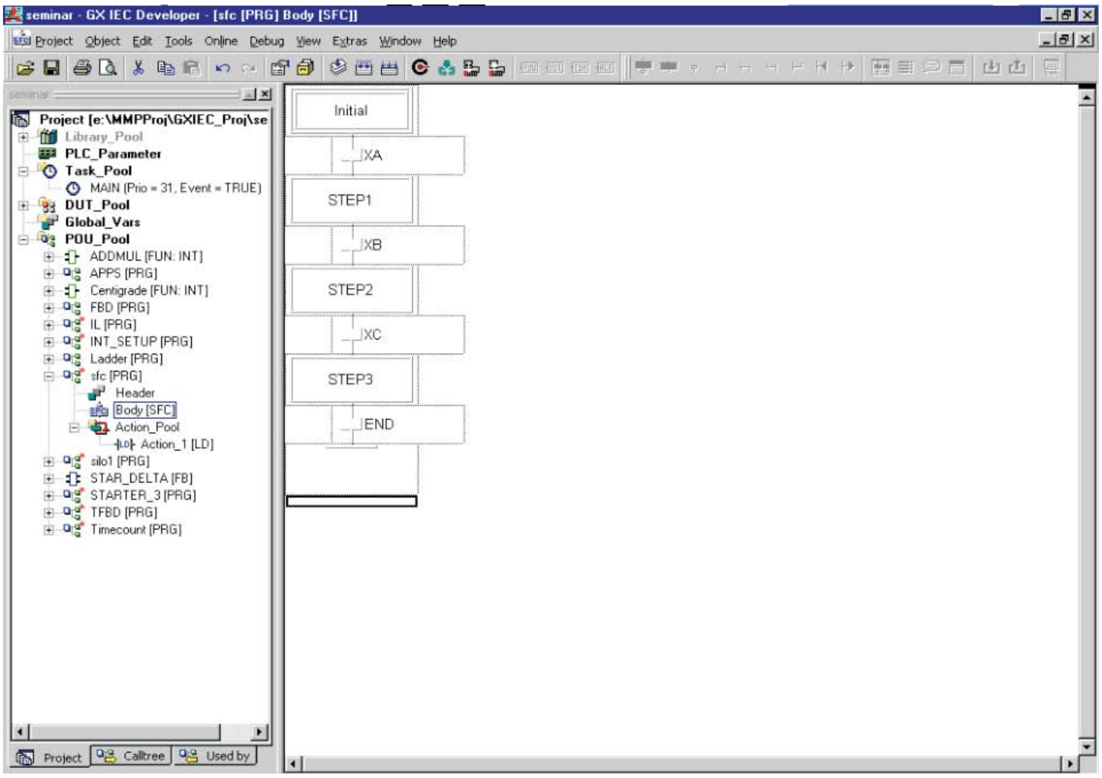
New Actions are created by clicking on the ***ACT*** button on the toolbar. Select the required editor, as for POUs:



Actions can be programs within their own right. Action_1 may be a complete ladder interlocking routine, consisting of many networks

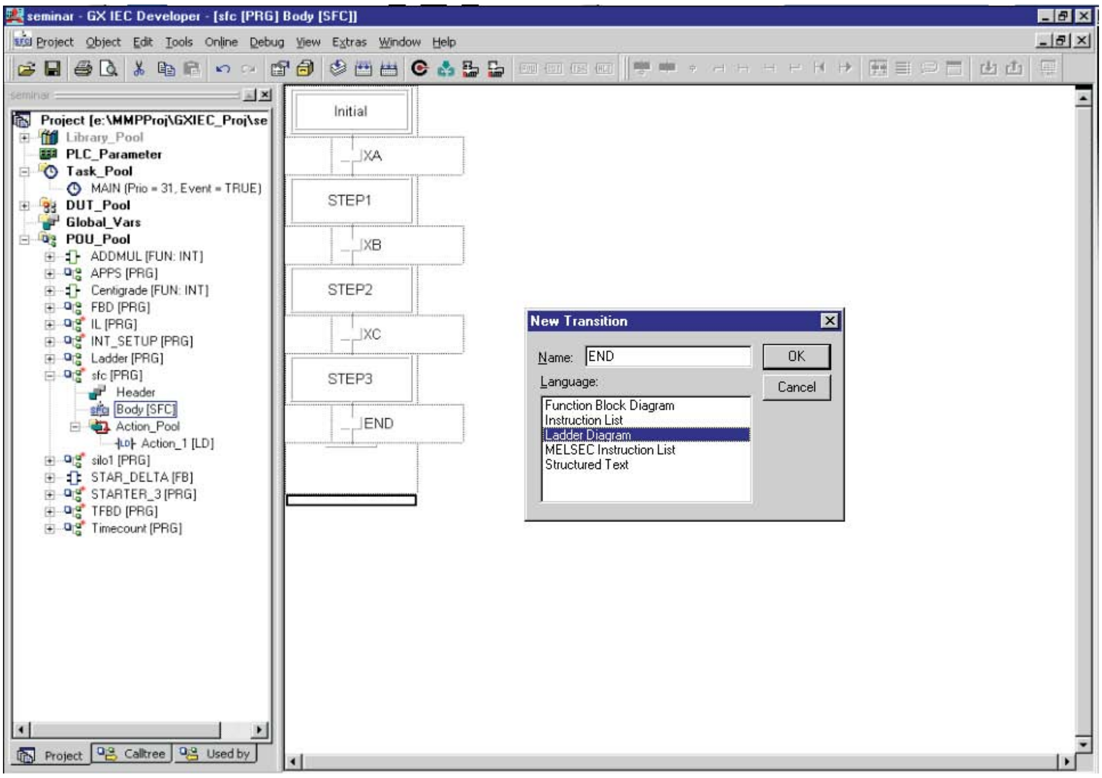


Each Transition can be a simple device i.e. Mitsubishi address XA, or an identifier name, or more complex, as a single network program written in either IEC, IL, LD or FBD:

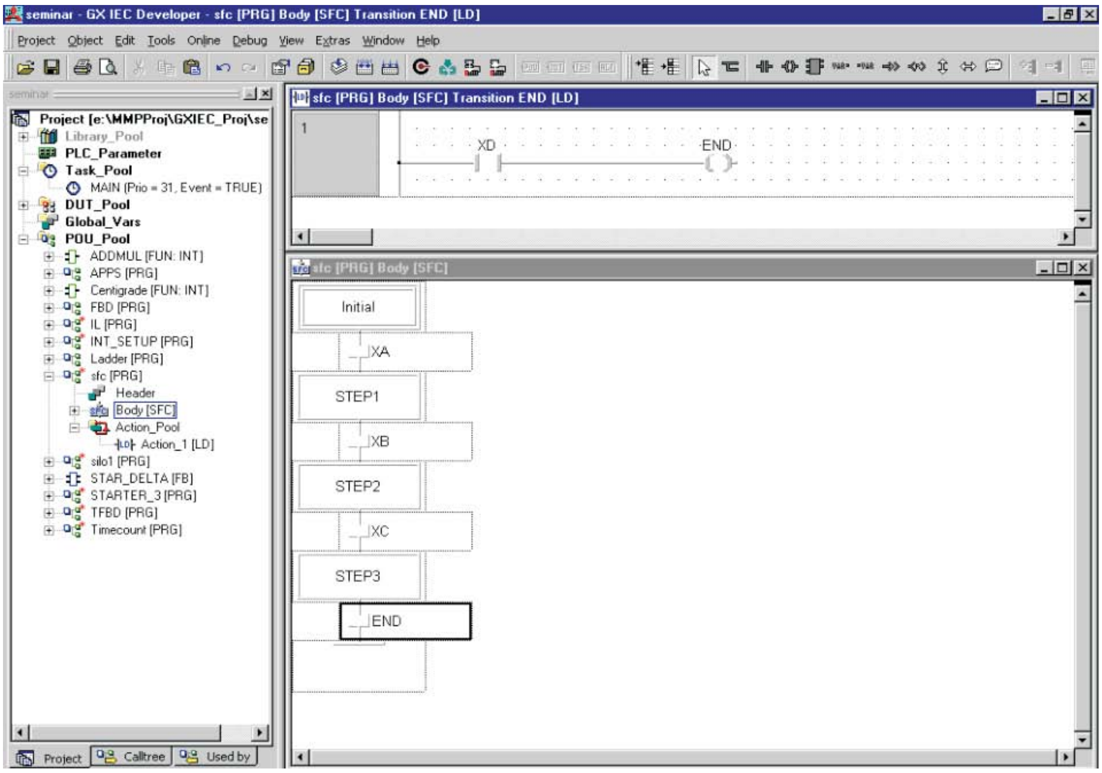


15.5 Complex Transitions

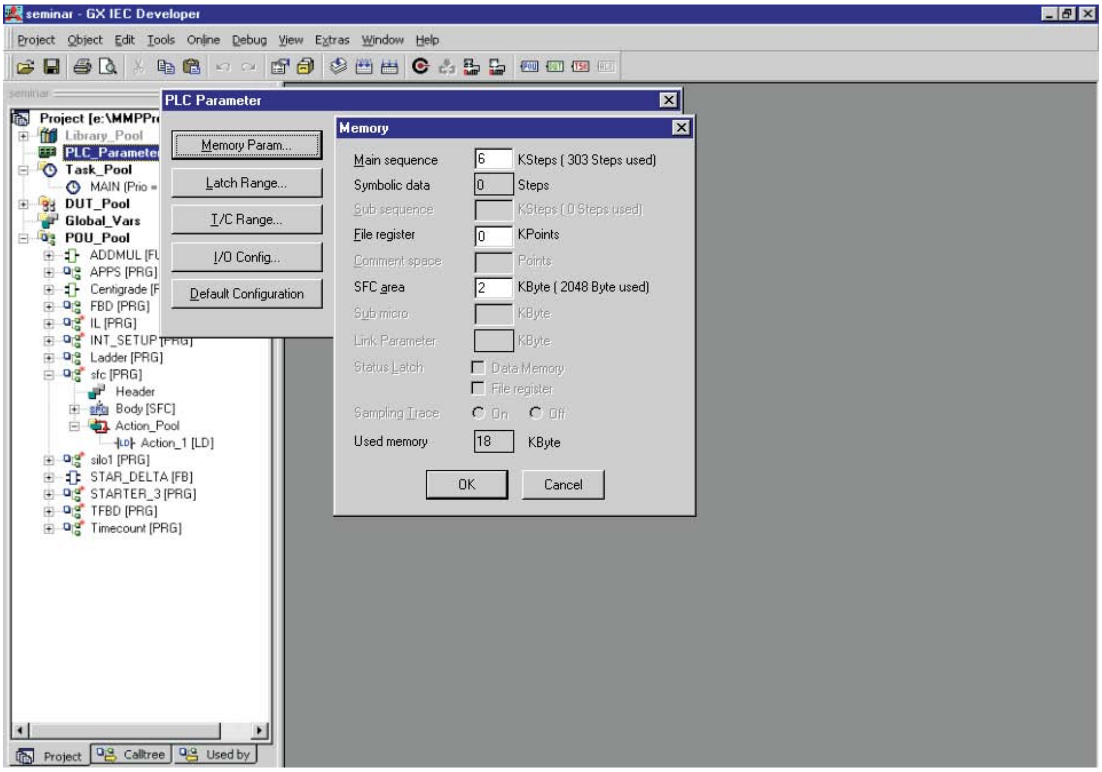
To program a complex transition, input a Transition name and hit the enter key. Choose the required editor, as for Actions:



The transition could be a complex expression but it only consists of one network:

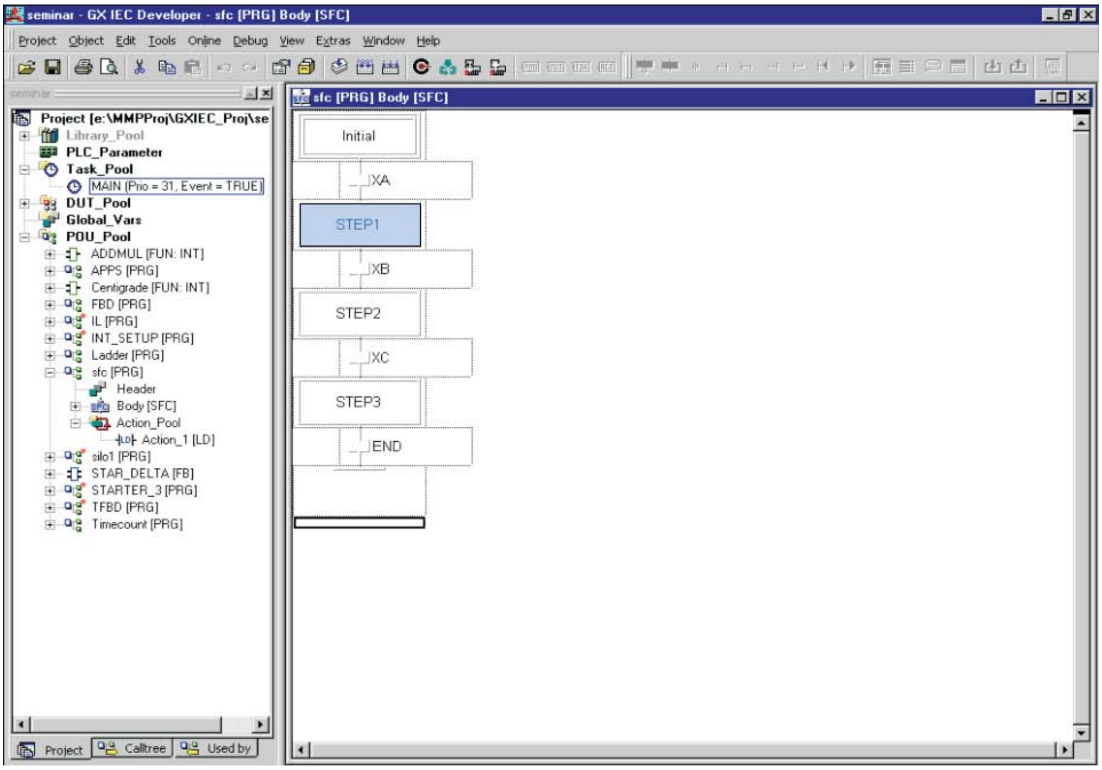


For A(ns) Series PLC's, SFC's reside in the micro computer area of the memory Cassette. This area must be allocated from PLC Parameters / Memory, as shown below:



This is not the case for Q series, as 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:



16 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.

16.1 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 *)
```

16.1.1 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
```

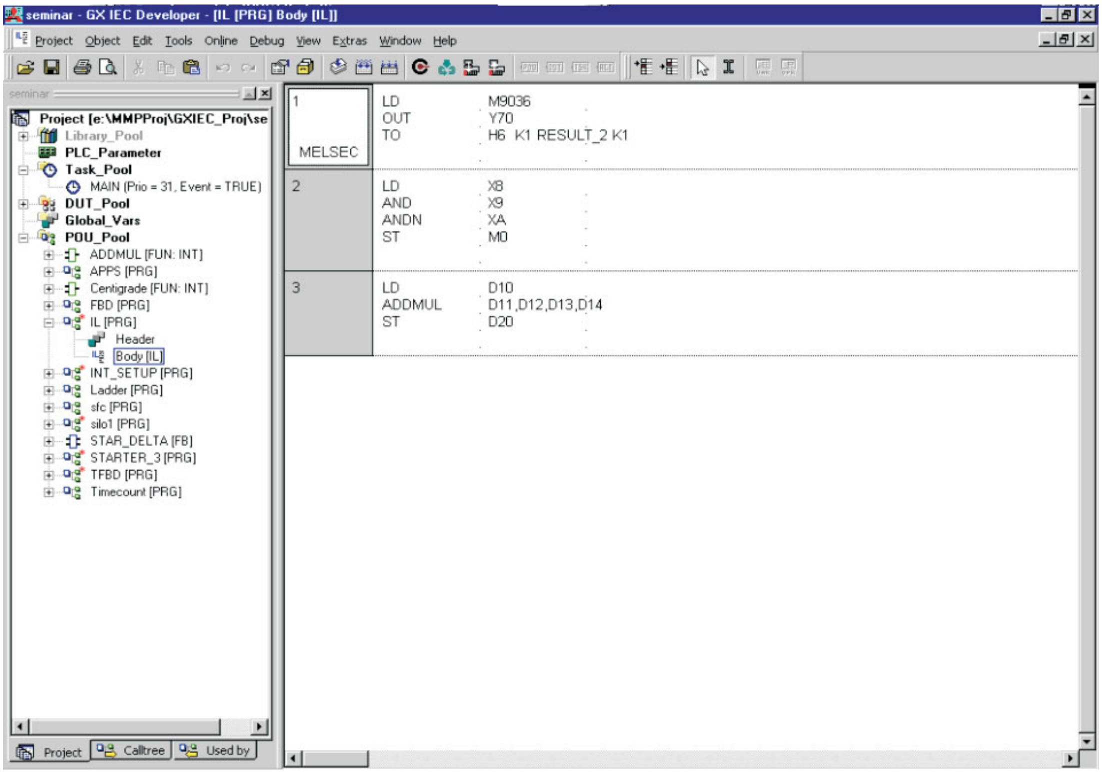
Use of an “_E” function can simplify still further. To Perform : “ + D0 D1 D2 ” and then “ + D2 K50 D3 ” from a conditional input X0 becomes:

```
LD      X0
ADD_E   D0,D1,D2,50,D3
```

This is because the ADD_E function has an Enable Output (ENO) feature.

16.2 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:



17 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.

17.1 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

17.2 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.

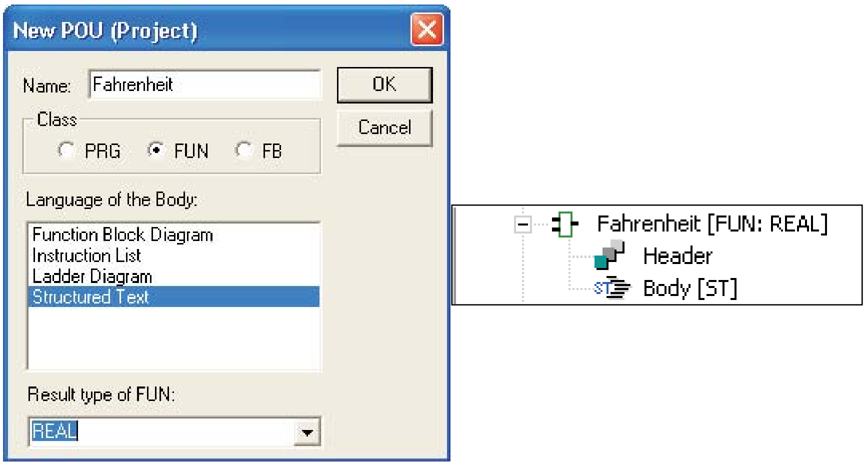
The Formula used is as follows:

$$Fahrenheit = \frac{Celsius \times 9}{5} + 32$$

The input and result variables will be in Floating Point (REAL) format.

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***):



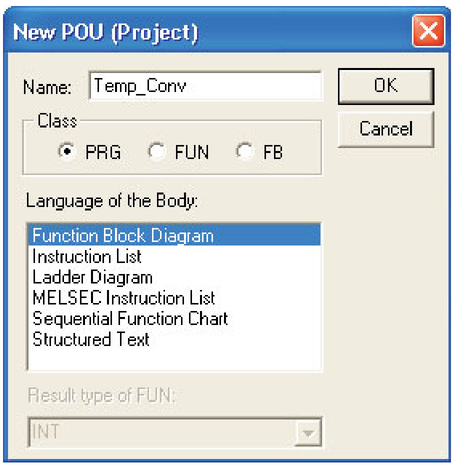
③ Create an entry in the header (LVL) of the Function “Fahrenheit”:

	Class	Identifier	Type	Initial	Comment
0	VAR_INPUT	Centigrade	REAL	0.0	

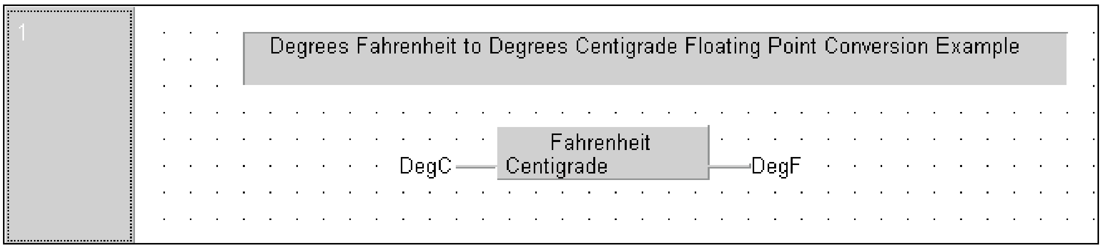
④ Open the Body of the Function “Fahrenheit” and enter the following simple ST program:

Fahrenheit := (Centigrade*9.0/5.0+32.0);

⑤ Create a new POU with a name “Temp_Conv”, Class: ***PRG***, Language: ***Function Block Diagram***



⑥ Open the body of the program POU “Temp_Conv” and enter the following program example:



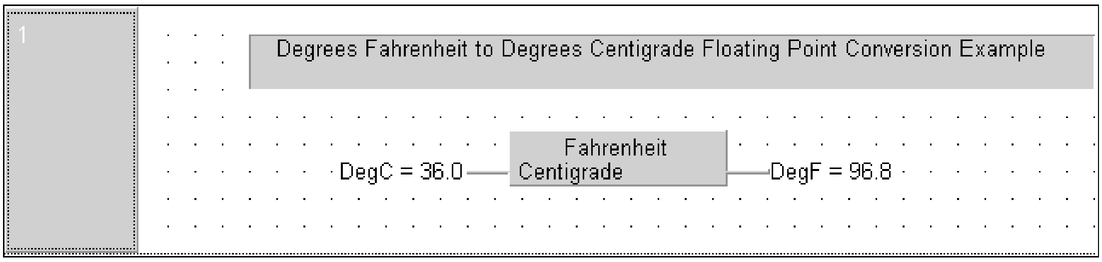
⑦ Edit the LVL (Header) of the POU “Temp_Conv” to include 2 local variables as shown below:

	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.



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.

