MELSEC System Q

Programmable Logic Controllers

Beginner´s Manual

Art. no.: 209093
29082007
Version A

About This Manual

The texts, illustration, diagrams and examples in this manual are provided for information purposes only. They are intended as aids to help explain the installation, operation, programming and use of the programmable logic controllers of the MELSEC System Q.

If you have any questions about the installation and operation of any of the products described in this manual please contact your local sales office or distributor (see back cover).

You can find the latest information and answers to frequently asked questions on our website at *www.mitsubishi-automation.com*.

Beginner's Manual for the programmable logic controllers of the MELSEC System Q
Art. no.: 209093

Version			Revisions / Additions / Corrections
A	08/2007	pdp-dk	First edition

Safety Guidelines

For use by qualified staff only

This manual is only intended for use by properly trained and qualified electrical technicians who are fully acquainted with the relevant automation technology safety standards. All work with the hardware described, including system design, installation, configuration, maintenance, service and testing of the equipment, may only be performed by trained electrical technicians with approved qualifications who are fully acquainted with all the applicable automation technology safety standards and regulations. Any operations or modifications to the hardware and/or software of our products not specifically described in this manual may only be performed by authorised Mitsubishi Electric staff.

Proper use of the products

The programmable logic controllers of the MELSEC System Q are only intended for the specific applications explicitly described in this manual. All parameters and settings specified in this manual must be observed. The products described have all been designed, manufactured, tested and documented in strict compliance with the relevant safety standards. Unqualified modification of the hardware or software or failure to observe the warnings on the products and in this manual may result in serious personal injury and/or damage to property. Only peripherals and expansion equipment specifically recommended and approved by Mitsubishi Electric may be used with the programmable logic controllers of the MELSEC System Q.

All and any other uses or application of the products shall be 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, configuration, maintenance, servicing and testing of these products. The regulations listed below are particularly important in this regard. This list does not claim to be complete, however; you are responsible for being familiar with and conforming to the regulations applicable to you in your location.

- VDE Standards
  - VDE 0100
    Regulations for the erection of power installations with rated voltages below 1000 V
  - VDE 0105
    Operation of power installations
  - VDE 0113
    Electrical installations with electronic equipment
  - VDE 0160
    Electronic equipment for use in power installations
  - 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 safety regulations

- Accident prevention regulations
  - VBG Nr.4
    Electrical systems and equipment

Safety warnings in this manual

In this manual warnings that are relevant for safety are identified as follows:

DANGER:
Failure to observe the safety warnings identified with this symbol can result in health and injury hazards for the user.

WARNING:
Failure to observe the safety warnings identified with this symbol can result in damage to the equipment or other property.

General safety information and precautions

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

DANGER

- ***Observe all safety and accident prevention regulations applicable to your specific application. Always disconnect all power supplies before performing installation and wiring work or opening any of the assemblies, components and devices.***
- ***Assemblies, components and devices must always be installed in a shockproof housing fitted with a proper cover and fuses or circuit breakers.***
- ***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.***
- ***Take appropriate steps to ensure that cable damage or core breaks in the signal lines cannot cause undefined states in the equipment.***
- ***You are responsible for taking the necessary precautions to ensure that programs interrupted by brownouts and power failures can be restarted properly and safely. In particular, you must ensure that dangerous conditions cannot occur under any circumstances, even for brief periods.***
- ***EMERGENCY OFF facilities conforming to EN 60204/IEC 204 and VDE 0113 must remain fully operative at all times and in all PLC operating modes. The EMERGENCY OFF facility reset function must be designed so that it cannot ever cause an uncontrolled or undefined restart.***
- ***You must implement both hardware and software safety precautions to prevent the possibility of undefined control system states caused by signal line cable or core breaks.***
- ***When using modules always ensure that all electrical and mechanical specifications and requirements are observed exactly.***

Contents

1 Introduction

1.1 About this Manual

This manual will help you to familiarise yourself with the use of the programmable logic controllers of the MELSEC System Q . It is designed for users who do not yet have any experience with programming programmable logic controllers (PLCs).

Programmers who already have experience with PLCs from other manufacturers can also use this manual as a guide for making the transition to the MELSEC System Q.

1.2 More Information

You can find more detailed information on the individual products in the MELSEC System Q in the operating and installation manuals of the individual modules.

See the MELSEC System Q Technical Catalogue, art. no. 136731, for a general overview of all the controllers in the MELSEC System Q. This catalogue also contains information on special function modules and the available accessories.

The communications capabilities using MELSEC or open networks like Ethernet or Profibus are documented in detail in the Technical Catalogue Networks, art. no. 136730.

The Hardware Manuals of the MELSEC System Q support you when designing a controller system and also during installation and start-up of the PLC.

For an introduction to using the programming software package see the GX IEC Developer Beginner's Manual, art. no. 043596 and the Reference Manual, art. no. 043597.

You can find detailed documentation of all programming instructions in the Programming Manual for the MELSEC A/Q Series and the MELSEC System Q, art. no. 87431. Additional program examples are given in almost all manuals for special function modules.

NOTE

All Mitsubishi manuals and catalogues can be downloaded free of charge from the Mitsubishi website at *www.mitsubishi-automation.com*.

2 Programmable Logic Controllers

2.1 What is a PLC?

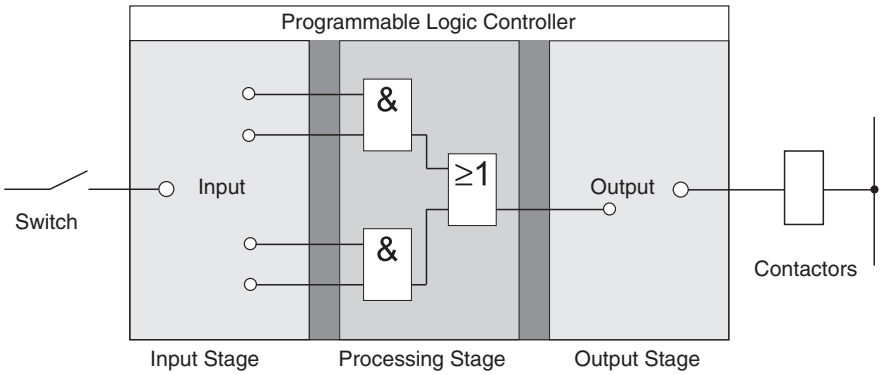
In contrast to conventional controllers with functions determined by their physical wiring the functions of programmable logic controllers or PLCs are defined by a program. PLCs also have to be connected to the outside world with cables, but the contents of their program memory can be changed at any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