18 Ethernet Communications

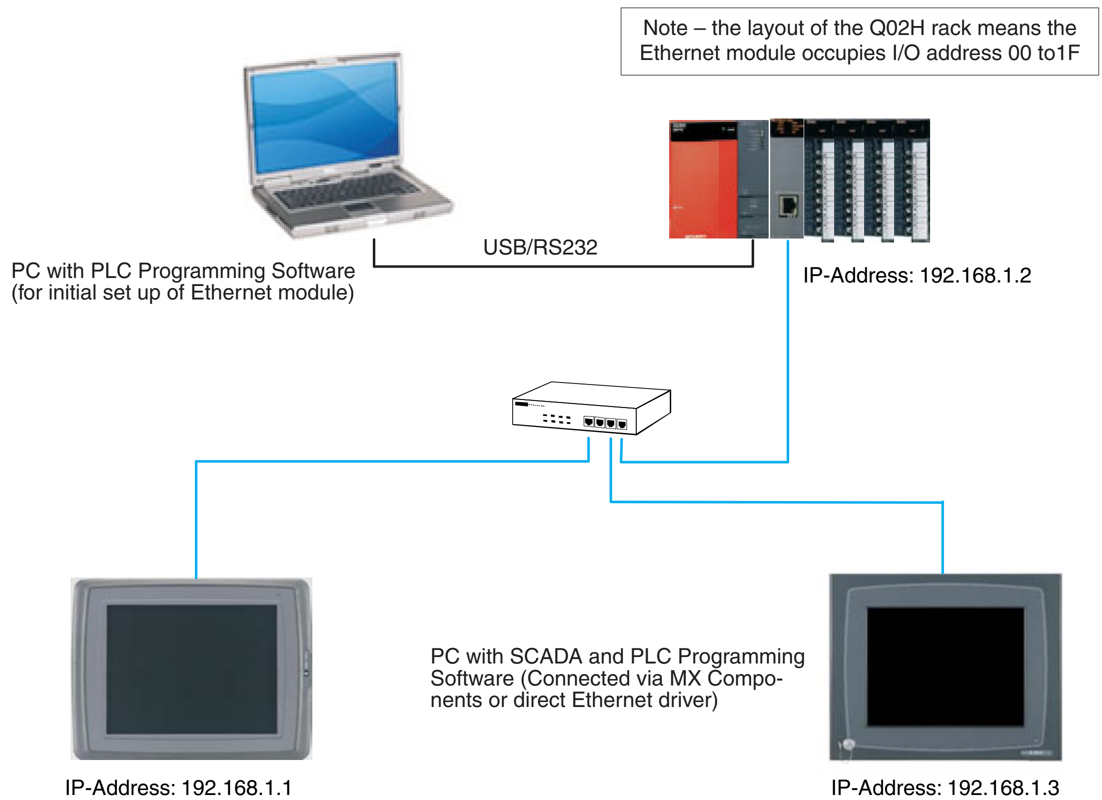
18.1 Configuring Qn Ethernet Module by Parameter

This section provides a step-by-step guide to setting up a QJ71E71 type Ethernet module (to be referred to as ‘module’ from now on) by parameter setting, GX IEC Developer 7.00 or later.

As an example, this section will show how to set up a module for allowing TCP/IP communications between a Q02H, a SCADA PC and an E1071 HMI. Also shown is how the programming software can be configured to communicate with the Q02H via Ethernet once the settings have been made.

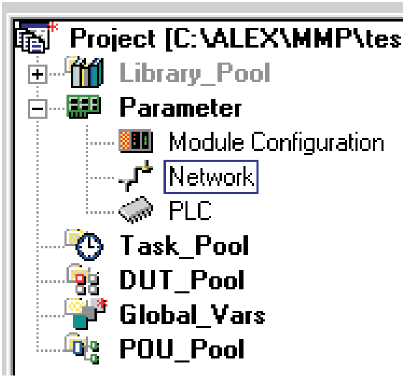
The diagram below shows the layout of the example Ethernet network. Proposed IP addresses are shown next to the Ethernet nodes.

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.

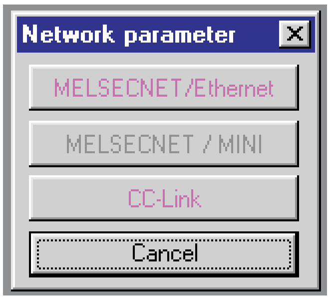


18.1.1 Configuring the PLC (using initial set up PC)

① Using the programming software, call up the ***Network*** Parameter selection box by double clicking on the option highlighted by the arrow.

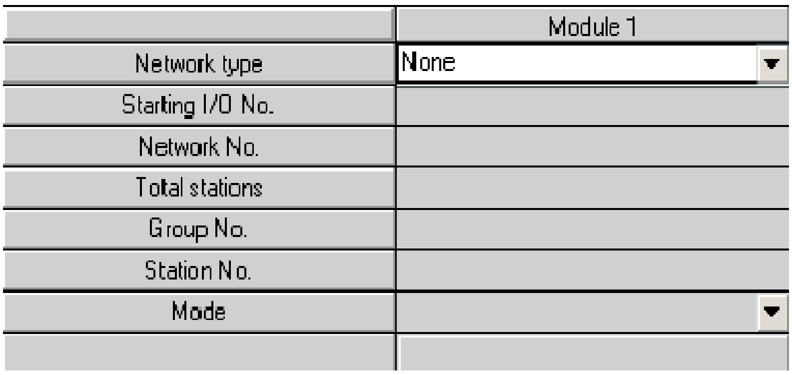


② When the box has been opened, select ***MELSECNET/Ethernet*** as shown below.

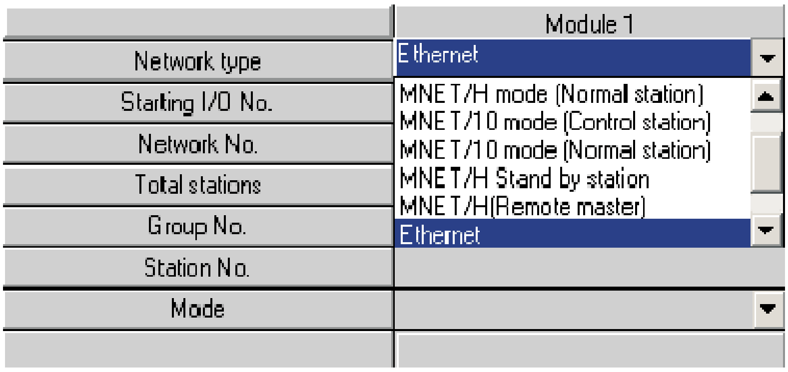


This opens up the dialogue box to allow the Ethernet module to be configured which can be seen below.

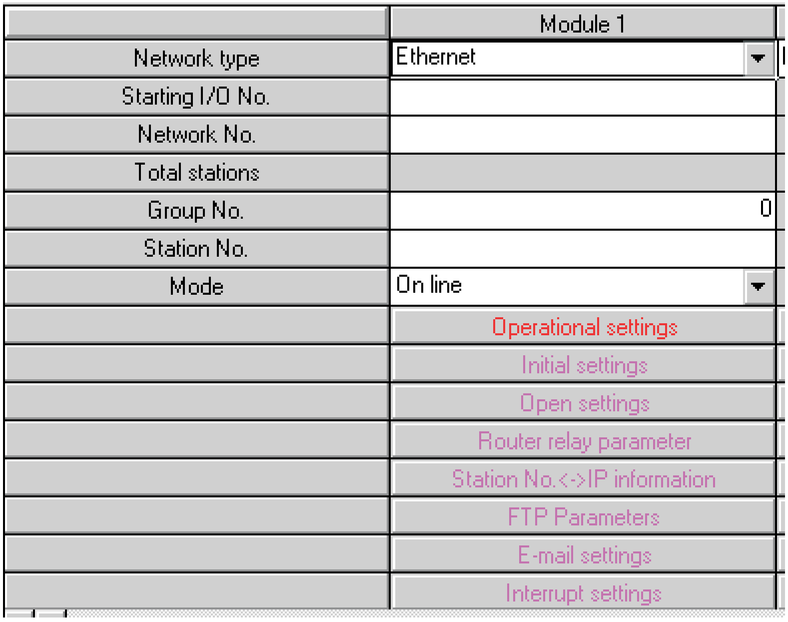
③ In the Network type window, click on the down arrow, to show the available selections:



④ Ethernet is the final option in the list. Select it as shown below:



⑤ The dialogue box now shows the specific setting options for the module. The buttons in the bottom half of the table that are in red are for setting the mandatory parts of the module, those in magenta are optional, and are set as required.



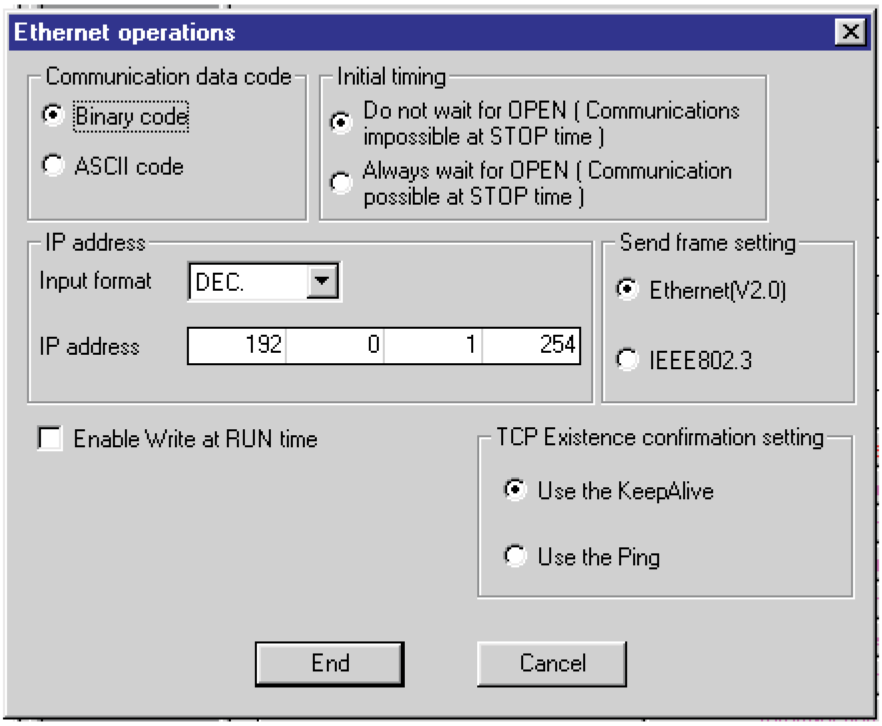
⑥ Click in the boxes in the top half of the table and enter the values as required. The table below shows the settings for the Q02H in the example system described earlier.

	Module 1	
Network type	Ethernet	
Starting I/O No.	0000	
Network No.	1	<— see Note below
Total stations		
Group No.	0	
Station No.	2	<— see Note below
Mode	On line	
	Operational settings	
	Initial settings	
	Open settings	
	Router relay parameter	
	Station No.<->IP information	
	FTP Parameters	
	E-mail settings	
	Interrupt settings	

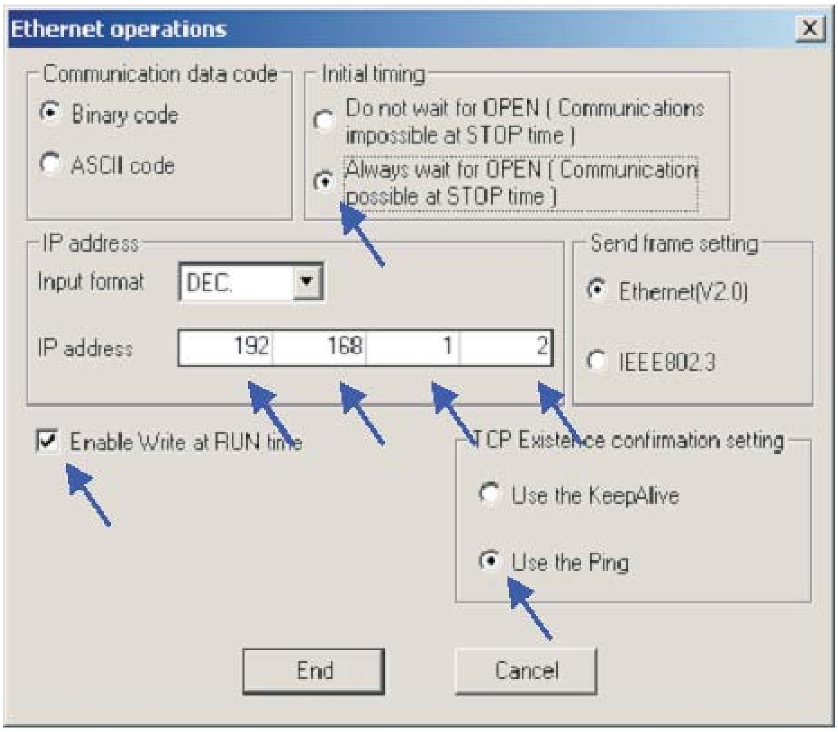
NOTE

The “network number” and “station number” settings are used to identify the module when Qn PLC’s use the Ethernet for Peer-to-Peer communications (not covered in this document). These settings are also used when the programming software is to communicate to the Qn PLC across the Ethernet network. This subject is covered later in the document.

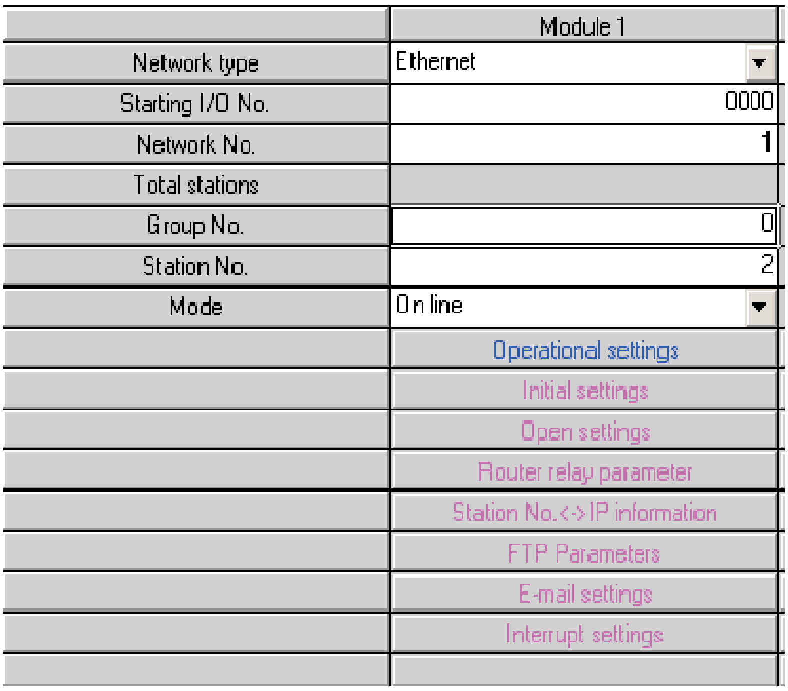
⑦ Next, click on the ***Operational settings*** to bring up the dialogue shown below. The settings already there are the defaults that the programming software applies.



⑧ The dialogue below shows the settings required for the example system described earlier. The arrows highlight the differences for clarity.



⑨ After the settings here are made, click ***End*** to return to the main network parameter setting window. Note that the ***Operational settings*** button has now changed to blue, indicating that changes have been made.



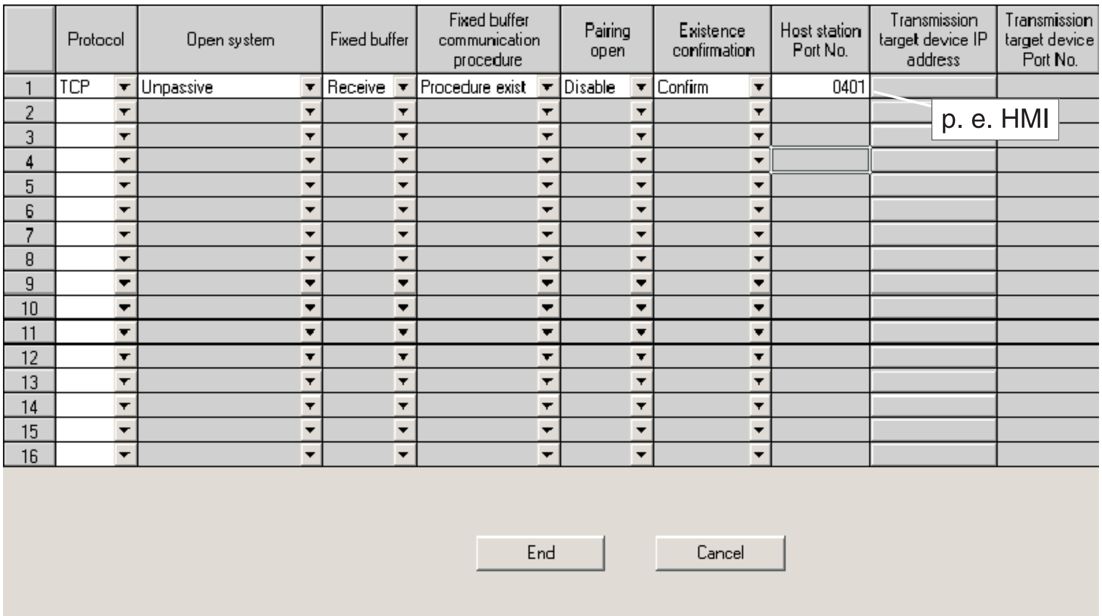
⑩ Next, click on ***Open settings*** to bring up the following dialogue. This is where the settings for the Scada and HMI will be made.

NOTE

There is no need to set anything here, if the Ethernet card is **only** to be used for program monitor/edit using the programming software (as described later).

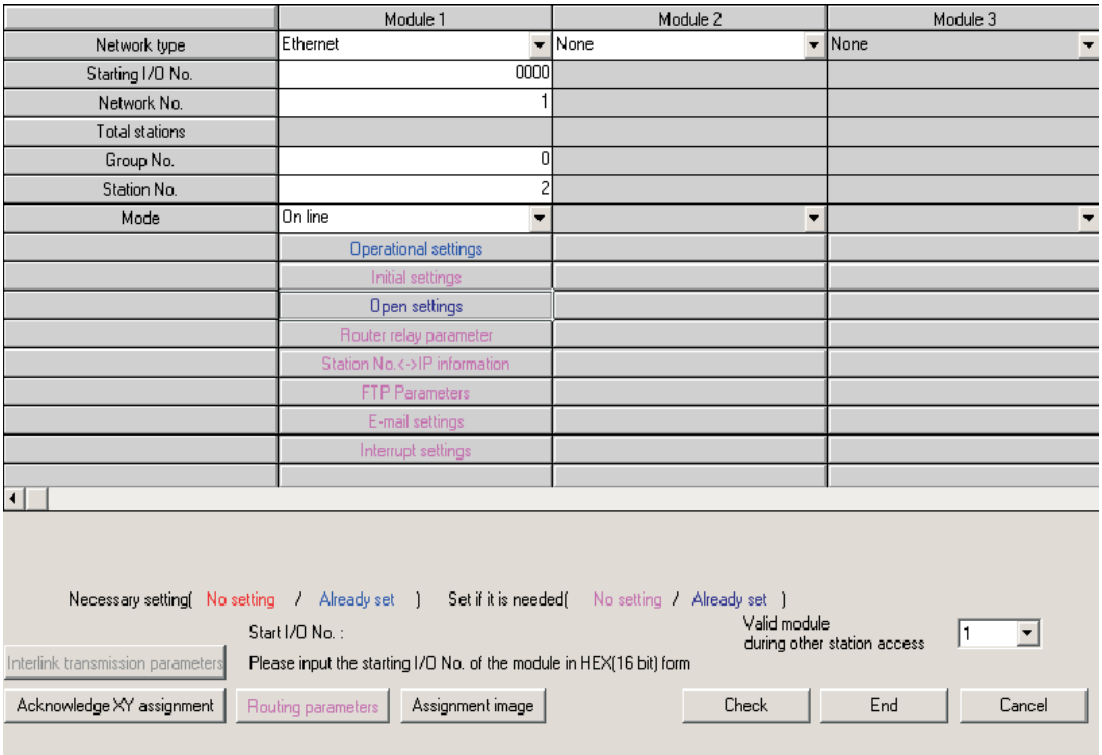
	Protocol	Open system	Fixed buffer	Fixed buffer communication procedure	Pairing open	Existence confirmation	Host station Port No.	Transmission target device IP address	Transmission target device Port No.
1									
2									
3									
4									
5									
6									
7									
8									
9									
10									
11									
12									
13									
14									
15									
16									

The dialogue below shows the settings required for communication with both the Scada and the HMI, for the example system described earlier. The settings are made by selecting the required options from the drop-down lists in each window, or typing as required.



	Protocol	Open system	Fixed buffer	Fixed buffer communication procedure	Pairing open	Existence confirmation	Host station Port No.	Transmission target device IP address	Transmission target device Port No.
1	TCP	Unpassive	Receive	Procedure exist	Disable	Confirm	0401		
2									
3									
4									
5									
6									
7									
8									
9									
10									
11									
12									
13									
14									
15									
16									

⑪ When the settings have been made, click ***End*** to return to the main network parameter setting window.

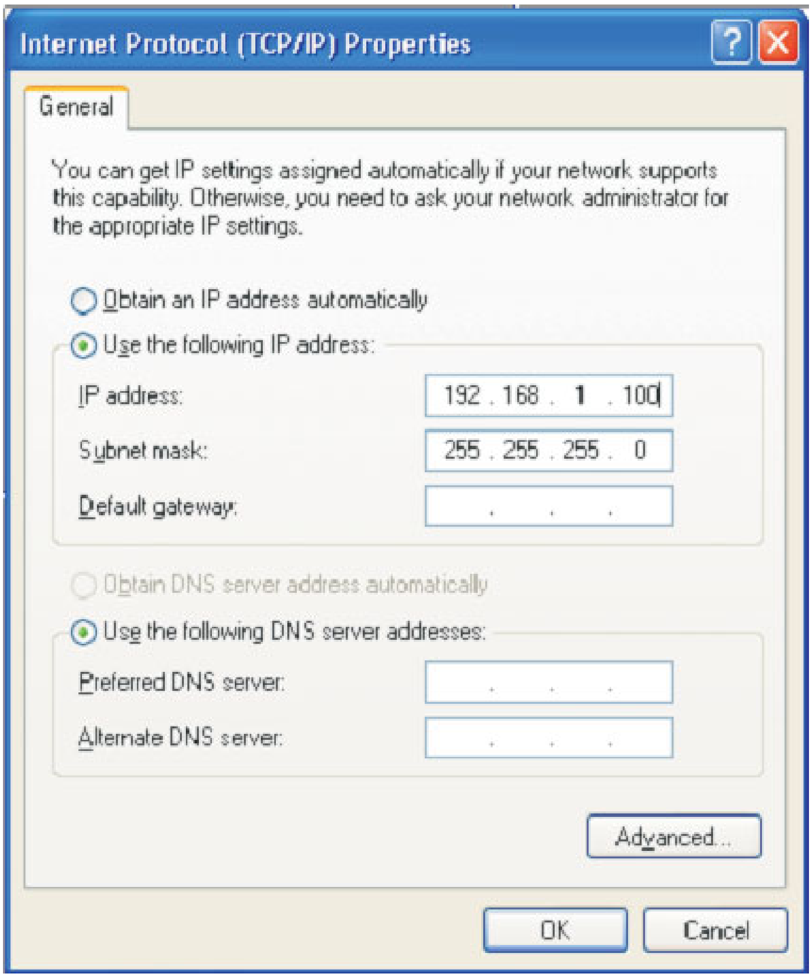


No more setting is required here for communications with the Scada or the HMI.

⑫ Click ***End*** to check and close the main network parameter setting dialogue. These settings will be sent to the PLC next time the parameters are downloaded.

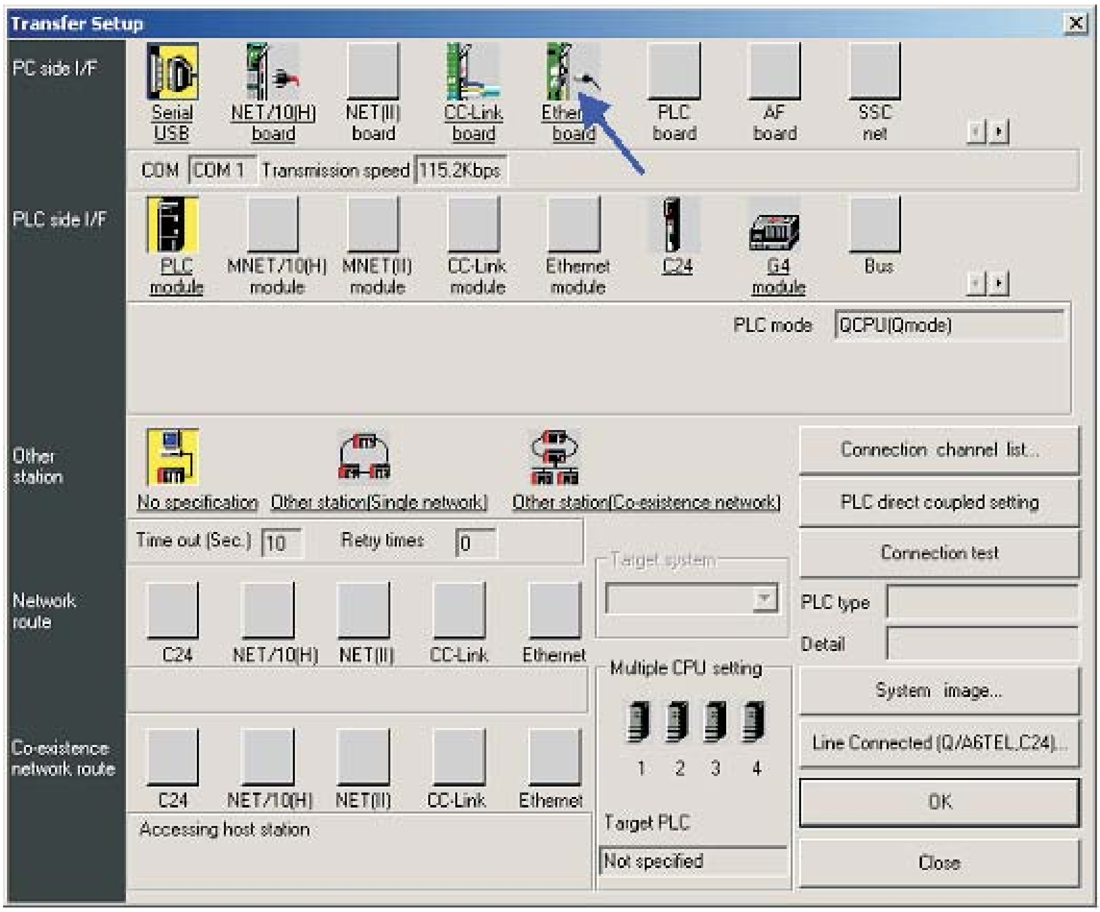
18.2 Configuring the PC on the Ethernet

① 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. Please note that after changing IP address, the PC may require a restart.



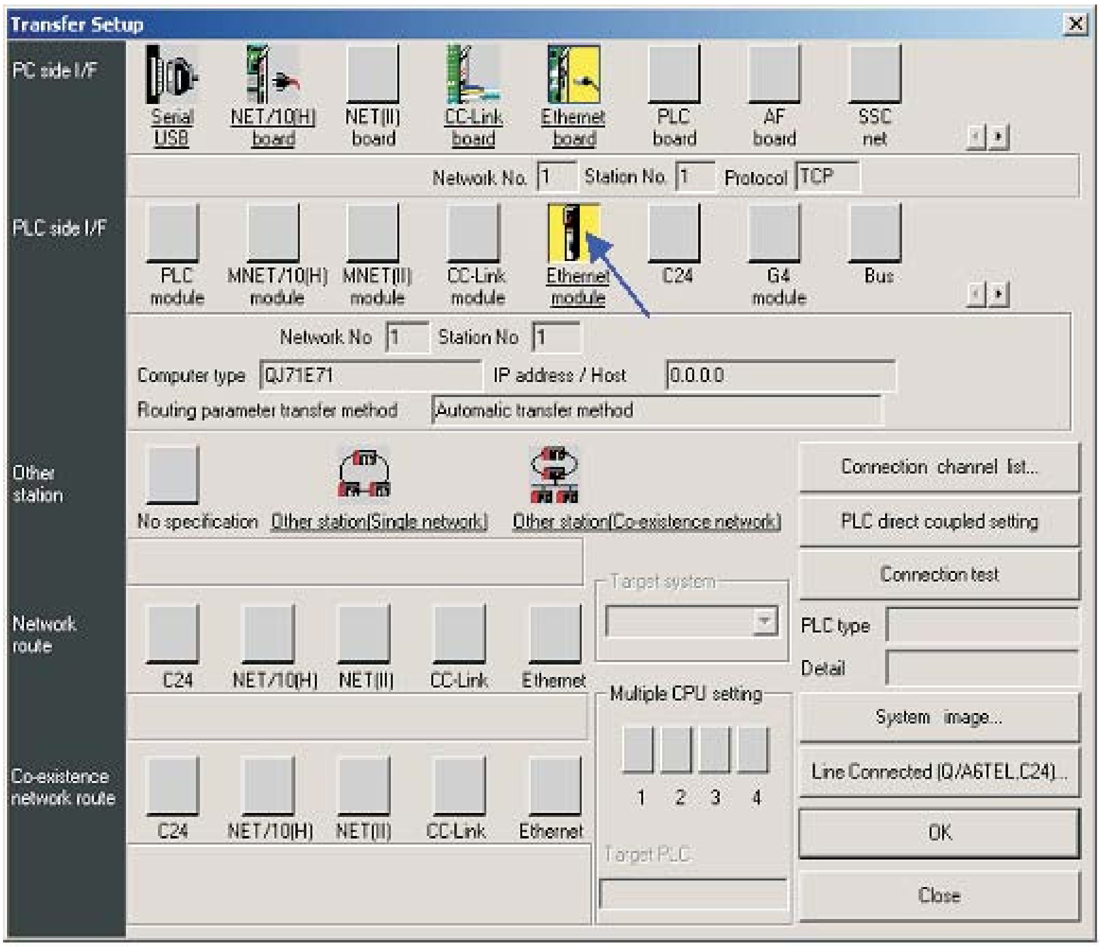
18.3 Configuring GX-IEC Developer to access the PLC on Ethernet

① Open the connection settings dialogue as shown



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).

③ The ***PC Side I/F*** should default to Network No. = 1, Station No = 1 and Protocol = TCP as shown below. If it does NOT show this, then double click on ***Ethernet board*** and make these settings in the appropriate places

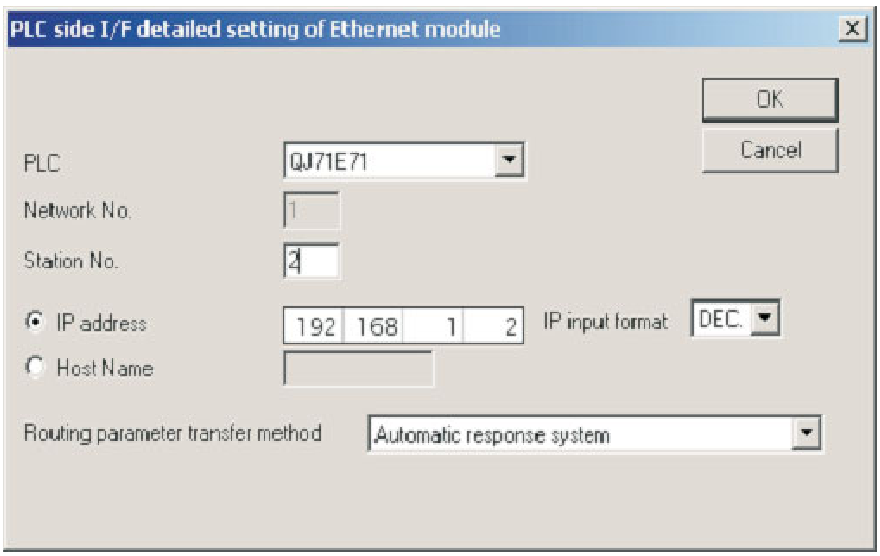


④ Next, double click on ***Ethernet module*** under ***PLC side I/F*** as shown above. This will open up the dialogue to allow the selection of the PLC to be communicated with over the Ethernet. Enter the settings shown, as these were the settings put into the PLC earlier. (refer back to parts 6 and 7 in section 18.1.1)

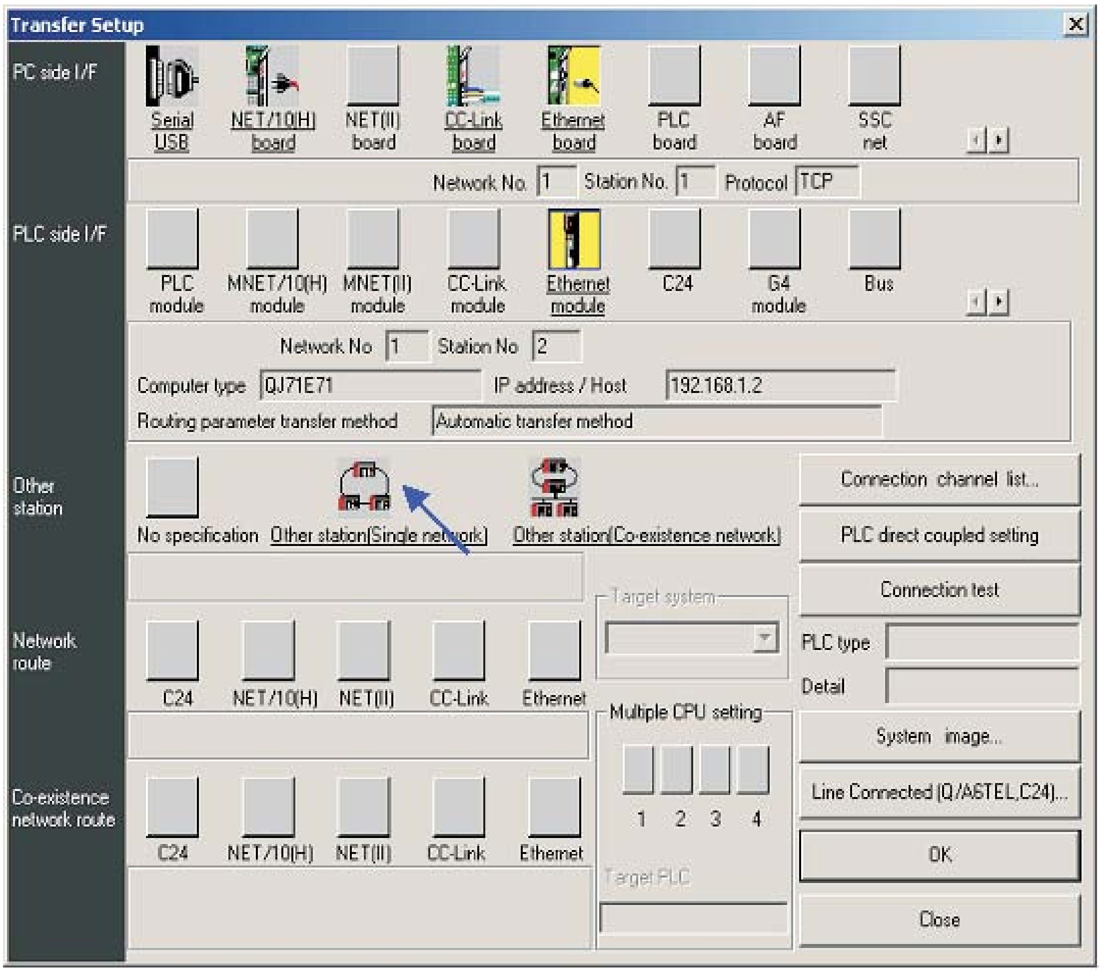
⑤ Click ***OK*** when done.

NOTE

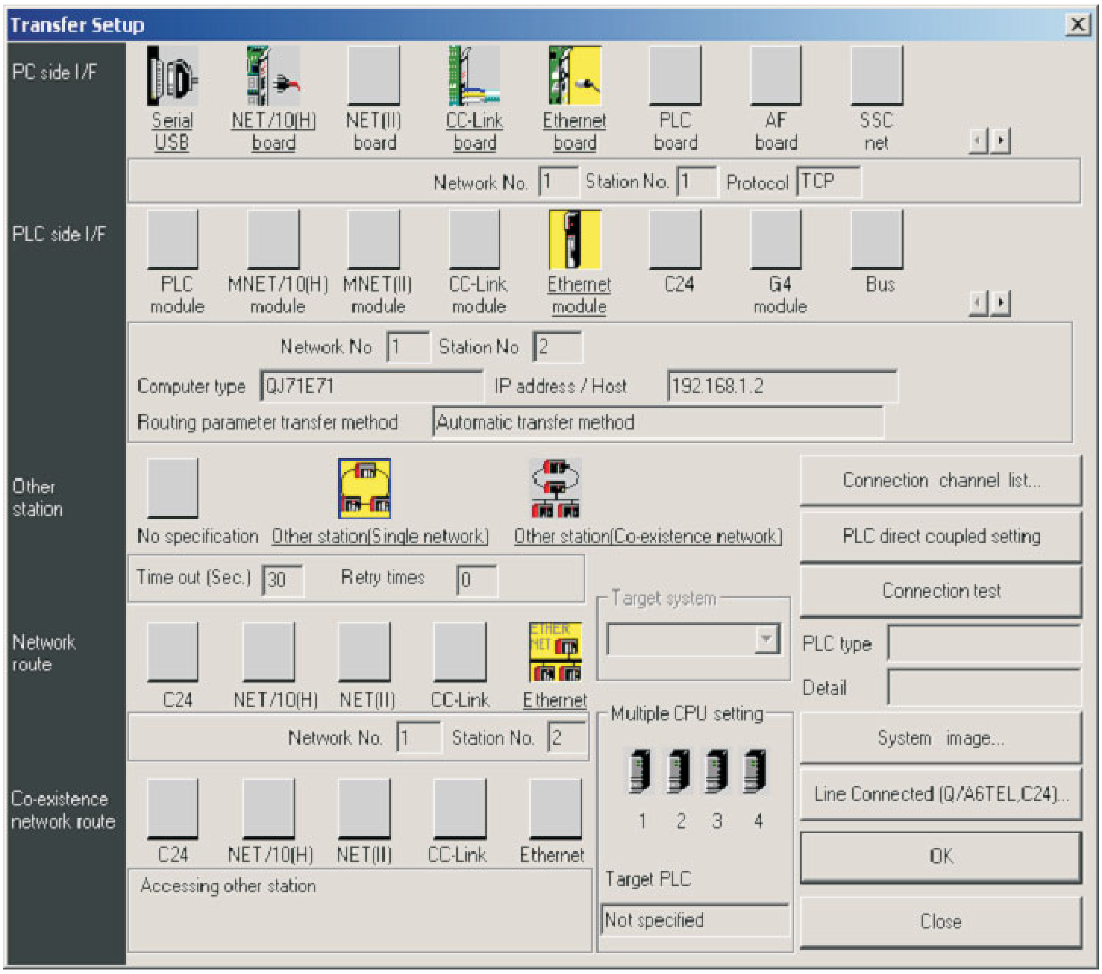
There is no need to specify a port number, as the programming software will use a MELSOFT Protocol dedicated port by default.



⑥ Next, single click on ***Other station (Single network)*** as shown below.

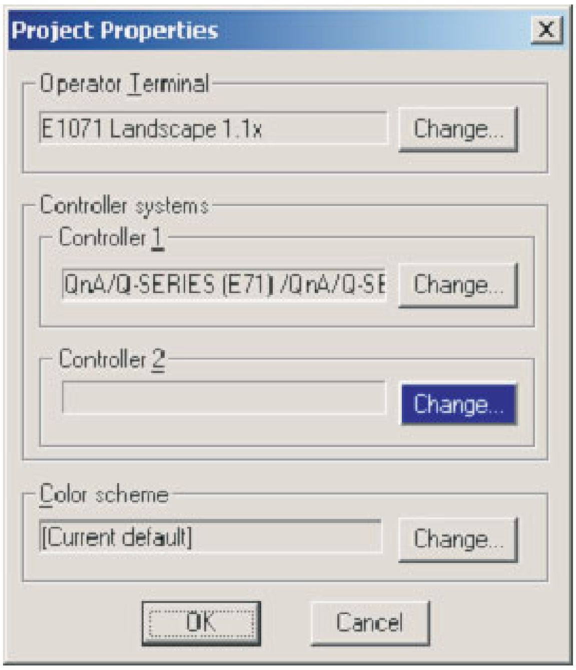


⑦ This will complete the setting, making the dialogue look as shown below. Click ***Connection test*** to confirm the settings are correct. Then click ***OK*** when finished.

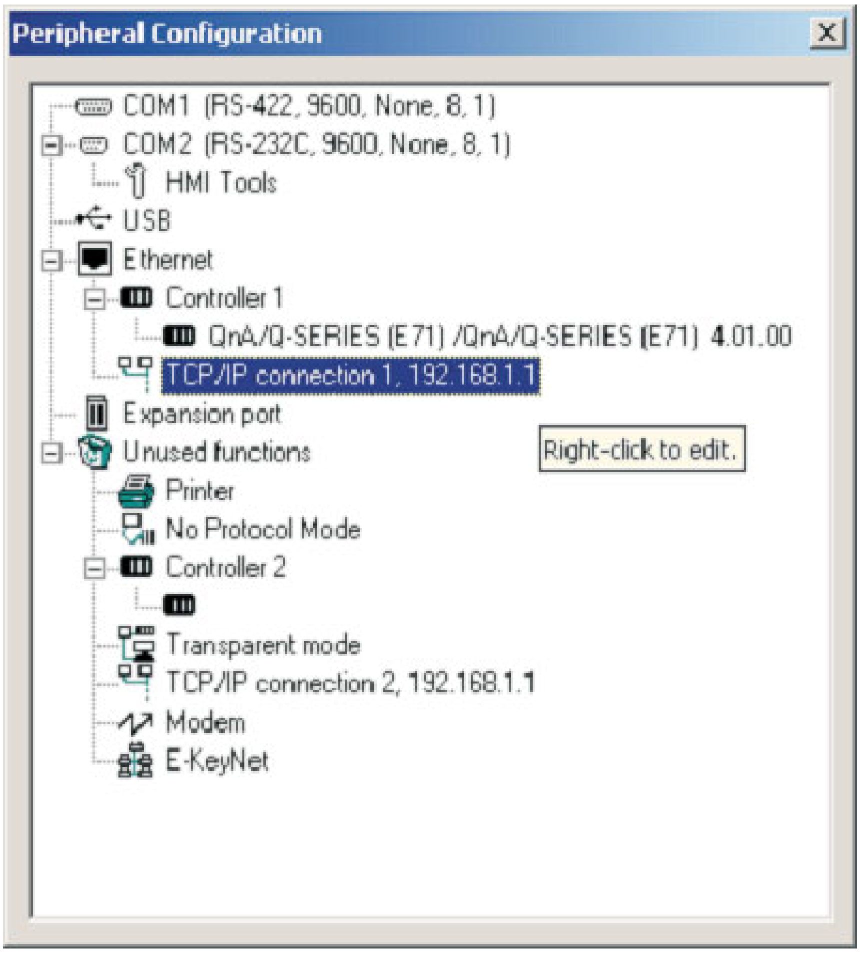


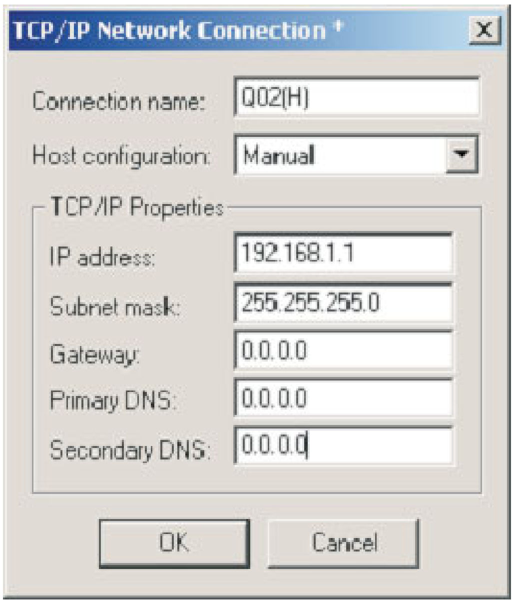
18.4 Setting up the HMI

① The E-Designer project for the example system needs to have the following settings.