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

- an input stage,
- a processing stage

and

- an output stage



The input stage

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

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

The processing stage

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

The output stage

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

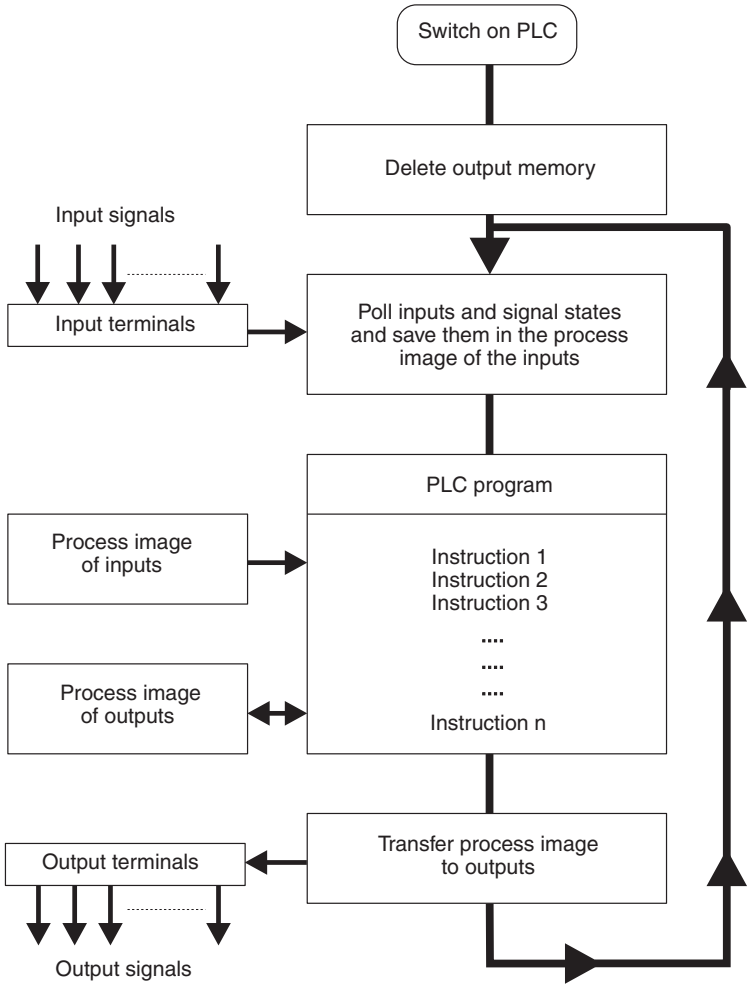
2.2 How PLCs Process Programs

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

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

Process image processing

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



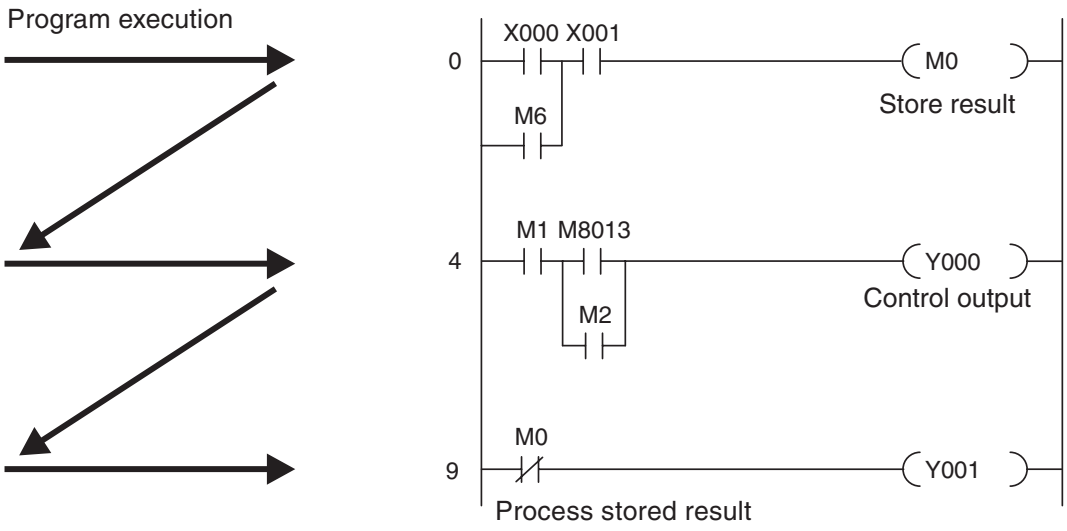
Input process image

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

Program execution

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

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



Output process image

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

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

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

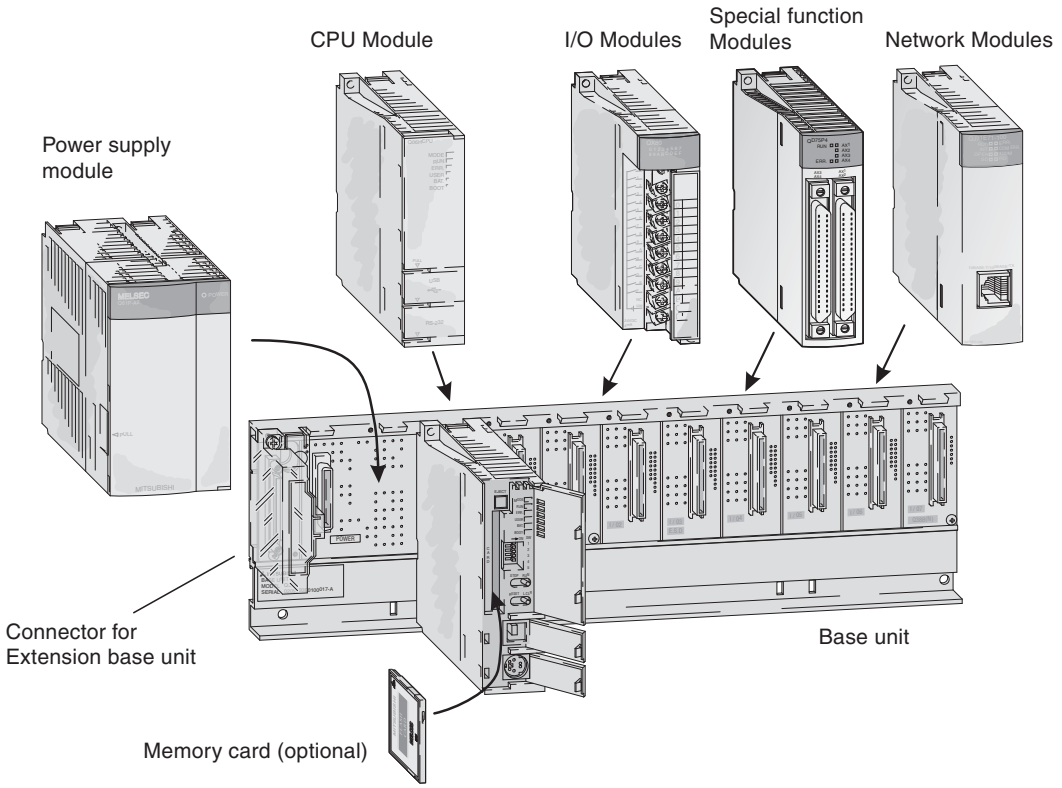
In a PLC it is not possible to respond to changes in input signal states until the next program cycle after the change. Nowadays this disadvantage is largely compensated by very short program cycle periods. The duration of the program cycle period depends on the number and type of instructions executed.

3 The MELSEC System Q

3.1 System Configuration

The MELSEC System Q is a powerful modular PLC with multiprocessor technology . Modular means that the configuration of the system can be adapted to an application individual and optimal.

The heart of a PLC consists of a base unit, a power supply module und at least one CPU module. The CPU executes the instructions in the PLC program. Depending on the application, more modules – for example input and output modules (I/O modules) and special function modules – can be installed to the base unit. The voltage for the installed modules is supplied by the power supply module.

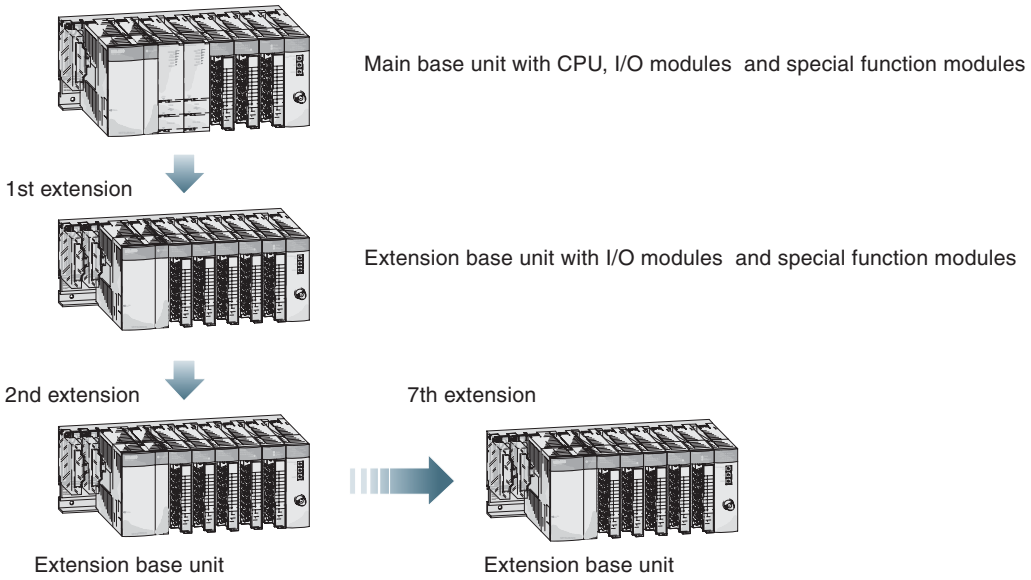


Communication between the individual modules and the CPU is performed via an internal bus connection of the base unit.