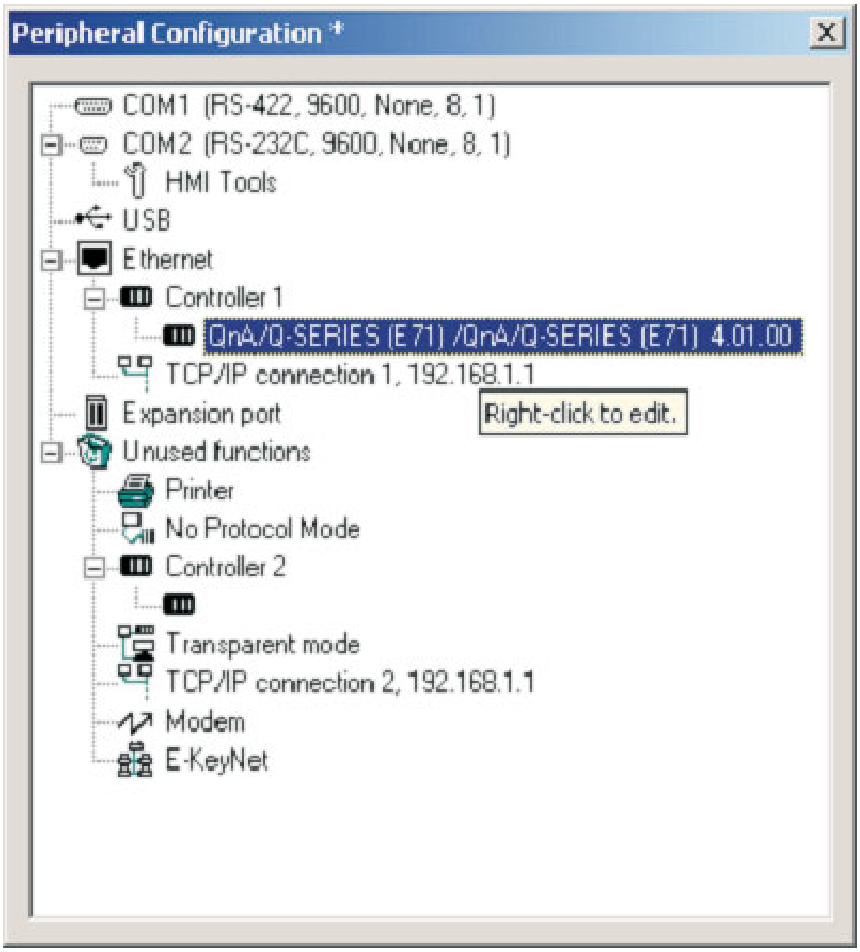


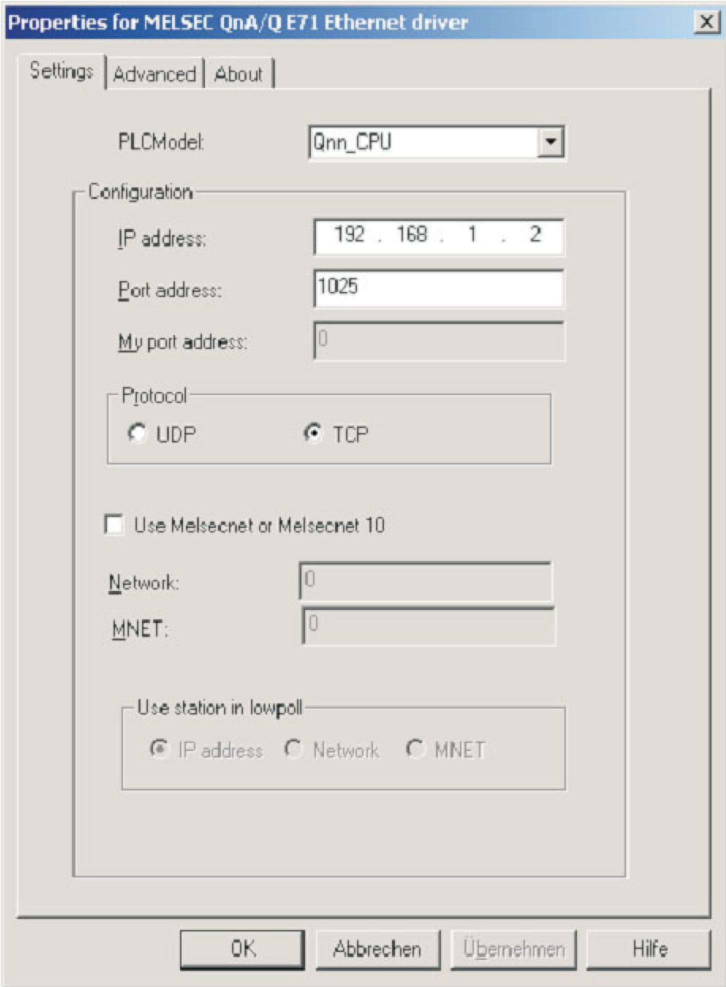
② Next, open up the ***Peripherals*** options under the System menu, and configure the HMI's TCP/IP connection as shown:





③ Then make the following settings for Controller 1 (i.e. the target PLC), according to the settings made in the PLC earlier.





As with the MQE settings earlier, note that E71 port number 1025, decimal 1025 is equal to hex 401 (set in the PLC Local station port number – refer back to part 10 of section 18.1.1).

④ Click ***OK***, exit the Peripheral settings and download these settings with the project.

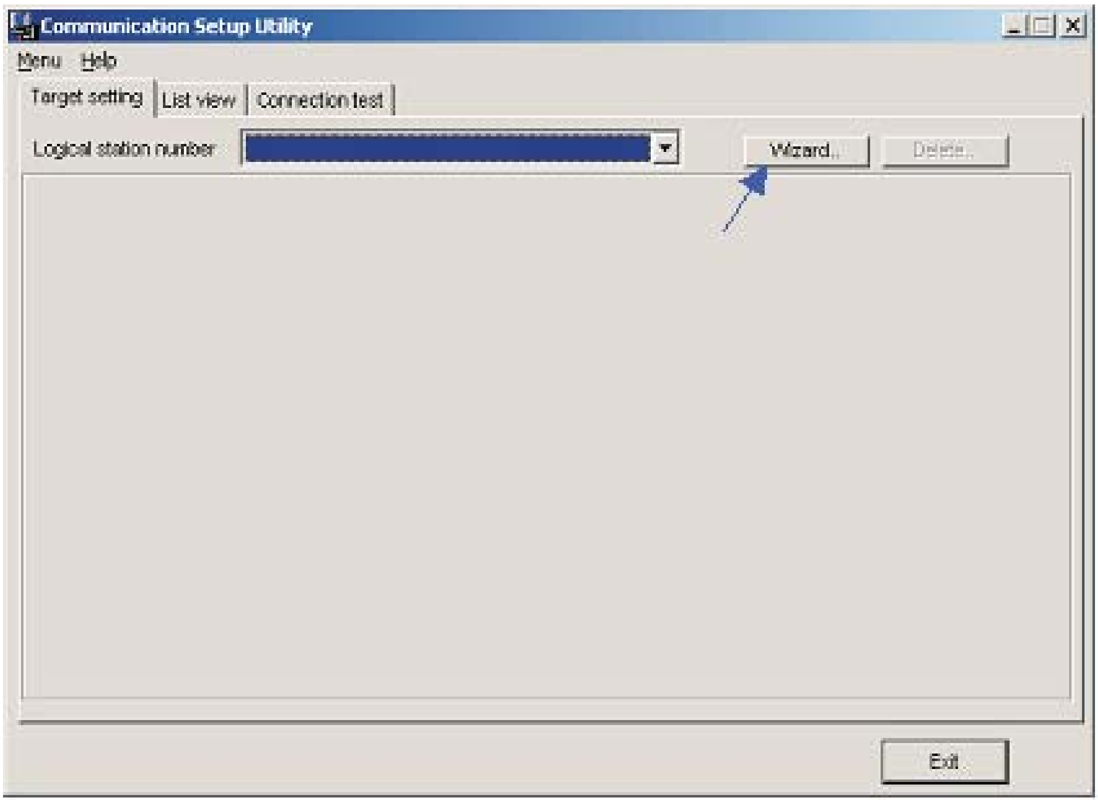
18.5 Communication via MX Component

MX Component is a tool designed to implement communication from PC to the PLC without any knowledge of communication protocols and modules.

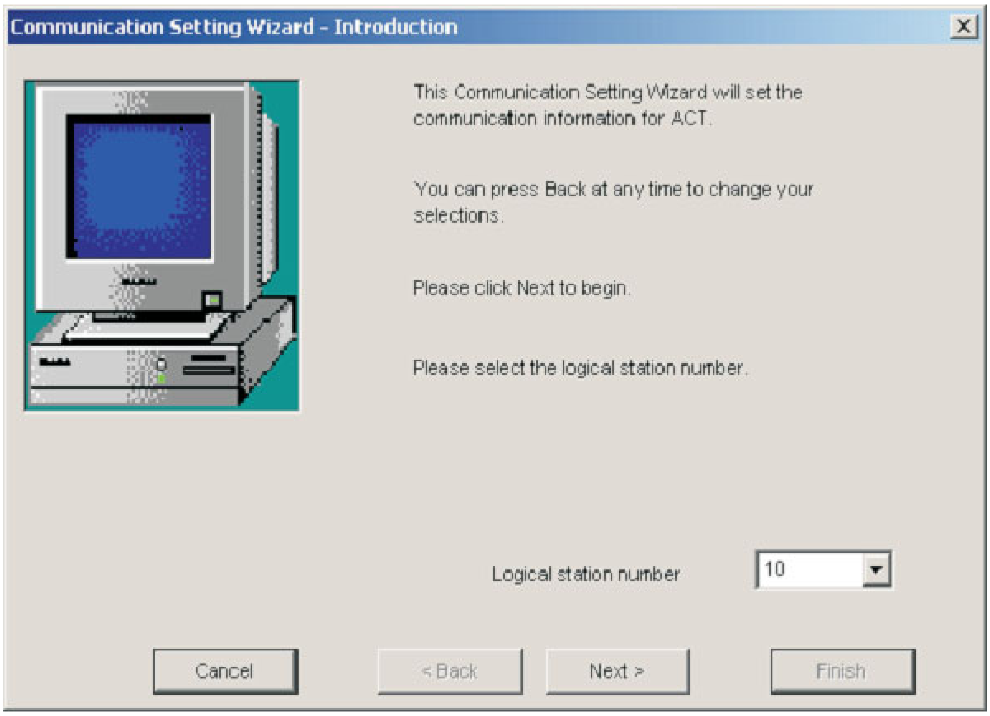
It supports serial CPU port connection, serial computer links (RS232C, RS422), Ethernet, CC-Link and MELSEC networks.

The figure below shows the easy way for creating of communication between a PC and a PLC via MX Component.

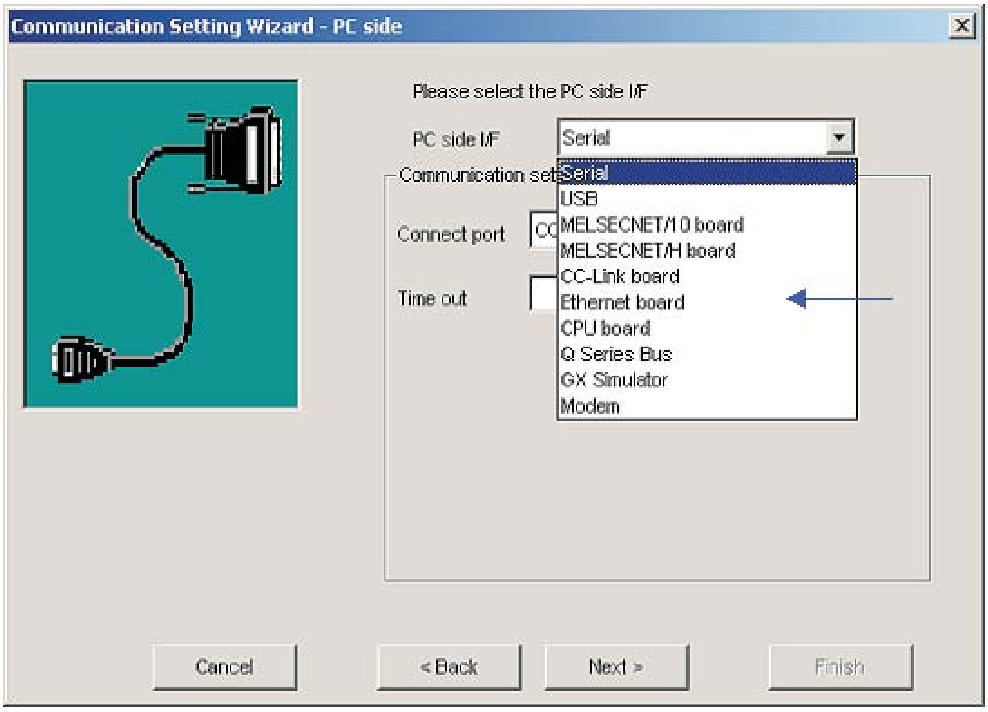
① Start the ***Communication Setting Utility*** and select the ***Wizard***



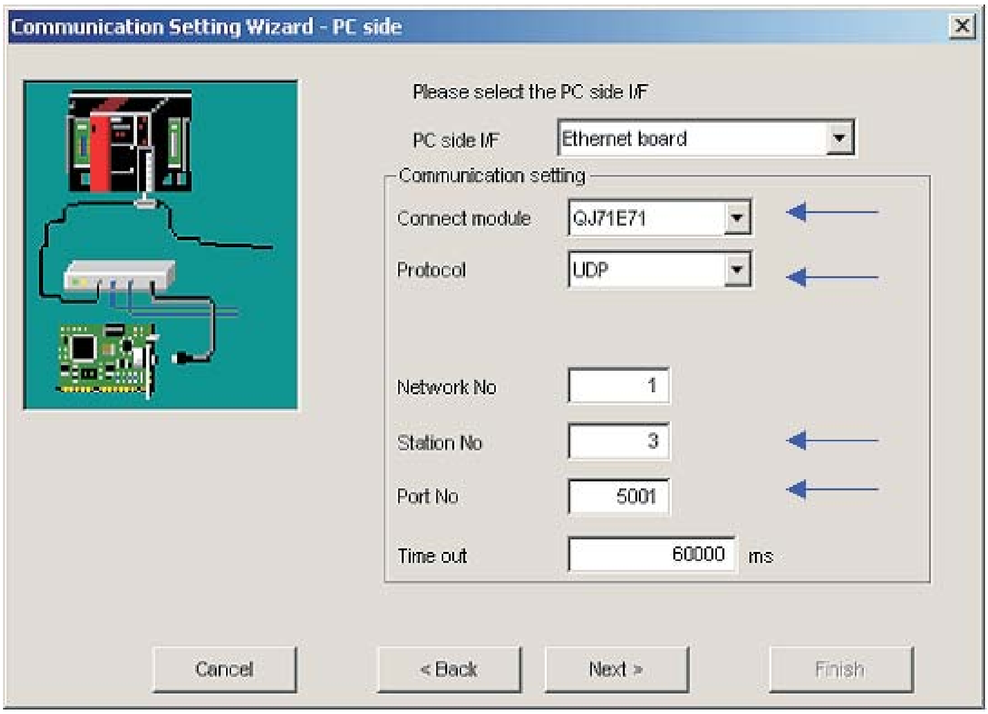
② First you must define the ***Logical station number***



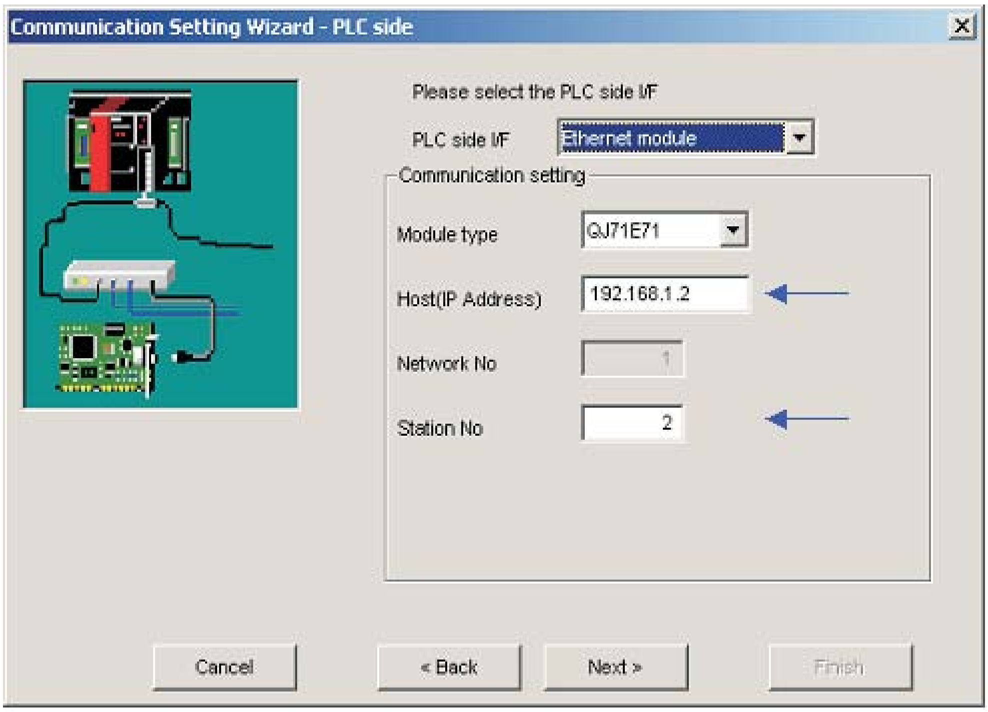
③ Next, configure the ***Communication Settings*** on the PC side



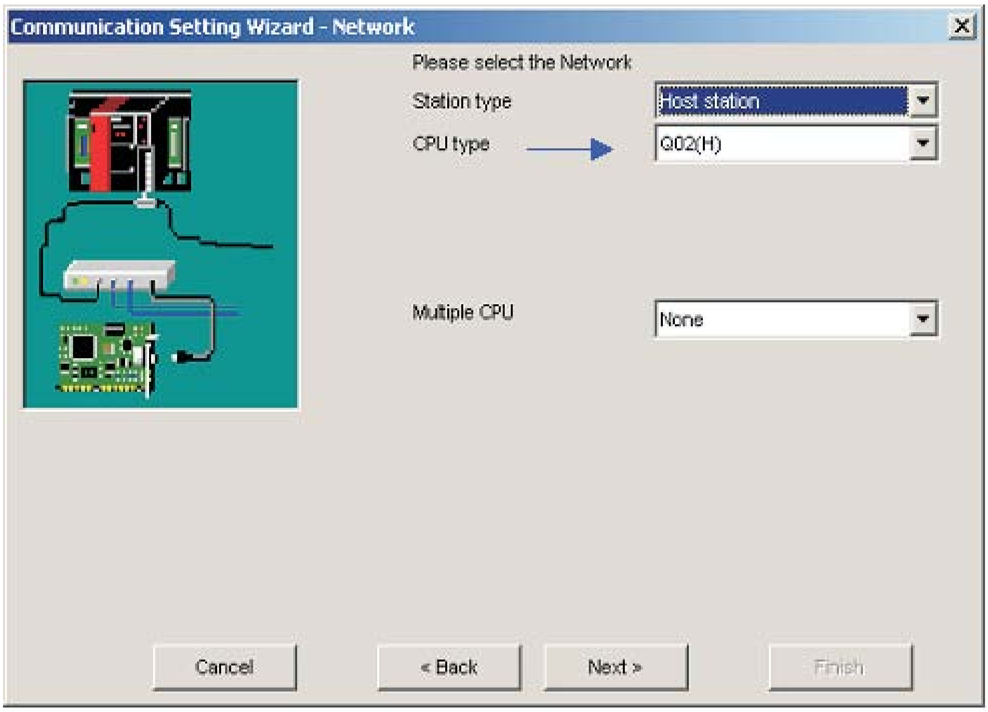
④ Select the UDP protocol and the default Port 5001



⑤ Configure the Communication settings of the PLC side required for the example system described earlier.

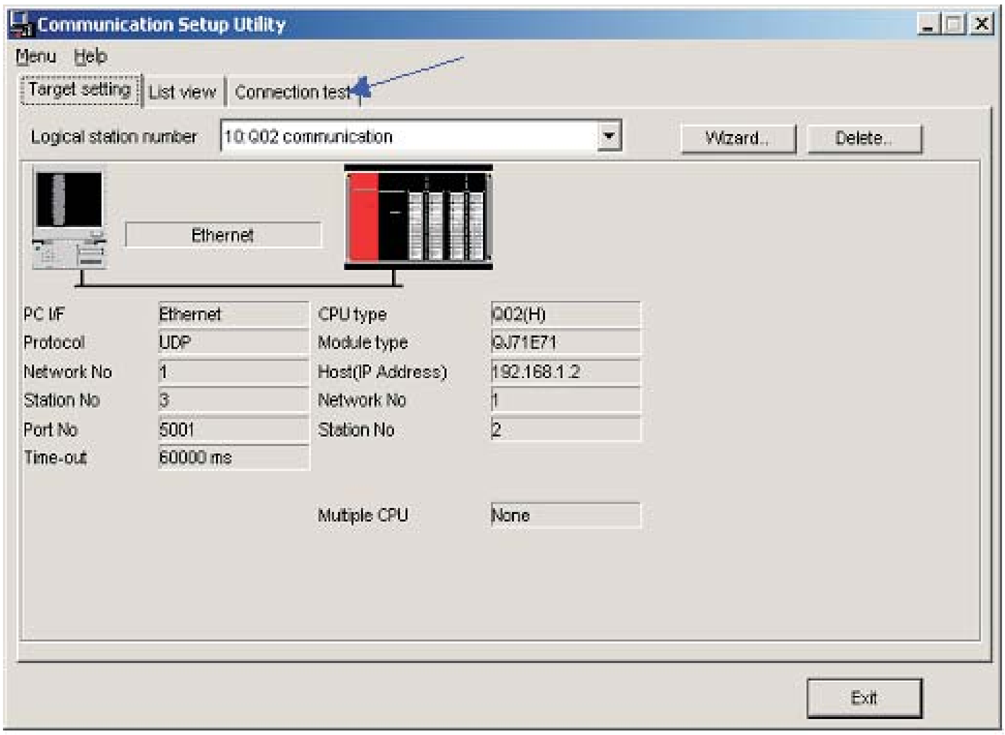


⑥ Select the correct CPU type.

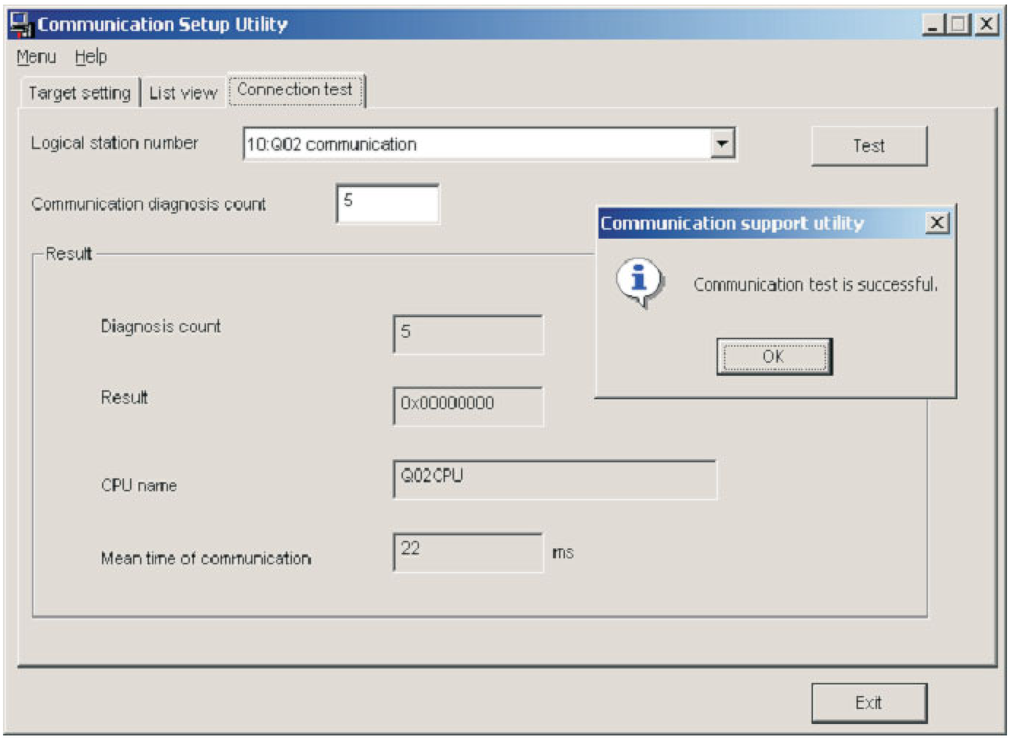


⑦ For the conclusion of the configuration define a name and press the ***Finish*** button

Now the definition of communication is finished. Under the folder ***Connection test*** the connection can be examined.