The base unit on which the CPU is mounted is called the main base unit. The base units of the MELSEC System Q are available in 5 different versions with up to 12 module slots.

Expandability

When more slots for modules are required, every main base unit can be complemented by extension base units. The base units are simply connected to one another by extension cables. If extension base units without a own power supply modules are used these cables also provide the installed modules with the power. Up to 7 extension base units can be connected to a main base unit. The total number of I/O and special function modules in all base units is 64.

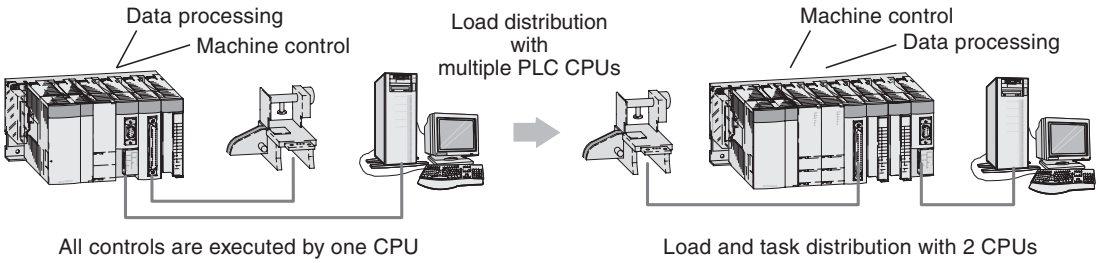


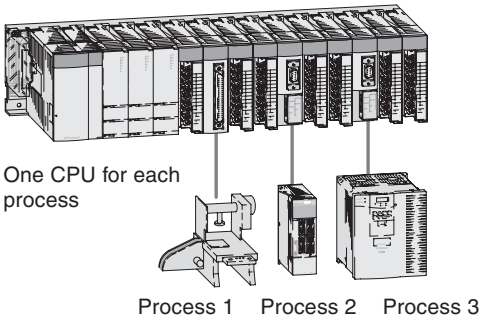
When choosing the power supply module, the total power consumption of the I/O modules, of the special function modules and of the peripherals must be taken into account. If necessary, an extension unit with a further power supply module should be used.

When wiring large plants or for machines with modular configuration, remote inputs and outputs (Remote I/O stations) which are situated on site offer many advantages. Thus the connections between inputs/outputs and sensors/actuators can be kept short. To connect a remote I/O station and the system with the PLC CPU only a network module and a network cable is required. Depending on the selected CPU type up to 4096 local (on main and extension base units) and up to 8192 remote I/O points can be addressed.

Load Distribution with Multiple PLC CPUs

Multiple PLC CPUs of the MELSEC System Q can be used together to allow a single system to exercise controls that are different in tact time, e.g. sequence control and data processing. Thus sequence control and data processing can be distributed to different CPUs.

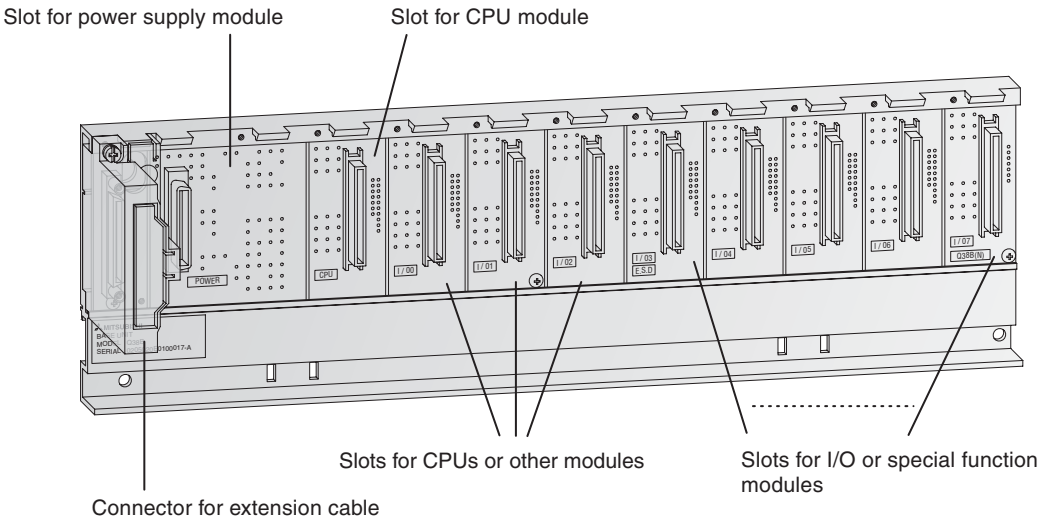




If load in excess of a CPU's processing capability is applied to a large scale system due to a large program size, using multiple CPUs to distribute the load improves the overall performance of the system.

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. On the extension base units, I/O and intelligent function modules can be mounted.The base units can be installed either directly using srews 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 the main base unit Q38RB 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 In the extension base unit Q68RB redundant power supply modules can be used.

3.2.1 Extensions Base Cables

The extension base cables are used for connections between the base units. The overall distance of all extension cables must not exceed 13.2 m.

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

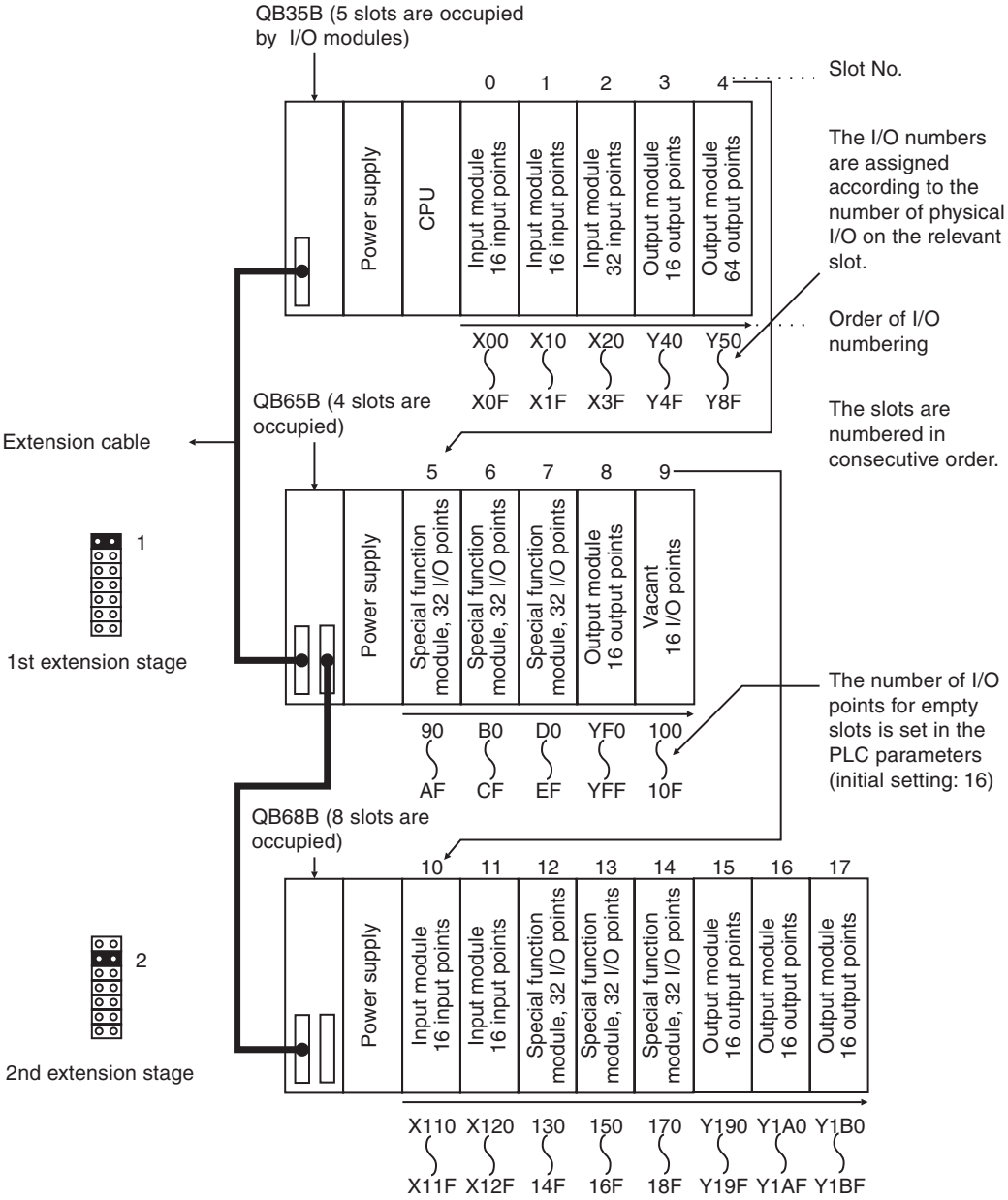
For connection of the base units without an own power supply (Q52B, Q55B) the cable QC05B is recommended.

3.2.2 Allocation of I/O Addresses

To address inputs and outputs of a PLC in the program they must be undisputed labeled. This is done by assigning a number to each input and output: the I/O address (see also chapter 4.1). These addresses are counted in hexadecimal numbers. (Please read more about different number systems later in chapter 4.3.)

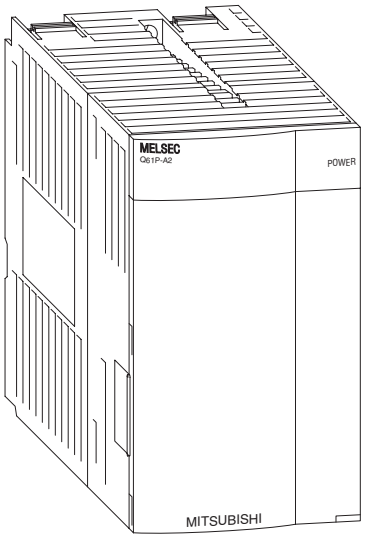
A CPU of the MELSEC System Q automatically recognises the slots available in main and extension base units and assigns addresses to the inputs and outputs accordingly.

However, the assignment can also be done with the aid of the programming software. Thus slots can be left empty or addresses can be reserved for future extensions.



The extension stage is set at the extension main unit with jumpers.

3.3 Power Supply Modules



The MELSEC System Q is powered b a DC voltage of 5 Volts. Power supply modules with input voltages of 24 V DC or 240 V AC are available.

The output voltage of the power supply module (5 V DC) is fed directly into the base unit and is not available at external terminals.

In addition to 5 V DC the power supply module Q62P also provides 24 DC for the supply of peripheral devices such as sensors. This output can be loaded with a maximum current of 0.6 A.

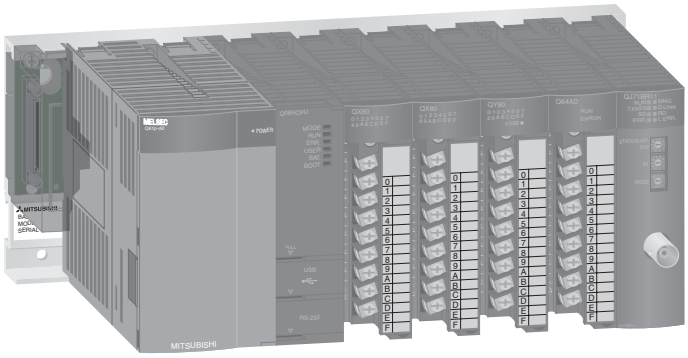
Item	Q63P	Q63RP	Q61P-A1	Q61P-A2	Q62P		Q64P	Q64RP
Input voltage	24 V DC		100–120 V AC	200–220 V AC	100–240 V AC		100–120 V AC 200–240 V AC	
Power consumption	45 W	65 W	105 VA	105 VA	105 VA		105 VA	160 VA
Output voltage	5 V DC		5 V DC		5 V DC	24 V DC	5 V DC	
Output current	6 A	8.5 A	6 A	6 A	3 A	0.6 A	8.5 A	

The power supply modules Q63RP and Q64RP are redundant power supply modules and can be used in combination with all CPUs except the Q00JCPU. For a system with redundant power supply two redundant power supply modules mounted on a redundant base unit are required. Thus the system availability is increased since the other power supply module takes over if one power supply module fails. The redundant power supplies are “hot swappable”, i.e. can be replaced while the system is in operation (replace in RUN mode).

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.

3.4 The CPU Modules

The MELSEC System Q offers 19 different CPU modules and therefore state-of-the-art performance. Up to four CPU modules can be mounted to one base unit und thus distribute control and communication tasks. As with other Mitsubishi controllers the power of the MELSEC System Q grows with your application – you simply replace or add a CPU.

CPU modules can be divided in:

- **PLC CPUs**

  Within the MELSEC System Q a PLC CPU performs the "classical" tasks of a PLC. This CPU executes the PLC program, polls inputs, controls outputs and communicates with special function modules.

- **Process CPUs**

  The process CPU modules of the MELSEC System Q have the functionality of the PLC CPUs and offer additional extended PID functions and integrated process functions with 52 special instructions. Thus this CPUs are suited to complex application e. g. in the chemical industry.

- **Redundant Process CPUs**

  Offering all functions of process CPUs, redundant process CPUs of the MELSEC System Q ensure maximum system availability for critical process and manufacturing automation tasks.

  A redundant setup consists of two identically-configured PLCs (power supply, CPU, network modules etc.) which are conneted by a cable. One PLC controls the process while the other is in “hot standby”. If the active system fails the hot standby system cuts in automatically and takes over, without any interruption. This significantly reduces down time and restart overheads and costs.

- **PC CPU**

  The PC CPU module is a compact personal computer of high value which can be installed on the main base unit. Here the Q-PC masters PC typical applications as well as PLC applications. Therefore, it is suitable as an integrated PC within control systems - e.g. for visualization, data bases, and log-trace functions of the Microsoft application or for programming the System Q in a high-level language. In addition, the system can be controlled as soft PLC according to IEC1131 via the optional SX-Controller software.

  For the connection to the peripherals I/O and special function modules from the MELSEC System Q can be used.

- **C-Controller CPU**

  The C-Controller allows the integration and programming of the automation platform System Q with C++. Using the worldwide established real time operating system VxWorks, realisation of complex tasks, communication and protocols becomes easy.

- **Motion CPUs**

  The motion controller CPU controls and synchronizes the connected servo amplifiers and servo motors. A motion system besides the controller CPU as well includes a PLC CPU. Only after combining a highly dynamic positioning control and a PLC an innovative and autarkical motion control system is created.

  While the Motion CPU controls large-scale servo movements the PLC CPU is responsible for the machine control and the communication at the same time.

In this Beginners Manual only the PLC CPU are described in detail. For information about the other CPU modules please refer to the Technical Catalogue MELSEC System Q, art. No. 136731 and the manuals for the individual modules.