Select the ***Logical station number*** for which you want to accomplish the test. The ***Diagnosis count*** shows how many successful connection came. ***Result*** shows the test results. In case of an error an error number is indicated.



After configuring the communication paths you can access all controller devices (read/write) with Microsoft programming languages like MS Visual Basic, MS C++ etc.

The Mitsubishi MX components described above are powerful, user-friendly tools that make it very easy to connect your Mitsubishi PLC with the PC world.

A Appendix A

A.1 Special Relay Functionality for A & Q Series PLC´s

Diagnostic special relays (SM) are internal relays the application of which is fixed in the PLC. Therefore, they 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.

NOTES

The special relays SM1200 to SM1255 are used for QnA CPU. These relays are vacant with a Q CPU.

The special relays from SM1500 onward are dedicated for Q4AR CPU.

The headings in the table that follows have the following meanings.

Item	Meaning
Number	Indicates the number of the diagnostic special relay.
Name	Indicates the name of the diagnostic special relay.
Meaning	Contains the function of the diagnostic special relay in brief.
Description	Contains a detailed description of the diagnostic special relay.
Set by (if set)	Indicates whether the diagnostic special relay was set by the system or the user. **<Set by>** **S**: Set by the system **U**: Set by the user (via sequence program or a programming terminal in test mode) **S/U**: Set by the system or user Is indicated only if the setting is done by the system. **<if set>** **END processing**: Set during END processing **Initial**: Set during initial processing (Power ON, STOP->RUN) **Status change**: Set after status change **Error**: Set after error **Instruction execution**: Set during instruction execution **Request**: Set for user request (through SM, etc.)
A CPU M9[ ] [ ] [ ]	Indicates special relay M9 [ ] [ ] [ ] corresponding to the A CPU (Change and notation when contents changed). Items indicated as „New“ were newly added to the Q-Series/System Q CPU.
Valid for:	Indicates the corresponding CPU: ●: Can be applied to all types of CPU **Q CPU**: Can be applied to a System Q CPU **QnA CPU**: Can be applied to a CPU of the QnA series and Q2AS series **CPU name**: Can be applied only to the specific CPU (e.g. Q4AR CPU) **Rem**: Can be applied to a remote MELSECNET/H I/O module

Diagnostic Information

Number	Name	Meaning	Description	Set by (if set)	A CPU M9[ ][ ][ ]	Valid for:
SM0	Diagnostic errors	OFF: No error ON: Error	ON if diagnosis results show error occurrence (Includes external diagnosis). Stays ON subsequently even if normal operations restored.	S (Error)	New	● Rem
SM1	Self-diagnostic error	OFF: No self diagnosis errors ON: Self-diagnosis	Comes ON when an error occurs as a result of self-diagnosis. Stays ON subsequently even if normal operations restored.	S (Error)	M9008	
SM5	Error common information	OFF: No error common information ON: Error common information	When SM0 is ON, ON if there is error common information.	S (Error)	New	
SM16	Error individual information	OFF: No error individual information ON: Error individual information	When SM0 is ON, ON if there is error individual information.	S (Error)	New	
SM50	Error reset	OFF -> ON: Error reset	Conducts error reset operation.	U	New	
SM51	Battery low latch	OFF: Normal ON: Battery low	ON if battery voltage at CPU or memory card drops below rated value. Stays ON subsequently even after normal operation is restored. Synchronous with BAT. ALARM LED.	S (Error)	M9007	●
SM52	Battery low	OFF: Normal ON: Battery low	Same as SM51, but goes OFF subsequently when battery voltage returns to normal.	S (Error)	M9006	
SM53	AC DOWN detection	OFF: AC DOWN detected ON: AC DOWN not detected	Comes ON when a AC power supply module is used and a momentary power interruption not exceeding 20 ms has occured; reset by turning the power OFF then ON again.	S (Error)	M9005	●
			Comes ON when a DC power supply module is used and a momentary power interruption not exceeding 10 ms has occured; reset by turning the power OFF then ON again.			Q CPU
			Comes ON when a DC power supply module is used and a momentary power interruption not exceeding 1 ms has occured; reset by turning the power OFF then ON again.			QnA CPU
SM54	MINI link errors	OFF: Normal ON: Error	Goes ON if MINI (S3) link error is detected at even one of the installed AJ71PT32 (S3) modules. Stays ON subsequently even after normal operation is restored.	S (Error)	M9004	QnA CPU
SM56	Operation errors	OFF: Normal ON: Operation error	ON when operation error is generated. Stays ON subsequently even if normal operation is restored.	S (Error)	M9011	●
SM60	Blown fuse detection	OFF: Normal ON: Module with blown fuse	Comes ON even if there is only one output module with a blown fuse and remains ON even after return to normal. Blown fuse state is checked even for remote I/O station output modules.	S (Error)	M9000	● Rem
SM61	I/O module Verification error	OFF: Normal ON: Error	Comes ON if there is a discrepancy between the actual I/O modules and the registered information when the power is turned on. I/O module verification is also conducted for remote I/O station modules.	S (Error)	M9002	
SM62	Annunciator detection	OFF: Not detected ON: Detected	Goes ON if even one annunciator F goes ON.	S (Instruction execution)	M9009	●

Number	Name	Meaning	Description		Set by (if set)	A CPU M9[][][]	Valid for:
SM80	CHK detection	OFF: Not detected ON: Detected	Goes ON if error is detected by CHK instruction. Stays ON subsequently even after normal operation is restored.		S (Instruction execution)	New	QnA CPU, Q CPU (except Q00J, Q00 and Q01CPU)
SM90	Startup of watch-dog timer for step transition (Enabled only when SFC program exists)	OFF: Not startet (watchdog timer reset) ON: Started (watchdog timer started)	Corresponds to SD90	Goes ON when measurement of step transition watchdog timer is commenced. Resets watchdog timer when it goes OFF.	U	M9108	
SM91			Corresponds to SD91			M9109	
SM92			Corresponds to SD92			M9110	
SM93			Corresponds to SD93			M9111	
SM94			Corresponds to SD94			M9112	
SM95			Corresponds to SD95			M9113	
SM96			Corresponds to SD96			M9114	
SM97			Corresponds to SD97			New	
SM98			Corresponds to SD98			New	
SM99			Corresponds to SD99			New	

System Information

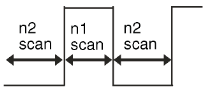
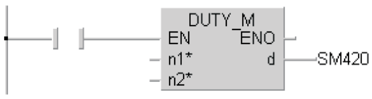
Number	Name	Meaning	Description	Set by (if set)	A CPU M9[][][]	Valid for:
SM202	LED off command	OFF -> ON: LED off	At change from OFF to ON, the LEDs corresponding to the individual bits at SD202 go off.	U	New	● (except Q00J, Q00, Q01CPU)
SM203	STOP contact	STOP state	Goes ON at STOP state.	S (Status change)	M9042	●
SM204	PAUSE contact	PAUSE state	GoesONat PAUSE state.	S (Status change)	M9041	
SM205	STEP-RUN contact	STEP-RUN state	Goes ONat STEP-RUN state.	S (Status change)	M9054	● (except Q00J, Q00 and Q01CPU)
SM206	PAUSE enable coil	OFF: PAUSE disabled ON: PAUSE enabled	PAUSE state is entered if this relay is ON when the remote PAUSE contact goes ON.	U	M9040	●
	Device test request acceptance status	OFF: Device test not yet executed ON: Device test executed	Comes ON when the device test mode is executed on the programming software.	S (Request)	New	Q00J Q00 and Q01 CPU
SM210	Clock data set request	OFF: Ignored ON: Set request	When this relay goes from OFF to ON, clock data being stored from SD210 through SD213 after execution of END instruction for changed scan is written to the clock device.	U	M9025	●
SM211	Clock data error	OFF: No error ON: Error	ON when error is generated in clock data (SD210 through SD213) value and OFF if no error is detected.	S (Request)	M9026	
SM212	Clock data display	OFF: Ignored ON: Display	Displays clock data as month, day, hour, minute and second at the LED display at front of CPU. (Enabled only for Q3A-CPU and Q4A-CPU)	U	M9027	Q3A, Q4A Q4AR CPU
SM213	Clock data read request	OFF: Ignored ON: Read request	When this relay is ON, clock data is read to SD210 through SD213 as BCD values.	U	M9028	● Rem

Number	Name	Meaning	Description	Set by (if set)	A CPU M9[][][]	Valid for:
SM240	No. 1 CPU reset flag	OFF: No reset ON: CPU 1 has been reset	This flag comes ON when the CPU no. 1 has been reset or has been removed from the base. The other CPUs of the multi-CPU system are also put in reset status.	S (Status change)	New	Q02, Q02H, Q06H, Q12H, Q25H CPU with function ver. B or later
SM241	No. 2 CPU reset flag	OFF: No reset ON: CPU 2 has been reset	This flag comes ON when the CPU no. 2 has been reset or has been removed from the base. In the other CPUs of the multi-CPU system the error code 7000 („MULTI CPU DOWN“) will occure.	S (Status change)	New	
SM242	No. 3 CPU reset flag	OFF: No reset ON: CPU 3 has been reset	This flag comes ON when the CPU no. 3 has been reset or has been removed from the base. In the other CPUs of the multi-CPU system the error code 7000 („MULTI CPU DOWN“) will occure.	S (Status change)	New	
SM243	No. 4 CPU reset flag	OFF: No reset ON: CPU 4 has been reset	This flag comes ON when the CPU no. 4 has been reset or has been removed from the base. In the other CPUs of the multi-CPU system the error code 7000 („MULTI CPU DOWN“) will occure.	S (Status change)	New	
SM244	No. 1 CPU error flag	OFF: No error ON: CPU no.1 is stopped due to an error	The set flag indicates that an error has occured which has stopped the CPU. The flag goes OFF when the CPU is normal or when an error occurs which will not stop the CPU.	S (Status change)	New	Q02, Q02H, Q06H, Q12H, Q25H CPU with function ver. B or later
SM245	No. 2 CPU error flag	OFF: No error ON: CPU no.2 is stopped due to an error		S (Status change)	New	
SM246	No. 3 CPU error flag	OFF: No error ON: CPU no.3 is stopped due to an error		S (Status change)	New	
SM247	No. 4 CPU error flag	OFF: No error ON: CPU no.41 is stopped due to an error		S (Status change)	New	

System Clocks

Number	Name	Meaning	Description	Set by (if set)	A CPU M9[ ] [ ] [ ]	Valid for:
SM400	Always ON	ON OFF	This flag is normally ON	S (Every END processing)	M9036	●
SM401	Always ON	ON OFF	This flag is normally OFF	S (Every END processing)	M9037	
SM402	ON for 1 scan only after RUN	ON 1 scan OFF	After RUN, ON for 1 scan only. This connection can be used for scan execution type programs only.	S (Every END processing)	M9038	
SM403	After RUN, OFF for 1 scan only	ON 1 scan OFF	After RUN, OFF for 1 scan only. This connection can be used for scan execution type programs only.	S (Every END processing)	M9039	
SM404	ON for 1 scan only after RUN	ON 1 scan OFF	After RUN, ON for 1 scan only. This connection can be used for scan execution type programs only.	S (Every END processing)	New	● (except Q00J, Q00 and Q01CPU)
SM405	After RUN, OFF for 1 scan only	ON 1 scan OFF	After RUN, OFF for 1 scan only. This connection can be used for scan execution type programs only.	S (Every END processing)	New	
SM409	0.01 second clock	0.005 s 0.005 s	Repeatedly changes between ON and OFF at 5-ms interval. When power supply is turned OFF, or reset is performed, goes from OFF to start.	S (Status change)	New	Q CPU (except Q00J, Q00 and Q01CPU)
SM410	0.1 second clock	0.05 s 0.05 s	Repeatedly changes between ON and OFF at each designated time interval. Operation continues even during STOP. When power supply is turned OFF, or reset is performed, goes from OFF to start.	S (Status change)	M9030	●
SM411	0.2 second clock	0.1 s 0.1 s			M9031	
SM412	1 second clock	0.5 s 0.5 s			M9032	
SM413	2 second clock	1 s 1 s			M9033	
SM414	2x n second clock	n (s) n (s)	Goes between ON and OFF in accordance with the number of seconds designated by SD414.		M9034 format change	
SM415	2 x n ms clock	n (ms) n (ms)	Goes between ON and OFF in accordance with the number of milliseconds designated by SD415.	S (Status change)	New	Q CPU (except Q00J, Q00 and Q01CPU)

System Clocks (continued)

<table>
<tr><th>Number</th><th>Name</th><th>Meaning</th><th>Description</th><th>Set by (if set)</th><th>A CPU M9[ ] [ ] [ ]</th><th>Valid for:</th></tr>
<tr><td>SM420</td><td>User timing clock No. 0</td><td rowspan="10"></td><td rowspan="7">Relay repeats ON/OFF switching at fixed scan intervals.
When power supply is turned ON, or reset is performed, goes from OFF to start.
The ON/OFF intervals are set with the DUTY instruction.
</td><td rowspan="7">S
(Every END processing)</td><td>M9020</td><td rowspan="7">●</td></tr>
<tr><td>SM421</td><td>User timing clock No.1</td><td>M9021</td></tr>
<tr><td>SM422</td><td>User timing clock No. 2</td><td>M9022</td></tr>
<tr><td>SM423</td><td>User timing clock No. 3</td><td rowspan="2">M9023</td></tr>
<tr><td>SM424</td><td>User timing clock No. 4</td></tr>
<tr><td>SM430</td><td>User timing clock No. 5</td><td rowspan="2">M9024</td></tr>
<tr><td>SM431</td><td>User timing clock No. 6</td></tr>
<tr><td>SM432</td><td>User timing clock No. 7</td><td rowspan="3">For use with SM420 through SM424 low speed programs.</td><td rowspan="3">S
(Every END processing)</td><td rowspan="3">New</td><td rowspan="3">●
(except Q00J, Q00 and Q01CPU)</td></tr>
<tr><td>SM433</td><td>User timing clock No. 8</td></tr>
<tr><td>SM434</td><td>User timing clock No. 9</td></tr>
</table>

A.2 A to Q series conversion correspondences

For a conversion from the MELSEC A series to the MELSEC Q series the special relays M9000 through M9255 (A series) correspond to the diagnostic relays SM1000 through SM1255 (Q series).

These diagnostic special relays are all set by the system and cannot be changed by a user-program. Users intending to set or reset these relays should alter their programs so that only real Q/QnA series diagnostic special relays are applied. An exception are the special relays M9084 and M9200 through M9255. If a user can set or reset some of these special relays befor conversion, the user can also set and reset the corresponding relays among SM1084 and SM1200 through SM1255 after the conversion.

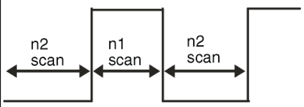
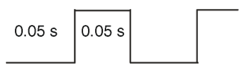
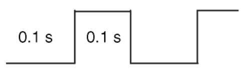
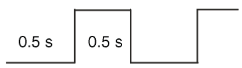
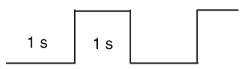
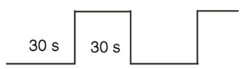
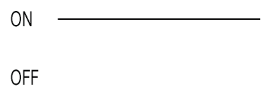
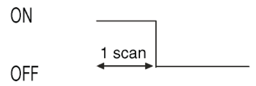
Refer to the manuals of the CPUs and the networks MELSECNET and MELSECNET/B for detailed information on the special relays of the A series.

NOTE

The processing time may be longer when converted special relays are used with a Q CPU. Don´t select ***A-PLC: Use special relay/special register from SM/SD 1000*** within the PC system setting in the GX Developer parameters when converted special relays are not used.

When a special relay for modification is provided, the device number should be changed to the provided System Q/QnA CPU special relay. When no special relay for modification is provided, the converted special relay can be used for the device number.

A CPU special relay	Special relay after conversion	Equivalent System Q/QnA diagnostic special relay	Name	Meaning	Valid for:
M9000	SM1000	—	Fuse blown	OFF: Normal ON: Fuse blown module with blown fuse present	System Q/ QnA CPU
M9002	SM1002	—	I/O module verification error	OFF: Normal ON: Error	
M9004	SM1004	—	MINI link error	OFF: Normal ON: Error	QnA CPU
M9005	SM1005	—	AC DOWN detection	OFF: AC DOWN not detected ON: AC DOWN detected	System Q/ QnA CPU
M9006	SM1006	—	Battery low	OFF: Normal ON: Battery low	
M9007	SM1007	—	Battery low (latched)	OFF: Normal ON: Battery low	
M9008	SM1008	SM1	Self-diagnostic error	OFF: No error ON: Error	
M9009	SM1009	SM62	Annunciator detection	OFF: No F number detected ON: F number detected	
M9011	SM1011	SM56	Operation error flag	OFF: No error ON: Error	
M9012	SM1012	SM700	Carry Flag	OFF: Carry OFF ON: Carry ON	
M9016	SM1016	The device does not work with a System Q/QnA CPU	Data memory clear flag	OFF: Ignored ON: Output cleared	
M9017	SM1017	The device does not work with a System Q/QnA CPU	Data memory clear flag	OFF: Ignored ON: Output cleared	

A CPU special relay	Special relay after conversion	Equivalent System Q/QnA diagnostic special relay	Name	Meaning	Valid for:
M9020	SM1020	—	User timing clock No. 0		
M9021	SM1021	—	User timing clock No. 1		
M9022	SM1022	—	User timing clock No. 2		
M9023	SM1023	—	User timing clock No. 3		
M9024	SM1024	—	User timing clock No. 4		
M9025	SM1025	—	Clock data set request	OFF: Ignored ON: Set request present used	
M9026	SM1026	—	Clock data error	OFF: No error ON: Error	
M9027	SM1027	—	Clock data display	OFF: Ignored ON: Display	
M9028	SM1028	—	Clock data read request	OFF: Ignored ON: Read request	
M9029	SM1029	The device does not work with a System Q/QnA CPU	Batch processing of data communications request	OFF: Batch processing not conducted ON: Batch processing conducted	
M9030	SM1030	—	0.1 second clock		
M9031	SM1031	—	0.2 second clock		System Q/ QnA CPU
M9032	SM1032	—	1 second clock		
M9033	SM1033	—	2 second clock		
M9034	SM1034	—	1 minute clock		
M9036	SM1036	—	Always ON		
M9037	SM1037	—	Always OFF		
M9038	SM1038	—	ON for 1 scan only after RUN		

A CPU special relay	Special relay after conversion	Equivalent QnA diagnostic special relay	Name	Meaning	Valid for:
M9039	SM1039	—	RUN flag (After RUN, OFF for 1 scan only)	ON 1 scan OFF	
M9040	SM1040	SM206	PAUSE enable coil	OFF: PAUSE disabled ON : PAUSE enabled	
M9041	SM1041	SM204	PAUSE status contact	OFF: PAUSE not in effect ON: PAUSE in effect	
M9042	SM1042	SM203	STOP status contact	OFF: STOP not in effect ON: STOP in effect	
M9043	SM1043	SM805	Sampling trace completed	OFF: Sampling trace in progress ON: Sampling trace completed	
M9044	SM1044	SM803	Sampling trace	0 1 STRA Same as execution 1 0 STRAR Same as execution	
M9045	SM1045	The device does not work with a System Q/QnA CPU.	Watchdog timer (WDT) reset	OFF: Does not reset WDT ON: Resets WDT	System Q/ QnA CPU
M9046	SM1046	SM802	Sampling trace	OFF: Trace not in progress ON: Trace in progress	
M9047	SM1047	SM801	Sampling trace preparations	OFF: Sampling Trace suspended ON: Sampling Trace started	
M9049	SM1049	SM701	Selection of number of characters output	OFF: Output until NUL ON: 16 characters output	
M9051	SM1051	The device does not work with a System Q/QnA CPU.	CHG instruction execution disable	OFF: Enabled ON: Disable	
M9052	SM1052	The device does not work with a System Q/QnA CPU.	SEG instruction switch	OFF: 7 segment display ON: I/O partial refresh	
M9054	SM1054	SM205	STEP RUN flag	OFF: STEP RUN not in effect ON: STEP RUN in effect	
M9055	SM1055	SM808	Status latch completion flag	OFF: Not completed ON: Completed	QnA CPU
M9056	SM1056		Main side P, I set request	OFF: Other than when P, I set being requested ON: P, I set being requested	
M9057	SM1057		Sub side P, I set request		
M9058	SM1058	These devices do not work with a System Q/QnA CPU.	Main program P, I set completion	Momentarily ON at P, I set completion	System Q/ QnA CPU
M9059	SM1059		Sub program P, I set completion	Momentarily ON at P, I set completion	
M9060	SM1060		Sub program 2 P, I set request	OFF: Other than when P, I set being requested ON: P, I set being requested	
M9061	SM1061		Sub program 3 P, I set request		

A CPU special relay	Special relay after conversion	Equivalent QnA diagnostic special relay	Name	Meaning	Valid for:
M9065	SM1065	SM711	Divided processing execution detection	OFF: Divided processing not underway ON: During divided processing	QnA CPU
M9066	SM1066	SM712	Divided processing request flag	OFF: Batch processing ON: Divided processing	QnA CPU
M9070	SM1070	The device does not work with a System Q/QnA CPU.	A8UPU/A8PUJ required search time	OFF: Read time not shortened ON: Read time shortened	System Q/ QnA CPU
M9081	SM1081	SM714	Communication request registration area BUSY signal	OFF: Empty spaces in communication request registration area ON: No empty spaces in communication request registration area	QnA CPU
M9084	SM1084	The device does not work with a System Q/QnA CPU.	Error check	OFF: Error check executed ON: No error check	System Q/ QnA CPU
M9091	SM1091	The device does not work with a System Q/QnA CPU.	Instruction error flag	OFF: No error ON: Error	System Q/ QnA CPU
M9094	SM1094	SM251	I/O change flag	OFF: Replacement ON: No replacement	QnA CPU
M9100	SM1100	SM320	Presence/absence of SFC program	OFF: SFC programs not used ON: SFC programs used	System Q/ QnA CPU
M9101	SM1101	SM321	Start/stop SFC program	OFF: SFC programs stop ON: SFC programs start	System Q/ QnA CPU
M9102	SM1102	SM322	SFC program start state	OFF: Initial Start ON: Continue	System Q/ QnA CPU
M9103	SM1103	SM323	Presence/absence of continuous transition	OFF: Continuous transition not effective ON: Continuous transition effective	System Q/ QnA CPU
M9104	SM1104	SM324	Continuous transition suspension flag	OFF: When transition is completed ON: When no transition	System Q/ QnA CPU
M9108	SM1108	SM90	Step transition watchdog timer start (equivalent of D9108)	OFF: Watchdog timer reset ON: Watchdog timer reset start	System Q/ QnA CPU
M9109	SM1109	SM91	Step transition watchdog timer start (equivalent of D9109)	OFF: Watchdog timer reset ON: Watchdog timer reset start	System Q/ QnA CPU
M9110	SM1110	SM92	Step transition watchdog timer start (equivalent of D9110)	OFF: Watchdog timer reset ON: Watchdog timer reset start	System Q/ QnA CPU
M9111	SM1111	SM93	Step transition watchdog timer start (equivalent of D9111)	OFF: Watchdog timer reset ON: Watchdog timer reset start	System Q/ QnA CPU
M9112	SM1112	SM94	Step transition watchdog timer start (equivalent of D9112)	OFF: Watchdog timer reset ON: Watchdog timer reset start	System Q/ QnA CPU
M9113	SM1113	SM95	Step transition watchdog timer start (equivalent of D9113)	OFF: Watchdog timer reset ON: Watchdog timer reset start	System Q/ QnA CPU
M9114	SM1114	SM96	Step transition watchdog timer start (equivalent of D9114)	OFF: Watchdog timer reset ON: Watchdog timer reset start	System Q/ QnA CPU
M9180	SM1180	SM825	Active step sampling trace execution flag	OFF: Trace will be started ON: Trace completed	System Q/ QnA CPU
M9181	SM1181	SM822	Active step sampling trace execution flag	OFF: Trace not being executed ON: Trace execution under way	System Q/ QnA CPU