PLC CPUs

- **Q00JCPU**

  CPU, power supply and a 5-slot base unit form an inseparable unit. Multi-CPU operation is not possible with a Q00JCPU.

  – Memory capacity for program: 8 k steps
  – Execution time for a logical instruction: 0.2 µs

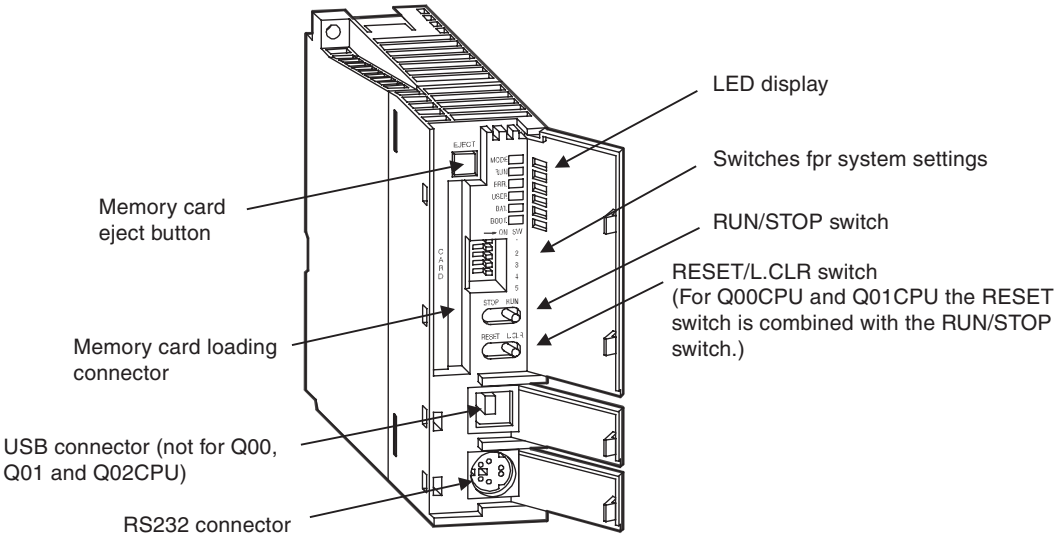
All of the following PLC CPUs are capable for Multi-CPU operation.

- **Q00CPU**
  – Memory capacity for program: 8 k steps
  – Execution time for a logical instruction: 0.16 µs
- **Q01CPU**
  – Memory capacity for program: 14 k steps
  – Execution time for a logical instruction: 0.10 µs
- **Q02CPU**
  – Memory capacity for program: 28 k steps
  – Execution time for a logical instruction: 0.079 µs
- **Q02HCPU**
  – Memory capacity for program: 28 k steps (extendable with memory card)
  – Execution time for a logical instruction: 0.034 µs
- **Q06HCPU**
  – Memory capacity for program: 60 k steps (extendable with memory card)
  – Execution time for a logical instruction: 0.034 µs
- **Q12HCPU**
  – Memory capacity for program: 124 k steps (extendable with memory card)
  – Execution time for a logical instruction: 0.034 µs
- **Q25HCPU**
  – Memory capacity for program: 252 k steps (extendable with memory card)
  – Execution time for a logical instruction: 0.034 µs

The following table shows the extension possibilities and the number of inputs and outputs for the PLC CPUs.

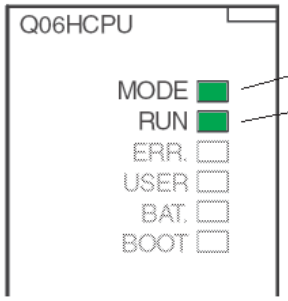
CPU Module	Number of connectable extension base units	Number of modules to be installed	Number of I/O points	
			Local (on main and extension base units)	Remote
Q00JCPU	2	16	256	2048
Q00CPU	4	24	1024	2048
Q01CPU				
Q02CPU	7	64	4096	8192
Q02HCPU				
Q06HCPU				
Q12HCPU				
Q25HCPU				

3.4.1 Part Names of CPU Modules



LED Display

– **MODE- und RUN-LED**



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 PLC 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“.

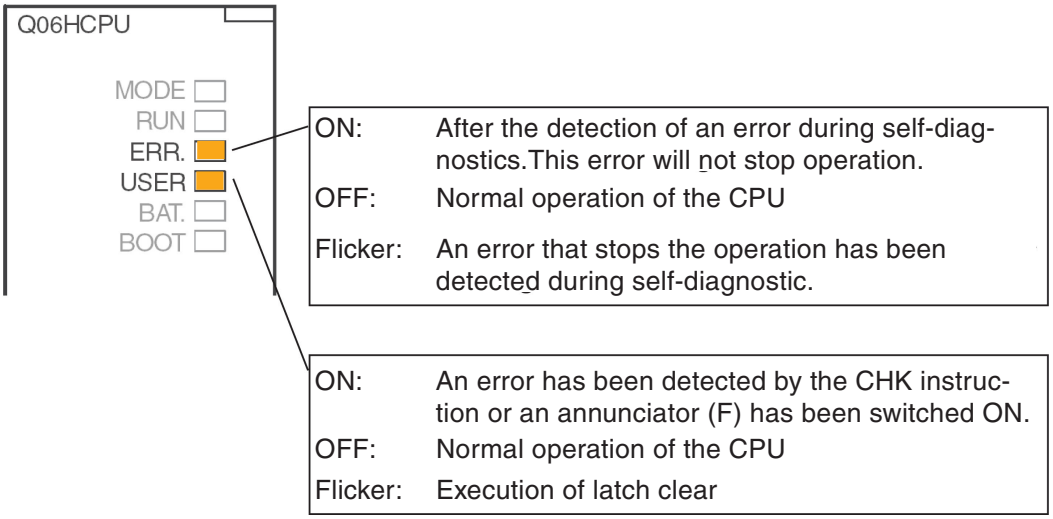
However, when you want to set the CPU to “RUN“ without clearing the device information:

① Switch the RUN/STOP switch from “STOP“ to “RUN

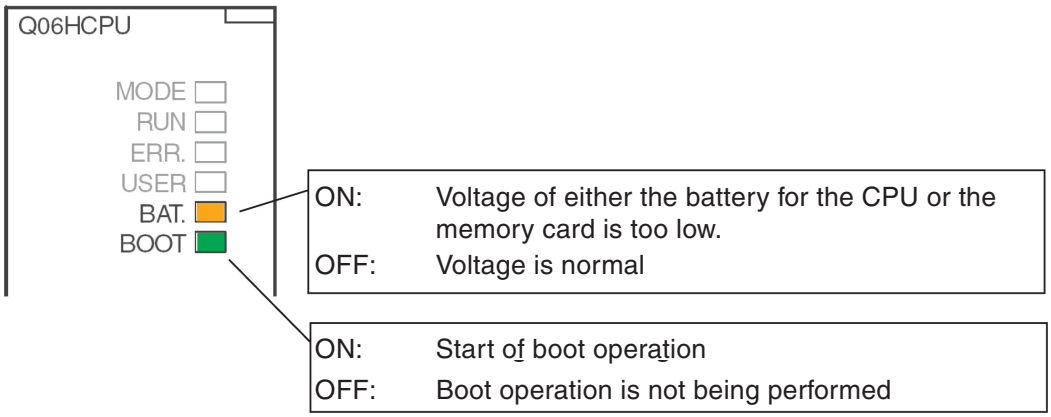
② Switch the RUN/STOP switch back to “STOP“

③ Switch the RUN/STOP switch to “RUN“.

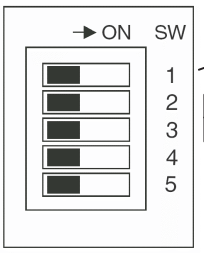
– **ERR. and USER LED**



– **BAT and BOOT LED**



System switches



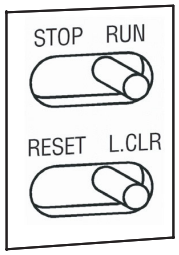
System protection
OFF: No protection
ON: Protection

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) (see also chapter 3.4.2).

All switches are shipped in the OFF position.

RUN/STOP Switch and RESET/L.CLR Switch



RUN: CPU executes sequence program

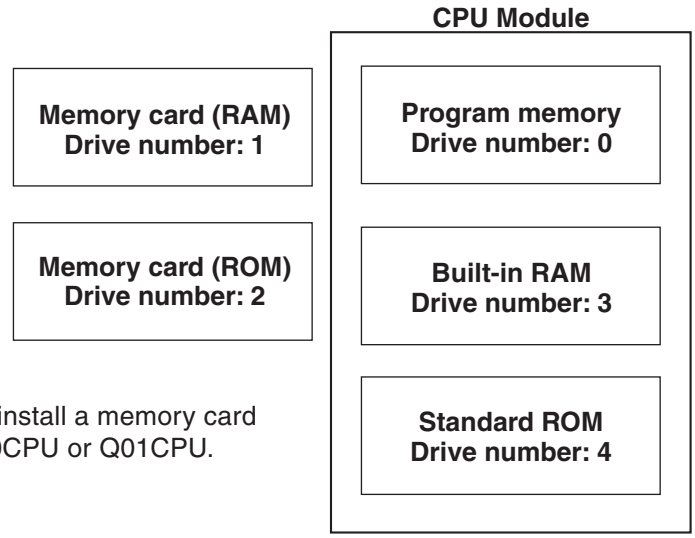
STOP: The execution of the sequence program is stopped

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)

3.4.2 Memory Organisation

The PLC CPUs uses multiple memories. These memories are identified by their drive number. In addition to the built-in memory high performance CPUs are equipped with a slot for a memory card.



It is not possible to install a memory card in a Q00JCPU, Q00CPU or Q01CPU.

Organisation of storage

- Q00JCPU, Q00CPU und Q01CPU

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

● = Storage is possible

○ = Storage is not possible

- Q02CPU, Q02HCPU, Q06HCPU, Q12HCPU and Q25HCPU:

Data	Built-in memory			Memory cards		
	Program 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

A program stored in the standard ROM or memory card (RAM or ROM) is transferred to the program memory at power-on and executed in the program memory. Hence, if the program is stored in the standard ROM or memory card (RAM or ROM), the program memory needs sufficient free space to accept that program.

For use of the debugging data for trace function, a failure history or a general-purpose file, the memory card must be loaded.

Overview of the data that can be stored

– Program
Ladder, list or SFC sequence program file. When running multiple programs, multiple program files are also stored in memory.

– Parameters
File storing PLC parameters and network parameters set during programming.

– Special function module parameter
Parameter file set using the GX Configurator. This file does not exist if you do not use the setting made with the GX Configurator.

– Device comment
File of device comments annotated to each device of the CPU. This file does not exist if device comments are not created.

– Device initial value
List of values given to devices in the CPU module at power-on. This file does not exist if device initial values are not used.

– File Register
File register (R, ZR) file. Setting different file names allows multiple file register files to be stored. Note that file registers can be stored in a ROM memory card (drive 2) but not in an ATA card (Q2MEM-8MBA/16MBA/32MBA). The file registers stored in a flash memory card allow read only in the program and do not allow data changes in the program.

– Local devices
Local devices are devices exclusively used with the corresponding programs in the pres-

ence of multiple programs.When processing any program, the corresponding local device data is transferred from the local device area to the executing device area and program processing is then performed.

– Debugging data
  Trace result storage file for the trace function used for program debugging.

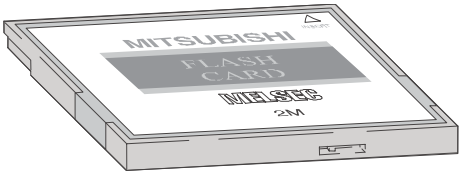
– Data file written by a FWRITE instruction
  These data can be stored on ATA memory cards (Q2MEM-8MBA/16MBA/32MBA)only.

Memory Cards

All PLC CPUs of the MELSEC System Q except the CPU modules Q00JCPU, Q00CPU and Q01CPU can be equipped with a memory card.

Before the memory card can be used for the first time the memory card must be formatted by GX Developer or GX IEC Developer.

A program stored in a memory card is transferred to the program memory at power-on and executed in the program memory. The behaviour at power-on can be set in the parameters (Boot file).

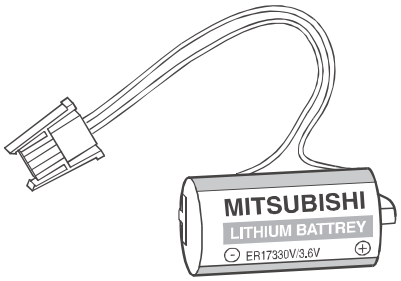


The write protect switch on the card will prevent any unintentional overwriting of stored data. A battery within the RAM memory card will hold the data during an interrupt of the power supply.

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		

3.4.3 Installation of the Battery for the CPU Module

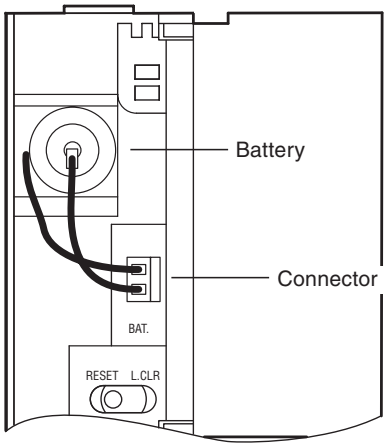


The PLC CPUs of the MELSEC System Q are equipped with a battery. During interruption of the power supply the battery can hold the data of the program memory, the built-in RAM and the clock for several thousand hours. However, this time depends on the type of CPU.

The battery should be changed every 10 years.

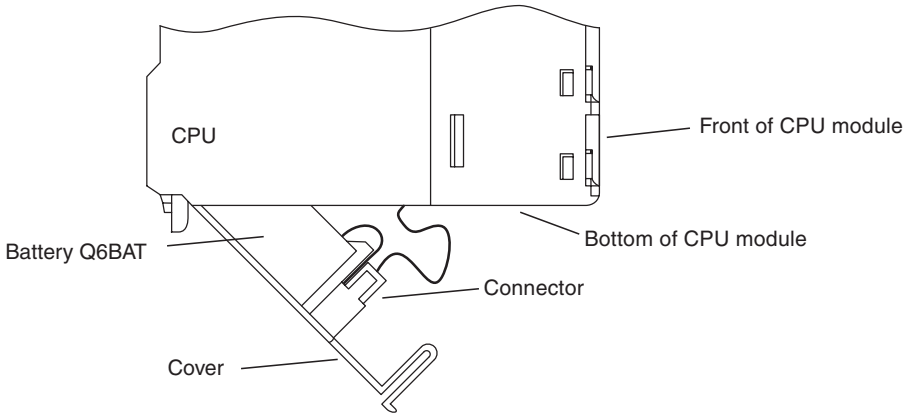
The SRAM memory cards have an own battery (Q2MEM-BAT) and are therefore independent from the battery of the CPU.

The CPU is shipped with battery installed but with its connector disconnected to prevent discharging and short circuits. Connect the battery before the CPU is used for the first time.



The battery of a Q00J, Q00 and Q01CPU is mounted behind the upper cover at the CPU modules front.

With all other PLC CPUs the battery is installed at the bottom side.



To connect the battery open the cover of the battery compartment of the CPU. Check that the battery is loaded correctly. Insert the battery connector into the connector pin at the case. Make sure that with the CPUs Q02(H), Q06H, Q12(P)H and Q25(P)HCPU the connector is inserted into the respective holder in the battery cover.