A CPU special relay	Special relay after conversion	Equivalent QnA diagnostic special relay	Name	Meaning	Valid for:
M9182	SM1182	SM821	Active step sampling trace permission	OFF: Trace disable/suspend ON: Trace enable	System Q/ QnA CPU
M9196	SM1196	SM325	Operation output at block stop	OFF: Coil output OFF ON: Coil output ON	
M9197 . M9198	SM1197 . SM1198	The device does not work with a System Q/QnA CPU	Switch between blown fuse and I/O verification error display	Display is changed depending on combination of M9197 ON/OFF state and M9198 ON/OFF state.	
M9199	SM1199	The device does not work with a System Q/QnA CPU	On-line recovery of sampling trace status latch data	OFF : Does not perform data recovery ON: Performs data recovery	
M9200	SM1200	—	LRDP instruction reception	OFF: Not accepted ON: Accepted	QnA CPU
M9201	SM1201	—	LRDP instruction completion	OFF: Not completed ON: End	
M9202	SM1202	—	LWTP instruction reception	OFF: Not accepted ON: Accepted	
M9203	SM1203	—	LWTP instruction completion	OFF: Not completed ON: End	
M9204	SM1204	—	LRDP instruction completion	OFF: Not completed ON: End	
M9205	SM1205	—	LWTP instruction completion	OFF: Not completed ON: End	
M9206	SM1206	—	Host station link parameter error	OFF: Normal ON: Abnormal	
M9207	SM1207	—	Link parameter check results	OFF: YES ON: NO	
M9208	SM1208	—	Sets master station B and W transmission range (for lower link master stations only).	OFF: Transmits to tier 2 and tier 3 ON: Transmits to tier 2 only	
M9209	SM1209	—	Link parameter check command (for lower link master stations only).	OFF: Executing the check function ON: Check non-execution	
M9210	SM1210	—	Link card error (for local station)	OFF: Normal ON: Abnormal	
M9211	SM1211	—	Link module error (for master station use)	OFF: Normal ON: Abnormal	
M9224	SM1224	—	Link state	OFF: Online ON: Offline, station-to-station test, or self-loopback test	
M9225	SM1225	—	Forward loop error	OFF: Normal ON: Abnormal	
M9226	SM1226	—	Reverse loop error	OFF: Normal ON: Abnormal	
M9227	SM1227	—	Loop test state	OFF: Not being executed ON: Forward or reverse loop test execution underway	

A CPU special relay	Special relay after conversion	Equivalent QnA diagnostic special relay	Name	Meaning	Valid for:
M9232	SM1232	—	Local station operation state	OFF: RUN or STEP RUN state ON: STOP or PAUSE state	QnA CPU
M9233	SM1233	—	Local station error detect state	OFF: No errors ON: Error detection	
M9235	SM1235	—	Local station, remote I/O station parameter error detect state	OFF: No errorsl ON: Error detection	
M9236	SM1236	—	Local station, remote I/O station parameter error detect state	OFF: No communications ON: Communications underway	
M9237	SM1237	—	Local station, remote I/O station error	OFF: Normal ON: Abnormal	
M9238	SM1238	—	Local station, remote I/O station forward or reverse loop error	OFF: Normal ON: Abnormal	
M9240	SM1240	—	Link state	OFF: Online ON: Offline, station-to-station test or self-loopback test	
M9241	SM1241	—	Forward loop line error	OFF: Normal ON: Abnormal	
M9242	SM1242	—	Reverse loop line error	OFF: Normal ON: Abnormal	
M9243	SM1243	—	Loopback implementation	OFF: Loopback not being conducted ON: Loopback implementation	
M9246	SM1246	—	Data not received	OFF: Reception ON: No reception	
M9247	SM1247	—	Data not received	OFF: Reception ON: No reception	
M9250	SM1250	—	Parameters not received	OFF: Reception ON: No reception	
M9251	SM1251	—	Link relay	OFF: Normal ON: Abort	
M9252	SM1252	—	Loop test state	OFF: Not being executed ON: Forward or reverse loop test execution underway	
M9253	SM1253	—	Master station operation state	OFF: RUN or STEP RUN state ON: STOP or PAUSE state	
M9254	SM1254	—	Local station other than host station operation state	OFF: RUN or STEP RUN state ON: STOP or PAUSE state	
M9255	SM1255	—	Local station other than host station error	OFF: Normal ON: Abnormal	

A.3 Special Registers (SD)

The special registers (SD) are internal registers with fixed application in the PLC. Therefore, they cannot be used like other registers in a sequence program. However, some of them can be written as needed in order to control the CPU.

Data stored in special registers are stored as BIN values if no special designation has been made to it.

Represented here are some of the most commonly used devices.

NOTES

The special registers SD1200 to SD1255 are used for QnA CPU. These relays are vacant with a Q CPU.

The special registers from SM1500 onward are dedicated for Q4AR CPU.

The headings in the table that follows have the following meanings.

Item	Meaning
Number	Indicates the number of the special register.
Name	Indicates the name of the special relgister.
Meaning	Contains the function of the special register in brief.
Description	Contains a detailed description of the special register.
Set by (if set)	Indicates whether the diagnostic special relay was set by the system or the user. **<Set by>** **S**: Set by the system **U**: Set by the user (via sequence program or a programming terminal in test mode) **S/U**: Set by the system or user Is indicated only if the setting is done by the system. **<if set>** **END processing**: Set during END processing **Initial**: Set during initial processing (Power ON, STOP->RUN) **Status change**: Set after status change **Error**: Set after error **Instruction execution**: Set during instruction execution **Request**: Set for user request (through SM, etc.)

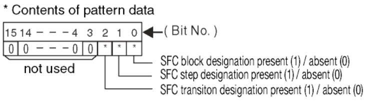
Number	Name	Meaning	Description	Set by (if set)	ACPU register D9 [ ] [ ] [ ]	Valid for:
SD0	Diagnostic errors	Diagnosis error code	Error codes for errors found by diagnosis are stored as BIN data. Contents identical to latest fault history information.	S (Error)	D9008 format change	●
SD1	Clock time for diagnosis error occurrence	Clock time for diagnosis error occurrence	Year (last two digits) and month that SD0 data was updated is stored as BCD 2-digit code. Example: **October 1995** **H9510** b15 b8 b7 b0 Year (0 to 99) | Month (1 to 31)	S (Error)	New	
SD2			The day and hour that SD0 was updated is stored as BCD 2-digit code. Example: **10 p.m. on 25th** **H2510** b15 b8 b7 b0 Day (1 to 31) | Hour (0 to 23)			
SD3			The minute and second that SD0 data was updated is stored as BCD 2-digit code. Example: **35 min 48s** **H3548** b15 b8 b7 b0 Minute (1 to 60) | Second (1 to 60)			
SD4	Error information categories	Error information category code	Category codes which help indicate what type of information is being stored in the common information areas (SD5 through SD15) and the individual information areas (SD16 through SD26) are stored here. b15 b8 b7 b0 Individual error info. | Common error info. The common information category codes store the following codes: 0: No error 1: Unit/module No. 2: File name/Drive name 3: Time (value set) 4: Program error location The individual information category codes store the following codes: 0: No error 1: (Open) 2: File name/Drive name 3: Time (value actually measured) 4: Program error location 5: Parameter number 6: Annunciator number 7: Check instruction malfunction number	S (Error)	New	

<table>
<tr><th>Number</th><th>Name</th><th>Meaning</th><th>Description</th><th>Set by (if set)</th><th>ACPU register D9 [ ] [ ] [ ]</th><th>Valid for:</th></tr>
<tr>
<td>SD5
SD6
SD7
SD8
SD9
SD10
SD11
SD12
SD13
SD14
SD15</td>
<td>Error common information</td>
<td>Error common information</td>
<td>
Common information corresponding to the error codes (SD0) is stored here.
The following four types of information are stored here:
(1) Unit/module No.
<table>
<tr><th>Number</th><th>Meaning</th></tr>
<tr><td>SD5</td><td>Station / module number</td></tr>
<tr><td>SD6</td><td>I/O number</td></tr>
<tr><td>SD7</td><td rowspan="9">Vacant</td></tr>
<tr><td>SD8</td></tr>
<tr><td>SD9</td></tr>
<tr><td>SD10</td></tr>
<tr><td>SD11</td></tr>
<tr><td>SD12</td></tr>
<tr><td>SD13</td></tr>
<tr><td>SD14</td></tr>
<tr><td>SD15</td></tr>
</table>
(2) File name/Drive name
Example:
File name = ABCDEFGH.IJK
<table>
<tr><th>Number</th><th colspan="2">Meaning</th></tr>
<tr><td>SD5</td><td colspan="2">Drive</td></tr>
<tr><td>SD6</td><td colspan="2" rowspan="4">File name
ASCII code: 8 characters</td></tr>
<tr><td>SD7</td></tr>
<tr><td>SD8</td></tr>
<tr><td>SD9</td></tr>
<tr><td>SD10</td><td>Extension</td><td>2EH(.)</td></tr>
<tr><td>SD11</td><td colspan="2">ASCII code: 3 characters</td></tr>
<tr><td>SD12</td><td colspan="2" rowspan="4">Vacant</td></tr>
<tr><td>SD13</td></tr>
<tr><td>SD14</td></tr>
<tr><td>SD15</td></tr>
</table>
<table>
<tr><th>b15</th><th>b0</th></tr>
<tr><td>B</td><td>A</td></tr>
<tr><td>D</td><td>C</td></tr>
<tr><td>F</td><td>E</td></tr>
<tr><td>H</td><td>G</td></tr>
<tr><td>I</td><td>.</td></tr>
<tr><td>K</td><td>J</td></tr>
</table>
(3) Time (value set)
<table>
<tr><th>Number</th><th>Meaning</th></tr>
<tr><td>SD5</td><td>Time: 1µs-steps (0 to 999 µs)</td></tr>
<tr><td>SD6</td><td>Time: 1ms-steps (0 to 999 ms)</td></tr>
<tr><td>SD7</td><td rowspan="9">Vacant</td></tr>
<tr><td>SD8</td></tr>
<tr><td>SD9</td></tr>
<tr><td>SD10</td></tr>
<tr><td>SD11</td></tr>
<tr><td>SD12</td></tr>
<tr><td>SD13</td></tr>
<tr><td>SD14</td></tr>
<tr><td>SD15</td></tr>
</table>
(4) Program error location
<table>
<tr><th>Number</th><th colspan="2">Meaning</th></tr>
<tr><td>SD5</td><td colspan="2" rowspan="4">File name
(ASCII code: 8 characters)</td></tr>
<tr><td>SD6</td></tr>
<tr><td>SD7</td></tr>
<tr><td>SD8</td></tr>
<tr><td>SD9</td><td>Extension</td><td>2EH (.)</td></tr>
<tr><td>SD10</td><td colspan="2">(ASCII code: 3 characters)</td></tr>
<tr><td>SD11</td><td colspan="2">Pattern*</td></tr>
<tr><td>SD12</td><td colspan="2">Block No.</td></tr>
<tr><td>SD13</td><td colspan="2">Step / transition No.</td></tr>
<tr><td>SD14</td><td colspan="2">Sequence step No. (L)</td></tr>
<tr><td>SD15</td><td colspan="2">Sequence step No. (H)</td></tr>
</table>
* Contents of pattern data
<table>
<tr><td>15</td><td>14 – – – 4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>(Bit No.)</td></tr>
<tr><td>0</td><td>0 – – – 0</td><td>0</td><td>*</td><td>*</td><td>*</td><td></td></tr>
</table>
not used
SFC block designation present (1) / absent (0)
SFC step designation present (1) / absent (0)
SFC transiton designation present (1) / absent (0)
</td>
<td>S
(Error)</td>
<td>New</td>
<td>●</td>
</tr>
</table>

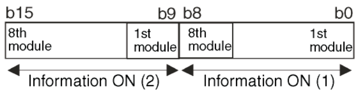
Meaning of extensions

SD10 (SD9)	SD11 (SD10)		Extension name	File type
Higher byte	Lower byte	Higher byte		
51H	50H	41H	QPA	Parameters
51H	50H	47H	QPG	Sequence program
51H	43H	44H	QCD	Device comment
51H	44H	49H	QDI	Device initial value
51H	44H	52H	QDR	File register
51H	44H	53H	QDS	Simulation data
51H	44H	4CH	QDL	Local device
51H	54H	53H	QTS	Sampling trace data (QnA-CPU only)
51H	54H	4CH	QTL	Status latch data (QnA-CPU only)
51H	54H	50H	QTP	Program trace data (QnA-CPU only)
51H	54H	52H	QTR	SFC trace file
51H	46H	44H	QFD	Trouble history data

<table>
<tr><th>Number</th><th>Name</th><th>Meaning</th><th>Description</th><th>Set by (if set)</th><th>ACPU register D9 [ ] [ ] [ ]</th><th>Valid for</th></tr>
<tr><td>SD16</td><td rowspan="11">Error individual information</td><td rowspan="11">Error individual information</td><td rowspan="11">
Individual information corresponding to the error codes (SD0) is stored here.
The following six types of information are stored here:

(1) File name/Drive name
<table>
<tr><th>Number</th><th colspan="2">Meaning</th></tr>
<tr><td>SD16</td><td colspan="2">Drive</td></tr>
<tr><td>SD17</td><td colspan="2" rowspan="4">File name
ASCII code: 8 characters</td></tr>
<tr><td>SD18</td></tr>
<tr><td>SD19</td></tr>
<tr><td>SD20</td></tr>
<tr><td>SD21</td><td>Extension</td><td>2EH(.)</td></tr>
<tr><td>SD22</td><td colspan="2">ASCII code: 3 characters</td></tr>
<tr><td>SD23</td><td colspan="2" rowspan="4">Vacant</td></tr>
<tr><td>SD24</td></tr>
<tr><td>SD25</td></tr>
<tr><td>SD26</td></tr>
</table>
Example:
File name =
ABCDEFGH.IJK
<table>
<tr><th>b15</th><th>b0</th></tr>
<tr><td>B</td><td>A</td></tr>
<tr><td>D</td><td>C</td></tr>
<tr><td>F</td><td>E</td></tr>
<tr><td>H</td><td>G</td></tr>
<tr><td>I</td><td>.</td></tr>
<tr><td>K</td><td>J</td></tr>
</table>
(2) Time (value actually measured)
<table>
<tr><th>Number</th><th>Meaning</th></tr>
<tr><td>SD16</td><td>Time: 1µs-steps (0 to 999 µs)</td></tr>
<tr><td>SD17</td><td>Time: 1ms-steps (0 to 999 ms)</td></tr>
<tr><td>SD18</td><td rowspan="9">Vacant</td></tr>
<tr><td>SD19</td></tr>
<tr><td>SD20</td></tr>
<tr><td>SD21</td></tr>
<tr><td>SD22</td></tr>
<tr><td>SD23</td></tr>
<tr><td>SD24</td></tr>
<tr><td>SD25</td></tr>
<tr><td>SD26</td></tr>
</table>
(3) Program error location
<table>
<tr><th>Number</th><th colspan="2">Meaning</th></tr>
<tr><td>SD16</td><td colspan="2" rowspan="4">File name
(ASCII code: 8 characters)</td></tr>
<tr><td>SD17</td></tr>
<tr><td>SD18</td></tr>
<tr><td>SD19</td></tr>
<tr><td>SD20</td><td>Extension</td><td>2EH (.)</td></tr>
<tr><td>SD21</td><td colspan="2">(ASCII code: 3 characters)</td></tr>
<tr><td>SD22</td><td colspan="2">Pattern*</td></tr>
<tr><td>SD23</td><td colspan="2">Block No.</td></tr>
<tr><td>SD24</td><td colspan="2">Step / transition No.</td></tr>
<tr><td>SD25</td><td colspan="2">Sequence step No. (L)</td></tr>
<tr><td>SD26</td><td colspan="2">Sequence step No. (H)</td></tr>
</table>
* Contents of pattern data


</td><td rowspan="11">S
(Error)</td><td rowspan="11">New</td><td rowspan="11">●</td></tr>
<tr><td>SD17</td></tr>
<tr><td>SD18</td></tr>
<tr><td>SD19</td></tr>
<tr><td>SD20</td></tr>
<tr><td>SD21</td></tr>
<tr><td>SD22</td></tr>
<tr><td>SD23</td></tr>
<tr><td>SD24</td></tr>
<tr><td>SD25</td></tr>
<tr><td>SD26</td></tr>
</table>

Number	Name	Meaning	Description	Set by (if set)	ACPU register D9 [ ] [ ] [ ]	Valid for:
SD54	MINI link errors	Error detection state	(1) The relevant station bit goes ON when any of the installed MINI (-S3) X(n+0) /X(n+20), X(n+6)/()n+26), X(n+7)/(n+27) or X(n+8)/Xn+28) goes ON. (2) Goes ON when communications between the installed MINI (-S3) and the CPU are not possible.  	S (Error)	D9004 format change	QnA-CPU
SD60	Blown fuse number	Number of module with blown fuse	Value stored here is the lowest station number of the module with the blown fuse, divided by 16.	S (Error)	D9000	● Rem
SD61	I/O module verification error	I/O module verification error module number	The lowest number of the module where the I/O module verification number took place.	S (Error)	D9002	
SD62	Annunciator number	Annunciator number	The first annunciator number to be detected is stored here.	S (Instruction execution)	D9009	●
SD63	Number of annunciators	Number of annunciators	Stores the number of annunciators searched.	S (Instruction execution)	D9124	

Number	Name	Meaning	Description	Set by (if set)	ACPU register D9 [ ] [ ] [ ]	Valid for
SD64	Table of detected annunciator numbers	Annunciator detection number	When F goes ON due to OUT F or SET F, the F numbers which go progressively ON from SD64 through SD79 are registered. F numbers turned OFF by RST F are deleted from SD64 to SD79, and are shifted to the data register following the data register where the deleted F numbers had been stored. Execution of the LEDR instruction shifts the contents of SD64 to SD79 up by one. (This can also be done by using the INDICATOR RESET switch on the front of the CPU of the Q3A/Q4ACPUI.) After 16 annunciators have been detected, detection of the 17th will not be stored from SD64 through SD79.	S (Instruction execution)	D9125	●
SD65					D9126	
SD66					D9127	
SD67					D9128	
SD68					D9129	
SD69					D9130	
SD70					D9131	
SD71					D9132	
SD72					New	
SD74					New	
SD75					New	
SD76					New	
SD77					New	
SD78					New	
SD79					New	
SD80	CHK number	CHK number	Error codes detected by the CHK instruction are stored as BCD code.	S (Instruction execution)	New	● (except Q00J, Q00 and Q01CPU)

		SET F50	SET F25	SET F19	SET F25	SET F15	SET F70	SET F65	SET F38	SET F110	SET F151	SET F210	LEDR	
SD62	0	50	50	50	50	50	50	50	50	50	50	50	99	Number detected
SD63	0	1	2	3	2	3	4	5	6	7	8	9	8	Number of annunciators detected
SD64	0	50	50	50	50	50	50	50	50	50	50	50	99	Number detected
SD65	0	0	25	25	99	99	99	99	99	99	99	99	15	
SD66	0	0	0	99	0	15	15	15	15	15	15	15	15	
SD67	0	0	0	0	0	0	70	70	70	70	70	70	65	
SD68	0	0	0	0	0	0	0	65	65	65	65	65	38	
SD69	0	0	0	0	0	0	0	0	38	38	38	38	110	
SD70	0	0	0	0	0	0	0	0	0	110	110	110	151	
SD71	0	0	0	0	0	0	0	0	0	0	151	151	210	
SD72	0	0	0	0	0	0	0	0	0	0	0	210	0	
SD73	0	0	0	0	0	0	0	0	0	0	0	0	0	
SD74	0	0	0	0	0	0	0	0	0	0	0	0	0	
SD75	0	0	0	0	0	0	0	0	0	0	0	0	0	
SD76	0	0	0	0	0	0	0	0	0	0	0	0	0	
SD77	0	0	0	0	0	0	0	0	0	0	0	0	0	
SD78	0	0	0	0	0	0	0	0	0	0	0	0	0	
SD79	0	0	0	0	0	0	0	0	0	0	0	0	0	

Number	Name	Meaning	Description		Set by (if set)	ACPU register D9 [ ] [ ] [ ]	Valid for:
SD90	Step transition watchdog timer setting value (Enabled only when SFC program exists)	F number for timer set value and time over error	Corresponds to SM90	F numbers that are set ON at setting value of step transition watchdog timer and watchdog timer over errors. b15 b8 b7 b0 Setting of F-number (0 to 255) / Setting of timer limit value (1 to 255 s, in 1 s steps) Timer is started by turning SM90 through SM99 ON during active step, and if the transition conditions for the relevant steps are not met within the timer limits, the designated annunciator (F) will go ON.	U	D9108	● (except Q00J, Q00 and Q01CPU)
SD91			Corresponds to SM91			D9109	
SD92			Corresponds to SM92			D9110	
SD93			Corresponds to SM93			D9111	
SD94			Corresponds to SM94			D9112	
SD95			Corresponds to SM95			D9113	
SD96			Corresponds to SM96			D9114	
SD97			Corresponds to SM97			New	
SD98			Corresponds to SM98			New	
SD99			Corresponds to SM99			New	
SD100	Transmission speed	Stores the transmission speed specified in the serial communication setting.	K96: 9600 bps, K192: 19.2 kbps, K384: 38.4 kbps, K576: 57.6 kbps, K1152: 115.2 kbps		S (power on or reset)	New	Q00JCPU Q00CPU Q01CPU
SD101	Communication settings	Stores the settings for serial communication	Bit 4 = OFF: Without sumcheck Bit 4 = ON: With sumcheck Bit 5 = OFF: Online program correction disabled Bit 5 = ON: Online program correction enabled The other bits have no function.			New	
SD102	Message waiting time	Stores the waiting time specified in the serial communication setting.	0: No waiting time 1 to F_H: Waiting time (unit: 10 ms) Default: 0			New	
SD105	CH1 transmission speed setting (RS232)	Stores the present transmission speed.	K3: 300 bps, K6: 600 bps, K24: 2400 bps, K48: 4800 bps, K96: 9600 bps, K192: 19.2 kbps, K384: 38.4 kbps, K576: 57.6 kbps, K1152: 115.2 kbps		S	New	Q CPU (except Q00J, Q00 and Q01CPU)
SD110	Data sending result	Stores the data sending result when the serial communication is used.	Stores the error code which occured during transmission using the serial communication.		S (Error)	New	Q00JCPU Q00CPU Q01CPU
SD111	Data receiving result	Stores the data receiving result when the serial communication is used.	Stores the error code which occured when data was received using the serial communication.		S (Error)	New	
SD120	Error number for external power supply OFF	Module number which has external power supply error	Stores the smallest head number of the module whose external power supply is OFF.		S (Error)	New	Q CPU (except Q00J, Q00 and Q01CPU)

<table>
<tr><th>Number</th><th>Name</th><th>Meaning</th><th>Description</th><th>Set by (if set)</th><th>ACPU register D9 [ ] [ ] [ ]</th><th>Valid for:</th></tr>
<tr><td>SD130</td><td rowspan="8">Modules with blown fuse</td><td rowspan="8">The bit pattern (16 Bit) indicates the modules with a blown fuse.
0 : No blown fuse
1 : Blown fuse detected</td><td rowspan="8">The number of output modules whose fuses have blown are input as a bit pattern in units of 16 points. If the module numbers are set by parameter, the parameter-set numbers are stored.