3.5 Digital Input and Output Modules

Input and output modules connect a PLC CPU with the process to be controlled. The digital inputs are used for inputting control signals from the connected switches, buttons or sensors. These inputs can read the values ON (power signal on) and OFF (no power signal). Digital output modules can switch external actuators ON or OFF.

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.

Output signals for example are used to control:

- Relays and contactors
- Signal lamps
- Solenoids
- Inputs of other devices e. g. inverters.

Overview of digital I/O module types

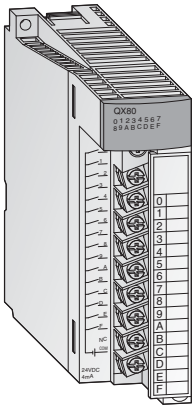
Type		Number of inputs/outputs			
		8	16	32	64
Input modules	120 V AC	○	●	○	○
	240 V AC	●	○	○	○
	48 V AC/DC	○	●	○	○
	24 V DC	○	●	●	●
	24 V DC (High speed)	●	○	○	○
	5 V DC / 12 V DC	○	●	●	●
Output modules	Relays	●	●	○	○
	Individual relay	●	○	○	○
	Triac output	○	●	○	○
	Transistor output (sink)	●	●	●	●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

● = Module is available

○ = Module is not available

3.5.1 Digital Input Modules

Input modules are available for various input voltages:



Input voltage \ No. of Inputs	Input module of MELSEC System Q			
	8	16	32	64
5 – 12 V DC		QX70	QX71	QX72
24 V DC		QX40 QX80	QX41 QX81	QX42 QX82
24 V DC (Interrupt module)		QI60		
48 V AC/DC		QX50		
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

NOTE

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).

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. 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 special modules such as the interrupt module QI60 should be adopted.

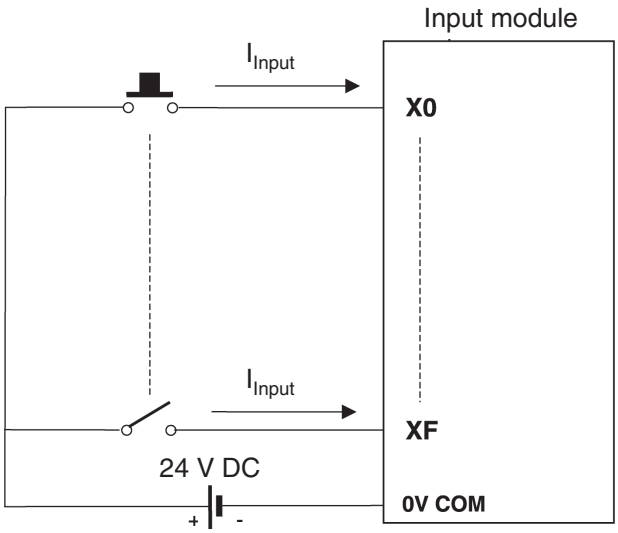
For the PLC to register a logical change in input condition, a minimum current has to flow through this input. This minimum current depends on the type of input module used and is 3 mA in most of the cases. Anything less than this will result in the input not turning on, even when a sensor connected to the input is switched on. The input current is limited by the input resistance. If the input voltage is higher than the rated voltage, the input current also increases. The input will accept up to a 7 mA signal, anything in excess of this could result in the input being damaged.

The PLC CPU polls the signal states of the inputs at the beginning of each program cycle and stores them. In the program the CPU accesses the stored states of the inputs. The stored states are updated again before the next program cycle is executed.

For the MELSEC System Q input modules for DC voltages are available for negative common or for positive common connections. For some modules like the QX71 you can choose between these two connection methods.

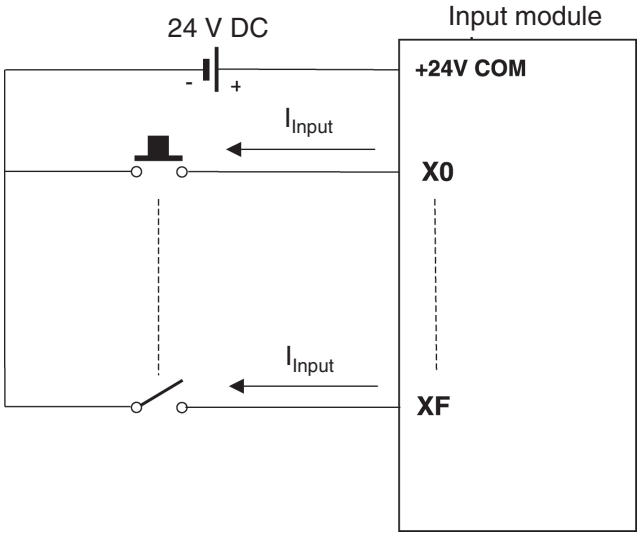
Negative common connection

A sensor connected to a negative common input module connects the positive pole of an external power supply with a PLC input. The negative pole of the power supply is connected to the common terminal for all inputs of this group. When the sensor is actuated, the input signal current flows into the input.



Positive common connection

A sensor connected to a positive common input module connects the negative pole of the external power supply with a PLC input. The common terminal for all inputs of this group is connected to the positive pole of the power supply. When the sensor is actuated, the input signal current flows out of the input.



Proximity Sensors and Optical Sensors

Proximity sensors issue a signal to the PLC when an object is in close proximity of the sensor. It is not necessary for the object to touch the sensor. This advantage makes many applications possible. There are two types of proximity sensors; inductive and capacitive.

There are also many varieties of optical sensors that may be found in industrial applications.

Most opto and proximity sensors utilise semiconductor outputs and these are available in two polarities, which are:

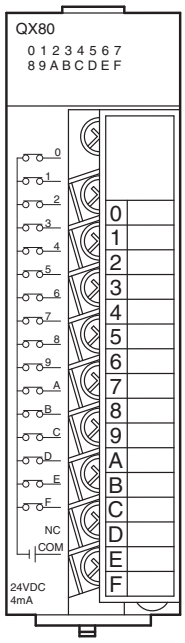
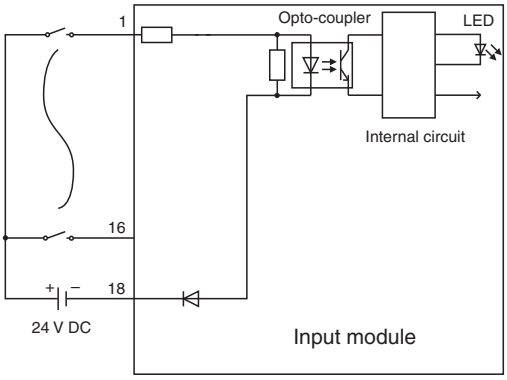
- PNP (source)
- NPN (sink)

The supply voltages to these sensors are commonly 24V DC.

Example for an negative common input module

Item		Specifications
Type of module		QX80
Number of input points		16
Isolation method		Opto-coupler
Rated input voltage		24 V DC (+20/-15%, ripple ratio within 5%)
Rated input current		Approx. 4 mA
Input derating		100 % (All inputs can be switched on simultaneously.)
Inrush current		Max. 200 mA for 1 ms (@ 132 V AC)
Voltage / current for ON		19 V DC or higher/ 3 mA or higher
Voltage / current for OFF		11 V DC or lower / 1.7 mA or lower
Input resistance		Approx. 5.6 kΩ
Response time	OFF → ON	1, 5, 10, 20, 70 ms (CPU parameter setting, initial setting: 10 ms)*
	ON → OFF	
Dielectric withstand voltage		560 V AC rms/3 cycles (altitude: 2000 m)
Insulation resistance		10 MΩ or more (by insulation resistance tester)
Noise immunity		By noise simulator of 500 V p-p noise voltage, 1μs noise width and 25 to 60 Hz noise frequency
		First transient noise IEC61000-4-4: 1kV
Groups of inputs		1 group with 16 inputs (Common terminal: terminal 18)
Operation indicator		1 LED for each input
External connections		18-point terminal block (M3 x 6 screws)
Applicable wire size		0.3 to 0.75 mm^2, core: 2.8 mm OD max.
Internal current consumption (5 V DC)		50 mA (all input points ON)
Weight		0.16 kg

* The response times for OFF -> ON and ON -> OFF can not be set separately.