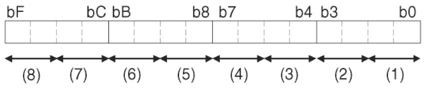
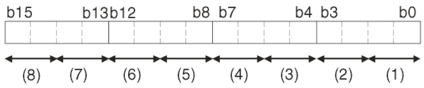
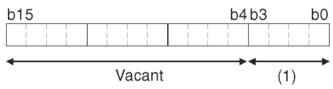
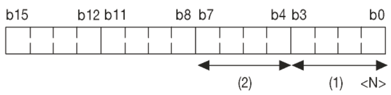
Blown fuses of remote station output modules will be detected also.

A set bit is not automatically cleared when the module with the blown fuse is replaced. The flag is cleared by an error reset operation.

<table><tr><th></th><th>b15</th><th>b14</th><th>b13</th><th>b12</th><th>b11</th><th>b10</th><th>b9</th><th>b8</th><th>b7</th><th>b6</th><th>b5</th><th>b4</th><th>b3</th><th>b2</th><th>b1</th><th>b0</th></tr><tr><td>SD130</td><td>0</td><td>0</td><td>0</td><td>1 (YC0)</td><td>0</td><td>0</td><td>0</td><td>1 (Y80)</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>SD131</td><td>1 (Y1F0)</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1 (Y1A)</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>SD137</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1 (Y1F30)</td><td>0</td><td>0</td><td>0</td></tr></table>Blown fuse at the module with the head I/O number Y1F80.</td><td rowspan="8">S
(Error)</td><td rowspan="8">New</td><td rowspan="16">Q00JCPU
Q00CPU
Q01CPU</td></tr>
<tr><td>SD131</td></tr>
<tr><td>SD132</td></tr>
<tr><td>SD133</td></tr>
<tr><td>SD134</td></tr>
<tr><td>SD135</td></tr>
<tr><td>SD136</td></tr>
<tr><td>SD137</td></tr>
<tr><td>SD150</td><td rowspan="8">I/O module verification error</td><td rowspan="8">The bit pattern (16 Bit) indicates the modules with verification errors.
0 : No I/O verification error
1 : I/O verification error present</td><td rowspan="8">When the power is turned on, the module numbers of the I/O modules whose information differs from the registered I/O module information are set in this register (in units of 16 points).

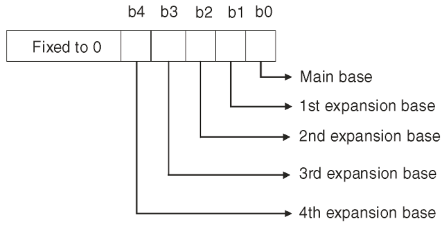
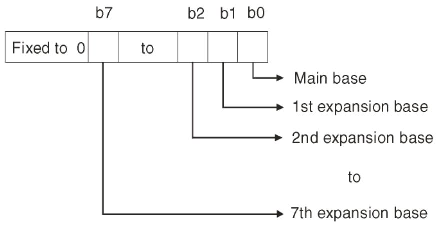
I/O module information is also detected.
.
<table><tr><th></th><th>b15</th><th>b14</th><th>b13</th><th>b12</th><th>b11</th><th>b10</th><th>b9</th><th>b8</th><th>b7</th><th>b6</th><th>b5</th><th>b4</th><th>b3</th><th>b2</th><th>b1</th><th>b0</th></tr><tr><td>SD150</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1 XY80</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1 XY0</td></tr><tr><td>SD151</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1 XY190</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>SD157</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1 XYFB0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>Verification error for the module with the head I/O number X/YFB0.</td><td rowspan="8">S
(Error)</td><td rowspan="8">New</td></tr>
<tr><td>SD151</td></tr>
<tr><td>SD152</td></tr>
<tr><td>SD153</td></tr>
<tr><td>SD154</td></tr>
<tr><td>SD155</td></tr>
<tr><td>SD156</td></tr>
<tr><td>SD157</td></tr>
</table>

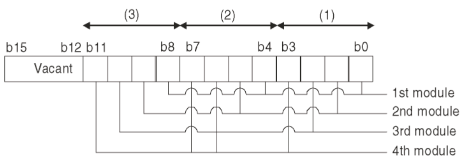
Number	Name	Meaning	Description	Set by (if set)	ACPU register D9 [ ] [ ] [ ]	Valid for:
SD200	State of switch	State of CPU switch	The status of the remote I/O module is stored in the following format: b15 ... b4 b3 ... b0 Vacant (b15–b4), (1) (b3–b0) (1) Remote I/O module switch status: Always 1: STOP	S (Continous)	New	Remote
			The CPU switch state is stored in the following format: b15 ... b8 b7 ... b4 b3 ... b0 Vacant, (2), (1) (1) CPU switch status: (0): RUN; (1): STOP (2) Memory card switch: Always OFF	S (Every END processing)	New	Q00JCPU Q00CPU Q01CPU
			The CPU switch state is stored in the following format: bF ... bBbA ... b8 b7 ... b4 b3 ... b0 (3), Free, (2), (1) (1) CPU switch status: (0): RUN; (1): STOP; (2): L.CLR (2) Memory card switch: Always OFF (3) DIP-Switch: b8 to bC correspond to SW1 through SW5 of system setting switch 1 . 0: OFF, 1: ON. bD,bE and bF are vacant	S (Every END processing)	New	Q CPU (except Q00J, Q00 and Q01CPU)
			The CPU switch state is stored in the following format: b15 ... b12 b11 ... b8 b7 ... b4 b3 ... b0 (3), Free, (2), (1) (1) : CPU Status: (0) : RUN; (1) : STOP; (2) : L.CLR (2) : Memory card switch: B4 corresponds to card A, B5 corresponds to card B. OFF for 0; ON for 1 (3) : DIP switch: B8 to B15 correspond to SW1 to SW8. OFF for 0; ON for 1	S (Every END processing)	New	QnA CPU

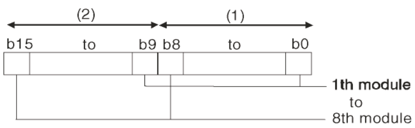
Number	Name	Meaning	Description	Set by (if set)	ACPU register D9 [ ] [ ] [ ]	Valid for:
SD201	LED status	State of CPU-LED	The following bit patterns are used to store the statuses of the LEDs of the CPU:  (1) : RUN (5) : BOOT (2) : ERROR (6) : Vacant (3) : USER (7) : Vacant (4) : BAT.ALARM (8) : MODE Bitpatterns for MODE 0: OFF 1: Green 2: Orange The areas 3 to 8 are not available for a Q00JCPU, Q00CPU or Q01CPU.	S (Status change)	New	System Q CPU
			Information concerning which of the following states the LEDs on the CPU are stored in the following bit patterns: 0 is off, 1 is on, and 2 is flicker  (1) : RUN (5) : BOOT (2) : ERROR (6) : Card A (memory card) (3) : USER (7) : Card B (memory card) (4) : BAT.ALARM (8) : Vacant	S (Status change)	New	QnA CPU
SD202	LED off	Bit pattern of LED that is turned off	Stored bit patterns of LEDs turned off (Only USER and BOOT enabled) Turned off at 1, not turned off at 0	U	New	QnA CPU
SD203	Operating state of CPU	Operating state of CPU	The operating status of the remote I/O module is stored in the following format:  (1) Remote I/O module operating status Always 2: STOP	S (Continous)	New	Remote
			The CPU operating state is stored as indicated in the following figure:  (1) : Operating state of CPU0 : RUN 1 : STEP-RUN 2 : STOP 3 : PAUSE (2) : STOP/PAUSE cause 0 : Key switch 1 : Remote contact 2 : Peripheral, computer link, or operation from some other remote source 3 : Internal program instruction 4 : Error Remark: Only the error that occurred first is stored.	S (Every END processing)	D9015 (format change)	●

Number	Name	Meaning	Description	Set by (if set)	ACPU register D9 [ ] [ ] [ ]	Valid for:
SD206	Device test execution type	Indicates the kind of device test	When a device test is being executed by a programming device, the contents of this register reflects the state of the test: 0 = Test not yet executed 1 = Test of input devices (X) 2 = Test of output devices (Y) 3 = Test of input and output devices (X/Y)	S (Request)	New	Remote
SD207	LED display priority ranking	Priorities 1 to 4	When error is generated, the LED display (flicker) is made according to the error number setting priorities. The setting areas for priorities are as follows: SD207: Priority 4, Priority 3, Priority 2, Priority 1 (4321H) SD208: Priority 8, Priority 7, Priority 6, Priority 5 (8765 H) SD209: –, –, Priority 10, Priority 9 (00A9 H) No display is made if "0" is set. However, even if "0" has been set, information concerning CPU operation stop (including parameter settings) errors will be indicated by the LEDs without conditions.	U	D9038	● (except Q00J, Q00 and Q01CPU)
SD208		Priorities 5 to 8			D9039 (format change)	
SD209		Priorities 9 to 10			New	
SD210	Clock data	Clock data (year, month)	The year (last two digits) and month are stored as BCD code at SD210 as shown below: b15 b12 b11 b8 b7 b4 b3 b0 Year Month Example: July 1993 = **H9307**	S/U (Request)	D9025	● Rem
SD211	Clock data	Clock data (day, hour)	The day and hour are stored as BCD code at SD211 as shown below: b15 b12 b11 b8 b7 b4 b3 b0 Day Hour Example: 31st, 10 a. m. = **H3110**		D9026	
SD212	Clock data	Clock data (minute, second)	The minutes and seconds (after the hour) are stored as BCD code at SD212 as shown below: b15 b12 b11 b8 b7 b4 b3 b0 Minute Second Example: 35 min, 48 sec. = **H3548**		D9027	

Number	Name	Meaning	Description	Set by (if set)	ACPU register D9 [ ] [ ] [ ]	Valid for:
SD213	Clock data	Clock data (day of the week)	The day of the week is stored as BCD code at SD213 as shown below: b15-------b12 b11-------b8 b7-------b4 b3-------b0 Higher digits of year (0 to 99) Day of week: 0 Sunday, 1 Monday, 2 Tuesday, 3 Wednesday, 4 Thursday, 5 Friday, 6 Saturday	S/U (Request)	D9028	Q CPU Rem
			The day of the week is stored as BCD code at SD213 as shown below: b15-------b12 b11-------b8 b7-------b4 b3-------b0 Always "0" Day of week: 0 Sunday, 1 Monday, 2 Tuesday, 3 Wednesday, 4 Thursday, 5 Friday, 6 Saturday	S/U (Request)		QnA CPU
SD220 SD221 SD222 SD223 SD224 SD226 SD227	LED display data	Display indicator data	LED display ASCII data (16 characters) stored here. b15 to b8 / b7 to b0 SD220: 15th character from the right / 16th character from the right SD221: 13th character from the right / 14th character from the right SD222: 11th character from the right / 12th character from the right SD223: 9th character from the right / 10th character from the right SD224: 7th character from the right / 8th character from the right SD225: 5th character from the right / 6th character from the right SD226: 3rd character from the right / 4th character from the right SD227: 1st character from the right / 2nd character from the right	S (Status change)	New	●
SD240	Base mode	0: Automatic mode 1: Detail mode	Stores the base mode	S (Initial)	New	Q CPU Rem
SD241	Number of extension bases	0: Basic only 1 to 7: Number of extension bases	Stores the number of extension bases being installed	S (Initial)	New	

Number	Name	Meaning	Description	Set by (if set)	ACPU register D9 [ ] [ ] [ ]	Valid for:
SD242	A/Q base differentiation	0: QA[ ] [ ]B is installed (A mode) 1: Q[ ] [ ]B is installed (Q mode)	 When no expansion base is installed, the value for b1 to b4 is fixed to "0".	S (Initial)	New	Q00JCPU Q00CPU Q01CPU
			 When no expansion base is installed, the value for b1 to b7 is fixed to "0".			System Q CPU (except Q00JCPU Q00CPU Q01CPU)
SD243 SD244	Number of base slots	Number of base slots The areas for the 5th to 7th expansion base are fixed to "0" for a Q00JCPU, Q00CPU or Q01CPU	bF bC bB b8 b7 b4 b3 b0 SM243: 3rd ext. / 2nd ext. / 1th ext. / Basic SM244: 7th ext / 6th ext. / 5th ext. / 4th ext. The number of slots being installed is stored in the respective areas for the basic base and the extension bases (ext.).	S (Initial)	New	System Q CPU
SD250	Loaded maximum I/O	Loaded maximum I/O No.	When SM250 goes from OFF to ON, the upper 2 digits of the final I/O number plus 1 of the modules loaded are stored as BIN values.	S (Request END)	New	●
SD251	Head I/O No. for replacement	Head I/O number for module replacement	Stores upper two digits of the first I/O number of an I/O module that is removed/replaced in the online status.	U	D9094	Q2A (S1) Q3A Q4A Q4AR
SD253	RS422 baud rate	RS422 baud rate	Stores the baud rate of RS422: 0: 9600 bps, 1: 19,2 bps, 2: 38,4 bps	S (When changed)	New	QnA CPU

<table>
<tr><th>Number</th><th>Name</th><th colspan="2">Meaning</th><th>Description</th><th>Set by (if set)</th><th>ACPU register D9 [ ] [ ] [ ]</th><th>Valid for:</th></tr>
<tr><td>SD254</td><td rowspan="9">MELSECNET/10 information</td><td colspan="2">Number of modules installed</td><td>Indicates the number of modules installed on NET/10</td><td rowspan="9">S (Initial)</td><td rowspan="9">New</td><td rowspan="5">●</td></tr>
<tr><td>SD255</td><td rowspan="5">Information from 1st module</td><td>I/O No.</td><td>NET/10 I/O number of first module installed</td></tr>
<tr><td>SD256</td><td>Network No.</td><td>NET/10 network number of first module installed</td></tr>
<tr><td>SD257</td><td>Group Number</td><td>NET/10 group number of first module installed</td></tr>
<tr><td>SD258</td><td>Station No.</td><td>NET/10 station number of first module installed</td></tr>
<tr><td>SD259</td><td>Standby information</td><td>In the case of standby stations, the module number of the standby station is stored. (1 to 4)</td><td rowspan="4">● (except Q00JCPU Q00CPU Q01CPU)</td></tr>
<tr><td>SD260 – SD264</td><td colspan="2">Information from 2nd module</td><td>Configuration is identical to that for the first module.</td></tr>
<tr><td>SD265 – SD269</td><td colspan="2">Information from 3rd module</td><td>Configuration is identical to that for the first module.</td></tr>
<tr><td>SD270 – SD274</td><td colspan="2">Information from 4th module</td><td>Configuration is identical to that for the first module.</td></tr>
<tr><td rowspan="2">SD280</td><td rowspan="2">CC-Link error</td><td colspan="2" rowspan="2">Error detection status</td><td>

(1) When Xn0 of the installed CC-Link goes ON, the bit corresponding to the station switches ON.
(2) When either Xn1 or XnF of the installed CC-Link switch OFF, the bit corresponding to the station switches ON.
(3) Switches ON when the CPU cannot communicate with the installed CC-Link.</td><td>S (error)</td><td>New</td><td>Q CPU</td></tr>
<tr><td>

(1) When Xn0 of the installed CC-Link goes ON, the bit corresponding to the station switches ON.
(2) When either Xn1 or XnF of the installed CC-Link switch OFF, the bit corresponding to the station switches ON.</td><td>S (error)</td><td>New</td><td>QnA</td></tr>
</table>

Number	Name	Meaning	Description	Set by (if set)	ACPU register D9 [ ] [ ] [ ]	Valid for:
SD290	Device allocation (Same as parameter contents)	Number of points allocated for X	Stores the number of points currently set for X	S (Initial)	New	● Rem
SD291		Number of points allocated for Y	Stores the number of points currently set for Y			
SD292		Number of points allocated for M	Stores the number of points currently set for M			
SD293		Number of points allocated for L	Stores the number of points currently set for L			
SD294		Number of points allocated for B	Stores the number of points currently set for B			● Rem
SD295		Number of points allocated for F	Stores the number of points currently set for F			●
SD296		Number of points allocated for SB	Stores the number of points currently set for SB			● Rem
SD297		Number of points allocated for V	Stores the number of points currently set for V			●
SD298		Number of points allocated for S	Stores the number of points currently set for S			
SD299		Number of points allocated for T	Stores the number of points currently set for T			
SD300		Number of points allocated for ST	Stores the number of points currently set for ST			
SD301		Number of points allocated for C	Stores the number of points currently set for C			
SD302		Number of points allocated for D	Stores the number of points currently set for D			● Rem
SD303	Device allocation (Same as parameter contents)	Number of points allocated for W	Stores the number of points currently set for W			
SD304		Number of points allocated for SW	Stores the number of points currently set for SW			
SD315	Time reserved for communication processing	Time reserved for communication processing	Reserves the designated time for communication processing with the GX developer or other units. The greater the value is designated, the shorter the response time for communication with other devices (GX Developer, serial communication units becomes. Setting range: 1 to 100 ms. If the specified value is out of range, it is assumed to no setting. The scan time becomes longer by the specified time.	END processing	New	System Q CPU

System Clocks / Counters

Number	Name	Meaning	Description	Set by (if set)	ACPU register D9 [ ] [ ] [ ]	Valid for:
SD412	1 second counter	Number of counts in 1-second units	Following programmable controller CPU RUN, 1 is added each second. Count repeats from 0 to 32767 to -32768 to 0	S (Status change)	D9022	
SD414	n = 1 second steps	2n second clock units	Stores value n of 2n second clock (Default is 30). Setting can be made between 1 and 32767.	U	New	
SD415	n = 1 ms steps	2n ms clock units	Stores value n of 2n ms clock (Default is 30). Setting can be made between 1 and 32767.	U	New	System Q CPU (except Q00JCPU Q00CPU Q01CPU)
SD420	Scan counter	Number of counts in each scan	Incremented by 1 for each scan execution after the PC CPU is set to RUN. Count repeats from 0 to 32767 to -32768 to 0.	S (Every END processing)	New	
SD430	Low speed scan counter	Number of counts in each scan	Incremented by 1 for each scan execution after the PC CPU is set to RUN. Count repeats from 0 to 32767 to -32768 to 0. Used only for low speed execution type programs.	S (Every END processing)	New	(except Q00JCPU Q00CPU Q01CPU)

A.3.1 Scan Information

Number	Name	Meaning	Description	Set by (if set)	ACPU register D9 [ ] [ ] [ ]	Valid for:
SD500	Execution program No.	Execution type of program being executed	Program number of program currently being executed is stored as BIN value.	S (Status change)	New	● (except Q00JCPU Q00CPU Q01CPU)
SD510	Low speed program No.	File name of low speed execution in progress	Program number of low speed program currently being executed is stored as BIN value. Enabled only when SM510 is ON.	S (Every END processing)	New	
SD520	Current scan time	Current scan time (in 1 ms units)	Stores current scan time (in 1 ms units) Range from 0 to 65535	S (Every END processing)	D9017 (format change)	●
SD521		Current scan time (in 1 s units)	Stores current scan time (in 1 s units) Range from 00000 to 900 (Example) A current scan of 23.6 ms would be stored as follows: D520 = 23 D521 = 600		New	
SD522	Initial scan time	Initial scan time (in 1 ms units)	Stores scan time for first scan (in 1 ms units). Range from 0 to 65535	S (First END processing)	New	● (except Q00JCPU Q00CPU Q01CPU)
SD523		Initial scan time (in 100 s units)	Stores scan time for first scan (in 1 s units). Range of 000 to 900			
SD524	Minimum scan time	Minimum scan time (in 1 ms units)	Stores minimum value of scan time (in 1 ms units). Range from 0 to 65535	S (Every END processing)	D9018 (format change)	●
SD525		Minimum scan time (in 100 s units)	Stores minimum value of scan time (in 100 s units). Range of 000 to 900		New	
SD526	Maximum scan time	Maximum scan time (in 1 ms units)	Stores meximum value of scan time, excepting the first scan. (in 1 ms units). Range from 0 to 65535	S (Every END processing)	D9019 (format change)	
SD527		Maximum scan time (in 100 s units)	Stores maximum value of scan time, excepting the first scan. (in 100 s units). Range of 000 to 900		New	
SD528	For low speed execution type programs current scan time	Current scan time (in 1 ms units)	Stores current scan time for low speed execution type program (in 1 ms units).	S (Every END processing)	New	● (except Q00JCPU Q00CPU Q01CPU)
SD529		Current scan time (in 100 s units)	Stores current scan time for low speed execution type program (in 100 s units). Range of 000 to 900			
SD532	Minimum scan time for low speed execution type programs	Minimum scan time (in 1 ms units)	Stores minimum value of scan time for low speed execution type program (in 1 ms units). Range from 0 to 65535	S (Every END processing)	New	
SD533		Minimum scan time (in 100 s units)	Stores minimum value of scan time for low speed execution type program (in 100 s units). Range of 000 to 900			
SD534	Maximum scan time for low speed execution type programs	Maximum scan time (in 1 ms units)	Stores the maximum scan time for all except low speed execution type program s first scan (in 1 ms units). Range from 0 to 65535	S (Every END processing)	New	
SD535		Maximum scan time (in 100 s units)	Stores the maximum scan time for all except low speed execution type program s first scan (in 100 s units). Range of 000 to 900			

Scan Information (continued)

Number	Name	Meaning	Description	Set by (if set)	ACPU register D9 [ ] [ ] [ ]	Valid for:
SD540	END processing time	END processing time (in 1 ms units)	Stores time from completion of scan program to start of next scan (in 1 ms units). Range from 0 to 65535	S (Every END processing)	New	●
SD541		END processing time (in 100 s units)	Stores time from completion of scan program to start of next scan (in 100 s units). Range of 000 to 900			
SD542	Constant scan wait time	Constant scan wait time (in 1 ms units)	Stores wait time when constant scan time has been set (in 1 ms units). Range from 0 to 65535	S (First END processing)	New	
SD543		Constant scan wait time (in 100 s units)	Stores wait time when constant scan time has been set (in 100 s units). Range of 000 to 900			
SD544	Cumulative execution time for low speed execution type programs	Cumulative execution time for low speed execution type programs (in 1 ms units)	Stores cumulative execution time for low speed execution type programs (in 1 ms units). Range from 0 to 65535 Cleared to 0 following 1 low speed scan	S (Every END processing)	New	● (except Q00JCPU Q00CPU Q01CPU)
SD545		Cumulative execution time for low speed execution type programs (in 100 s units)	Stores cumulative execution time for low speed execution type programs (in 100 s units). Range of 000 to 900 Cleared to 0 following 1 low speed scan			
SD546	Execution time for low speed execution type programs	Execution time for low speed execution type programs (in 1 ms units)	Stores low speed program execution time during 1 scan (in 1 ms units). Range from 0 to 65535 Stores each scan	S (Every END processing)	New	
SD547		Execution time for low speed execution type programs (in 100 s units)	Stores low speed program execution time during 1 scan (in 100 s units). Range of 000 to 900 Stores each scan			
SD548	Scan program execution time	Scan program execution time (in 1 ms units)	Stores execution time for scan execution type program during 1 scan (in 1 ms units). Range from 0 to 65535 Stores each scan	S (Every END processing)	New	●
SD549		Scan program execution time (in 100 s units)	Stores execution time for scan execution type program during 1 scan (in 100 s units). Range of 000 to 900 Stores each scan			
SD550	Service interval measurement module	Unit/module No.	Sets I/O number for module that measures service interval.	U	New	● (except Q00JCPU Q00CPU Q01CPU)
SD551	Service interval time	Module service interval (in 1 ms units)	When SM 551 is ON, stores service interval for module designated by SD 550 (in 1 ms units). Range from 0 to 65535	S (Request)	New	
SD552		Module service interval (in 100 s units)	When SM551 is ON, stores service interval for module designated by SD550 (in 1 s units). Range from 000 to 999			

Memory Cards

<table>
<tr><th>Number</th><th>Name</th><th>Meaning</th><th>Description</th><th>Set by (if set)</th><th>ACPU register D9 [ ] [ ] [ ]</th><th>Valid for:</th></tr>
<tr><td rowspan="2">SD600</td><td rowspan="2">Memory card A models</td><td rowspan="2">Memory card A models</td><td>Indicates memory card A model installed.
bF b8 b7 b4 b3 b0
0 ◄ ► 0
Drive 1 (RAM): 0: Does not exist, 1: SRAM
Drive 2 (ROM): 0: Does not exist, (1: SRAM), 2: ATA FLASH, 3: FLASH ROM</td><td>S
(Initial and card removal)</td><td>New</td><td>System Q CPU (except Q00JCPU Q00CPU Q01CPU)</td></tr>
<tr><td>Indicates memory card A model installed.
b15 b8 b7 b4 b3 b0
0 ◄ ► 0
Drive 1 (RAM): 0: Does not exist, 1: SRAM
Drive 2 (ROM): 0: Does not exist, 2: EEPROM, 3: FLASH ROM</td><td>S
(Initial and card removal)</td><td>New</td><td>QnA CPU</td></tr>
<tr><td>SD602</td><td>Drive 1 (RAM) capacity</td><td>Drive 1 capacity</td><td>Drive 1 capacity is stored in 1 k byte units</td><td>S
(Initial and card removal)</td><td>New</td><td rowspan="2">●
(except Q00JCPU Q00CPU Q01CPU)</td></tr>
<tr><td>SD603</td><td>Drive 2 (ROM) capacity</td><td>Drive 2 capacity</td><td>Drive 2 capacity is stored in 1 k byte units</td><td>S
(Initial and card removal)</td><td>New</td></tr>
<tr><td rowspan="2">SD604</td><td rowspan="2">Memory card A use conditions</td><td rowspan="2">Memory card A use conditions</td><td>The use conditions for memory card A are stored as bit patterns (in use when ON).
The significance of these bit patterns is indicated below:
b0 : BOOT operation (QBT)
b1 : Parameters (QPT)
b2 : Device comments (QCD)
b3 : Device initital value (QDI)
b4 : File Register (QDR)
b5 : Trace (QTS)
b6 :
b7 :
b8 :
b9 : CPU fault history (QFD)
bA : SFC trace (QTS)
bB : Local device (QDL)
bC :
bD :
bE :
bF :</td><td>S
(Status change)</td><td>New</td><td>System Q CPU (except Q00JCPU Q00CPU Q01CPU)</td></tr>
<tr><td>The use conditions for memory card A are stored as bit patterns (in use when ON).
The significance of these bit patterns is indicated below:
b0 : BOOT operation (QBT)
b1 : Parameters (QPT)
b2 : Device comments (QCD)
b3 : Device initital value (QDI)
b4 : File Register (QDR)
b5 : Sampling trace (QTS)
b6 : Status latch (QTL)
b7 : Program trace (QTP)
b8 : Simulation data (QDS)
b9 : CPU fault history (QFD)
bA : SFC trace (QTS)
bB : Local device (QDL)
bC :
bD :
bE :
bF :</td><td>S
(Status change)</td><td>New</td><td>QnA CPU</td></tr>
</table>

Memory Cards

Number	Name	Meaning	Description	Set by (if set)	ACPU register D9 [ ] [ ] [ ]	Valid for:
SD620	Memory card B models	Memory card B models	Indicates memory card B models installed bF 0 ◄──► 0 b8 b7 b4 b3 b0 Drive 3 (RAM): 0: Does not exist, 1: SRAM Drive 4 (ROM): 0: Does not exist, (1: SRAM), 2: ATA FLASH, 3: FLASH ROM The value for drive 4 is fixed to "3" because it has built-in FLASH ROM.	S (Initial)	New	System Q CPU
			Indicates memory card B models installed b15 0 ◄──► 0 b8 b7 b4 b3 b0 Drive 1 (RAM): 0: Does not exist, 1: SRAM Drive 2 (ROM): 0: Does not exist, 2: EEPROM, 3: FLASH ROM	S (Initial)	New	QnA CPU
SD622	Drive 3 (RAM) capacity	Drive 3 capacity	Drive 3 capacity is stored in 1k byte units With a Q CPU, this value is fixed to "61" because of the built-in 61k RAM.	S (Initial)	New	System Q CPU
			Drive 3 capacity is stored in 1k byte units	S (Initial)	New	Q2(S1) Q3A Q4A Q4AR CPU
SD623	Drive 4 (ROM) capacity	Drive 4 capacity	Drive 4 capacity is stored in 1k byte units	S (Initial)	New	Q2(S1) Q3A Q4A Q4AR System Q CPU
SD624	Drive 3 use conditions	Drive 3 use conditions	The use condition of drive 3 is indicated by bit 4: b4 = OFF: Drive 3 is not used b4 = ON: Drive 3 is used to store file registers	S (Status change)	New	Q00JCPU Q00CPU Q01CPU
	Drive 3 and 4 use conditions	Drive 3 and 4 use conditions	The use conditions for memory card B are stored as bit patterns (In use when ON) The significance of these bit patterns is indicated below: b0 : BOOT operation (QBT) b1 : Parameters (QPA) b2 : Device comments (QCD) b3 : Device initital value (QDI) b4 : File R (QDR) b5 : Trace (QTS) b6 : b7 : b8 : b9 : CPU fault history (QFD) bA : SFC trace (QTS) bB : Local device (QDL) bC : bD : bE : bF :	S (Status change)	New	System Q CPU (except Q00JCPU Q00CPU Q01CPU)
	Memory card B use conditions	Memory card B use conditions	The use conditions for memory card B are stored as bit patterns (In use when ON) The significance of these bit patterns is indicated below: b0 : BOOT operation (QBT) b1 : Parameters (QPT) b2 : Device comments (QCD) b3 : Device initital value (QDI) b4 : File Register (QDR) b5 : Sampling trace (QTS) b6 : Status latch (QTL) b7 : Program trace (QTP) b8 : Simulation data (QDS) b9 : CPU fault history (QFD) bA : SFC trace (QTS) bB : Local device (QDL) bC : bD : bE : bF :	S (Status change)	New	Q2(S1) Q3A Q4A Q4AR CPU

File Register Information

Number	Name	Meaning	Description	Set by (if set)	ACPU register D9 [ ] [ ] [ ]	Valid for:
SD640	File register drive	Drive number	Stores drive number being used by file register	S (Status change)	New	●
SD641 SD642 SD643 SD644 SD645 SD646	File register file name	File register file name	Stores file register file name (with extension) selected at parameters or by use of QDRSET instruction as ASCII code. b15 … b8 / b7 … b0 SD641: 2nd character / 1st character SD642: 4th character / 3rd character SD643: 6th character / 5th character SD644: 8th character / 7th character SD645: 1st char. of extension / 2EH (.) SD646: 3rd char. of extension / 2nd char. of extension	S (Status change)	New	
SD647	File register capacity	File register capacity	Stores the data capacity of the currently selected file register in 1 K word units.	S (Status change)	New	
SD648	File register block number	File register block number	Stores the currently selected file register block number.	S (Status change)	D9035	
SD650	Comment drive	Comment drive	Stores the comment drive number selected at the parameters or by the QCDSET instruction.	S (Status change)	New	● (except Q00JCPU Q00CPU Q01CPU)
SD651 SD652 SD653 SD654 SD655 SD656	Comment file name	Comment file name	Stores the comment file name (with extension) selected at the parameters or by the QCDSET instruction in ASCII code. b15 … b8 / b7 … b0 SD651: 2nd character / 1st character SD652: 4th character / 3rd character SD653: 6th character / 5th character SD654: 8th character / 7th character SD655: 1st char. of extension / 2EH (.) SD656: 3rd char. of extension / 2nd char. of extension	S (Status change)	New	
SD660	Boot operation designation file	Boot designation file drive number	Stores the drive number where the boot designation file (*.QBT) is being stored.	S (Initial)	New	
SD661 SD662 SD663 SD664 SD665 SD666		File name of boot designation file	Stores the file name of the boot designation file (*.QBT). b15 … b8 / b7 … b0 SD661: 2nd character / 1st character SD662: 4th character / 3rd character SD663: 6th character / 5th character SD664: 8th character / 7th character SD665: 1st char. of extension / 2EH (.) SD666: 3rd char. of extension / 2nd char. of extension	S (Initial)	New	

Instruction related registers

Number	Name	Meaning	Description	Set by (if set)	ACPU register D9 [ ] [ ] [ ]	Valid for:
SD705 SD706	Mask pattern	Mask pattern	During block operations, turning SM705 ON makes it possible to use the mask pattern being stored at SD705 (or at SD705 and SD706 if double words are being used) to operate on all data in the block with the masked values.	U	New	● (except Q00JCPU Q00CPU Q01CPU)
SD714	Number of vacant com-munication request regi-stration areas	0 to 32	Stores the number of vacant blocks in the communications request area for remote terminal modules connected to the AJ71PT32-S3.	S (During execution)	M9081	QnA CPU
SD715 SD716 SD717	IMASK instruction mask pattern	Mask pattern	Patterns masked by use of the IMASK instruction are stored in the following manner: b15 .... b0 SD715: I15 I14 I13 I12 I11 I10 I9 I8 I7 I6 I5 I4 I3 I2 I1 I0 SD716: I31 I30 I29 I28 I27 I26 I25 I24 I23 I22 I21 I20 I19 I18 I17 I16 SD717: I47 I46 I45 I44 I43 I42 I41 I40 I39 I38 I37 I36 I35 I34 I33 I32	S (During execution)	New	●
SD718 SD719	Accumulator	Accumulator	For use as replacement for accumulators used in A-series programs.	S/U	New	
SD720	Program No. destination for PLOAD instruction	Program number destination for PLOAD instruction	Stores the program number of the program to be loaded by the PLOAD instruction when designated. The destination range is from 1 to 124.	U	New	System Q CPU
SD730	No. of vacant registration area for CC-Link communi-cation request	0 to 32	Stores the number of vacant registration areas for the request for communication with the intelligent device station connected to A(1S)J61QBT61.	S (During execution)	New	QnA CPU
SD736	PKEY input	PKEY input	SD that temporarily stores keyboard data input by means of the PKEY instruction.	S (During execution)	New	● (except Q00JCPU Q00CPU Q01CPU)

Index

I

L

M

N

O

P

Global Partner. Local Friend.

EUROPE
MITSUBISHI ELECTRIC
EUROPE B.V.
German Branch
Gothaer Straße 8
D-40880 Ratingen
Phone: +49 (0) 2102 / 486-0
Fax: +49 (0) 2102 / 486-1120
e mail: megfamail@meg.mee.com

FRANCE
MITSUBISHI ELECTRIC EUROPE B.V.
French Branch
25, Boulevard des Bouvets
F-92741 Nanterre Cedex
Phone: +33 1 55 68 55 68
Fax: +33 1 55 68 56 85
email: factory.automation@fra.mee.com

IRELAND
MITSUBISHI ELECTRIC EUROPE B.V.
Irish Branch
Westgate Business Park, Ballymount
IRL-Dublin 24
Phone: +353 (0) 1 / 419 88 00
Fax: +353 (0) 1 / 419 88 90
e mail: sales.info@meir.mee.com

ITALY
MITSUBISHI ELECTRIC EUROPE B.V.
Italian Branch
Via Paracelso 12
I-20041 Agrate Brianza (MI)
Phone: +39 039 6053 1
Fax: +39 039 6053 312
e mail: factory.automation@it.mee.com

SPAIN
MITSUBISHI ELECTRIC EUROPE B.V.
Spanish Branch
Carretera de Rubí 76-80
E-08190 Sant Cugat del Vallés
Phone: +34 9 3 / 565 3160
Fax: +34 9 3 / 589 1579
e mail: industrial@sp.mee.com

UK
MITSUBISHI ELECTRIC EUROPE B.V.
UK Branch
Travellers Lane
GB-Hatfield Herts. AL10 8 XB
Phone: +44 (0) 1707 / 27 61 00
Fax: +44 (0) 1707 / 27 86 95
e mail: automation@meuk.mee.com

JAPAN
MITSUBISHI ELECTRIC CORPORATION
Office Tower "Z" 14 F
8-12,1 chome, Harumi Chuo-Ku
Tokyo 104-6212
Phone: +81 3 6221 6060
Fax: +81 3 6221 6075

USA
MITSUBISHI ELECTRIC AUTOMATION
500 Corporate Woods Parkway
Vernon Hills, IL 60061
Phone: +1 847 / 478 21 00
Fax: +1 847 / 478 22 83

AUSTRIA
GEVA
Wiener Straße 89
AT-2500 Baden
Phone: +43 (0) 2252 / 85 55 20
Fax: +43 (0) 2252 / 488 60
e mail: office@geva.at

BELARUS
TEHNIKON
Oktjabrskaya 16/5, Ap 704
BY-220030 Minsk
Phone: +375 (0)17 / 210 4626
Fax: +375 (0)17 / 210 4626
e mail: tehnikon@belsonet.net

BELGIUM
Koning & Hartman B.V.
Researchpark Zellik, Pontbeeklaan 43
BE-1731 Brussels
Phone: +32 (0)2 / 467 17 51
Fax: +32 (0)2 / 467 17 45
e mail: info@koningenhartman.com

BULGARIA
AKHNATON
Andrej Ljapchev Lbvd. Pb 21 4
BG-1756 Sofia
Phone: +359 (0) 2 / 97 44 05 8
Fax: +359 (0) 2 / 97 44 06 1
e mail: —

CZECH REPUBLIC
AutoCont
Control Systems s.r.o.
Nemocnicni 12
CZ-702 00 Ostrava 2
Phone: +420 59 / 6152 111
Fax: +420 59 / 6152 562
e mail: consys@autocont.cz

DENMARK
louis poulsen
industri & automation
Geminivej 32
DK-2670 Greve
Phone: +45 (0) 70 / 10 15 35
Fax: +45 (0) 43 / 95 95 91
e mail: lpia@lpmail.com

ESTONIA
UTU Elektrotehnika AS
Pärnu mnt.160i
EE-11317 Tallinn
Phone: +372 (0) 6 / 51 72 80
Fax: +372 (0) 6 / 51 72 88
e mail: utu@utu.ee

FINLAND
Beijer Electronics OY
Ansatie 6a
FIN-01740 Vantaa
Phone: +358 (0) 9 / 886 77 500
Fax: +358 (0) 9 / 886 77 555
e mail: info@beijer.fi

GREECE
UTECO A.B.E.E.
5, Mavrogenous Str.
GR-18542 Piraeus
Phone: +302 (0) 10 / 42 10 050
Fax: +302 (0) 10 / 42 12 033
e mail: sales@uteco.gr

HUNGARY
Meltrade Ltd.

Fertö Utca 14.
HU-1107 Budapest
Phone: +36 (0)1 / 431-9726
Fax: +36 (0)1 / 431-9727
e mail: office@meltrade.hu

ISRAEL
TEXEL Electronics Ltd.
Box 6272
IL-42160 Netanya
Phone: +972 (0) 9 / 863 08 91
Fax: +972 (0) 9 / 885 24 30
e mail: texel_me@netvision.net.il

KAZAKHSTAN
Kazpromautomatics Ltd.
2, Scladskaya Str.
KAZ-470046 Karaganda
Phone: +7 3212 50 11 50
Fax: +7 3212 50 11 50
e mail: info@kpakz.com

LATVIA
SIA POWEL
Lienes iela 28
LV-1009 Riga
Phone: +371 784 / 22 80
Fax: +371 784 / 22 81
e mail: utu@utu.lv

LITHUANIA
UAB UTU POWEL
Savanoriu pr. 187
LT-2053 Vilnius
Phone: +370 (0) 52323-101
Fax: +370 (0) 52322-980
e mail: powel@utu.lt

MOLDOVA
INTEHSIS SRL
Cuza-Voda 36/1-81
MD-2061 Chisinau
Phone: +373 (0)2 / 562 263
Fax: +373 (0)2 / 562 263
e mail: intehsis@mdl.net

NETHERLANDS
Koning & Hartman B.V.
Donauweg 2 B
NL-1000 AK Amsterdam
Phone: +31 (0)20 / 587 76 00
Fax: +31 (0)20 / 587 76 05
e mail: info@koningenhartman.com

NORWAY
Beijer Electronics A/S
Teglverksveien 1
N-3002 Drammen
Phone: +47 (0) 32 / 24 30 00
Fax: +47 (0) 32 / 84 85 77
e mail: info@beijer.no

POLAND
MPL Technology Sp. z o.o.
ul. Sliczna 36
PL-31-444 Kraków
Phone: +48 (0) 12 / 632 28 85
Fax: +48 (0) 12 / 632 47 82
e mail: krakow@mpl.pl

ROMANIA
Sirius Trading & Services srl
Str. Biharia No. 67-77
RO-013981 Bucuresti 1
Phone: +40 (0) 21 / 201 1146
Fax: +40 (0) 21 / 201 1148
e mail: sirius@siriustrading.ro

RUSSIA
Avtomatika Sever Ltd.
Lva Tolstogo Str. 7, Off. 311
RU-197376 St Petersburg
Phone: +7 812 1183 238
Fax: +7 812 1183 239
e mail: as@avtsev.spb.ru

RUSSIA
Consys
Promyshlennaya St. 42
RU-198099 St Petersburg
Phone: +7 812 325 3653
Fax: +7 812 147 2055
e mail: consys@consys.spb.ru

RUSSIA
Electrotechnical Systems Siberia
Shetinkina St. 33, Office 116
RU-630088 Novosibirsk
Phone: +7 3832 / 119598
Fax: +7 3832 / 119598
e mail: info@eltechsystems.ru

RUSSIA
Elektrostyle
Poslannikov Per., 9, Str.1
RU-107005 Moscow
Phone: +7 095 542 4323
Fax: +7 095 956 7526
e mail: info@estl.ru

RUSSIA
Elektrostyle
Krasnij Prospekt 220-1, Office No. 312
RU-630049 Novosibirsk
Phone: +7 3832 / 106618
Fax: +7 3832 / 106626
e mail: info@estl.ru

RUSSIA
ICOS
Industrial Computer Systems Zao
Ryazanskij Prospekt, 8A, Off. 100
RU-109428 Moscow
Phone: +7 095 232 0207
Fax: +7 095 232 0327
e mail: mail@icos.ru

RUSSIA
NPP Uralelektra
Sverdlova 11A
RU-620027 Ekaterinburg
Phone: +7 34 32 / 532745
Fax: +7 34 32 / 532745
e mail: elektra@etel.ru

RUSSIA
STC Drive Technique
Poslannikov Per., 9, Str.1
RU-107005 Moscow
Phone: +7 095 790 7210
Fax: +7 095 790 7212
e mail: info@privod.ru

SERBIA AND MONTENEGRO
INEA SR d.o.o.
Karadjordjeva 12/260
SCG-113000 Smederevo
Phone: +381 (0)26/ 617 - 163
Fax: +381 (0)26/ 617 - 163
e mail: inea_sr@verat.net

SLOVAKIA
AutoCont Control s.r.o.
Radlinského 47
SK-02601 Dolný Kubín
Phone: +421 435868 210
Fax: +421 435868 210
e mail: info@autocontcontrol.sk

SLOVENIA
INEA d.o.o.
Stegne 11
SI-1000 Ljubljana
Phone: +386 (0) 1-513 8100
Fax: +386 (0) 1-513 8170
e mail: inea@inea.si

SWEDEN
Beijer Electronics AB
Box 426
S-20124 Malmö
Phone: +46 (0) 40 / 35 86 00
Fax: +46 (0) 40 / 35 86 02
e mail: info@beijer.se

SWITZERLAND
ECONOTEC AG
Postfach 282
CH-8309 Nürensdorf
Phone: +41 (0) 1 / 838 48 11
Fax: +41 (0) 1 / 838 48 12
e mail: info@econotec.ch

SOUTH AFRICA
CBI Ltd.
Private Bag 2016
ZA-1600 Isando
Phone: +27 (0) 11/ 928 2000
Fax: +27 (0) 11/ 392 2354
e mail: cbi@cbi.co.za

TURKEY
GTS
Darülaceze Cad. No. 43 Kat. 2
TR-80270 Okmeydani-Istanbul
Phone: +90 (0) 212 / 320 1640
Fax: +90 (0) 212 / 320 1649
e mail: gts@turk.net

UKRAINE
CSC Automation Ltd.
15, M. Raskova St., Fl. 10, Office 1010
UA-02002 Kiev
Phone: +380 (0) 44 / 494 3355
Fax: +380 (0) 44 / 494 3366
e mail: csc-a@csc-a.kiev.ua

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)2102-486112 /// info@mitsubishi-automation.com /// www.mitsubishi-automation.de
Specifications subject to change /// Art. no. 170295 /// 08.2006