Appearance	Circuit Diagram	Terminal	Signal
		1	X00
		2	X01
		3	X02
		4	X03
		5	X04
		6	X05
		7	X06
		8	X07
		9	X08
		10	X09
		11	X0A
		12	X0B
		13	X0C
		14	X0D
		15	X0E
		16	X0F
		17	Vacant
		18	COM

Function of a negative common input module

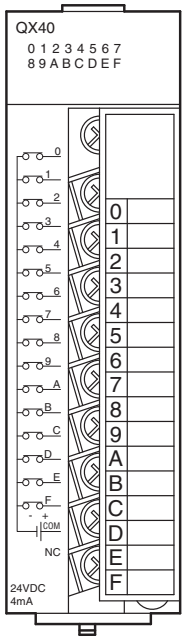
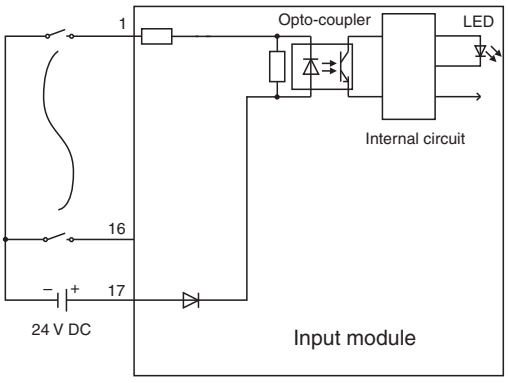
Referring to the preceding circuit diagram for the QX80, when the push button is closed, the direction of current flow will be as follows:

- From the +24 Volt terminal of the external power supply, through the push button and into the terminal 1 of the input module.
- Terminal 1 is connected to the negative pole of the external power supply (terminal 18) via a resistor and the LED of an opto-coupler. Thus a current flows trough 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-coupler 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 LED at the front side of the input module lits in this case and indicates the signal state.

Example for an positive common input module

Item		Specifications
Type of module		QX40
Number of input points		16
Isolation method		Opto-coupler
Rated input voltage		24 V DC (+20/-15%, ripple ratio within 5%)
Rated input current		Approx. 4 mA
Input derating		100 % (All inputs can be switched on simultaneously.)
Inrush current		Max. 200 mA for 1 ms (@ 132 V AC)
Voltage / current for ON		19 V DC or higher/ 3 mA or higher
Voltage / current for OFF		11 V DC or lower / 1.7 mA or lower
Input resistance		Approx. 5.6 kΩ
Response time	OFF → ON	1, 5, 10, 20, 70 ms (CPU parameter setting, initial setting: 10 ms)*
	ON → OFF	
Dielectric withstand voltage		560 V AC rms/3 cycles (altitude: 2000 m)
Insulation resistance		10 MΩ or more (by insulation resistance tester)
Noise immunity		By noise simulator of 500 V p-p noise voltage, 1μs noise width and 25 to 60 Hz noise frequency
		First transient noise IEC61000-4-4: 1kV
Groups of inputs		1 group with 16 inputs (Common terminal: terminal 17)
Operation indicator		1 LED for each input
External connections		18-point terminal block (M3 x 6 screws)
Applicable wire size		0.3 to 0.75 mm^2, core: 2.8 mm OD max.
Internal current consumption (5 V DC)		50 mA (all input points ON)
Weight		0.16 kg

* The response times for OFF -> ON and ON -> OFF can not be set separately.

Appearance	Circuit Diagram	Terminal	Signal
		1	X00
		2	X01
		3	X02
		4	X03
		5	X04
		6	X05
		7	X06
		8	X07
		9	X08
		10	X09
		11	X0A
		12	X0B
		13	X0C
		14	X0D
		15	X0E
		16	X0F
		17	COM
		18	Vacant

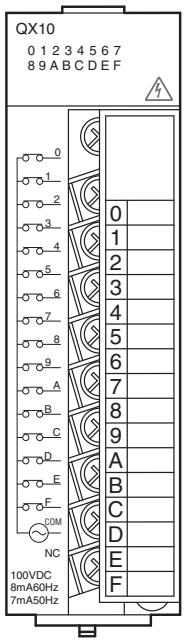
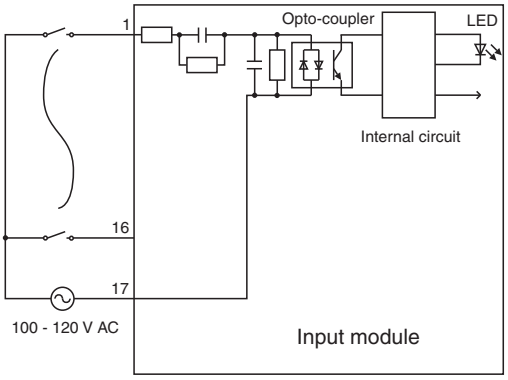
Function of a positive common input module

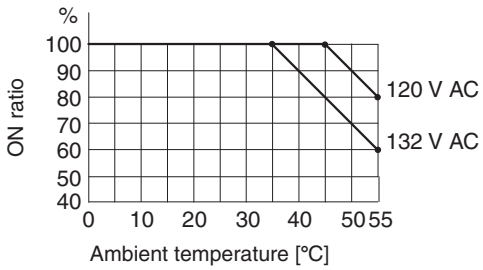
In the preceding diagram, when the push button connected to terminal 1 is closed, the direction of current flow will be as follows:

- From the +24 Volt terminal of the external power supply to the Common terminal (terminal 17) .
- Through the LED of the opto-coupler and then through the input resistor network circuit to terminal 1 (terminal for input X0) of the input module.
- 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 correspondent LED at the front side of the input module lits in this case and indicates the signal state.
- It then flows through the push button and then back to the negative pole of the external power supply.

Example for an AC input module

Item		Specifications
Type of module		QX10
Number of input points		16
Isolation method		Opto-coupler
Rated input voltage		100 to 120 V AC (+10/-15 %) 50/60 Hz (±3Hz) (distortion factor within 5 %)
Rated input current		approx. 8 mA @100 V AC, 60 Hz; approx. 7 mA @ 100 V AC, 50 Hz
Input derating		refer to derating chart below
Inrush current		Max. 200 mA for 1 ms (@ 132 V AC)
Voltage / current for ON		80 V AC or higher/ 5 mA or higher (50 Hz, 60 Hz)
Voltage / current for OFF		30 V DC or lower / 1 mA or lower (50 Hz, 60 Hz)
Input resistance		approx. 15 kΩ @ 60 Hz, approx. 18 kΩ @ 50 Hz
Response time	OFF → ON	15 ms or less (100 V AC, 50 Hz, 60 Hz)
	ON → OFF	20 ms or less (100 V AC, 50 Hz, 60 Hz)
Dielectric withstand voltage		1780 V AC rms/3 cycles (altitude: 2000 m)
Insulation resistance		10 MΩ or more (by insulation resistance tester)
Noise immunity		By noise simulator of 1500 V p-p noise voltage, 1μs noise width and 25 to 60 Hz noise frequency
		First transient noise IEC61000-4-4: 1kV
Groups of inputs		1 group with 16 inputs (Common terminal: terminal 17)
Operation indicator		1 LED for each input
External connections		18-point terminal block (M3 x 6 screws)
Applicable wire size		0.3 to 0.75 mm^2, core: 2.8 mm OD max.
Internal current consumption (5 V DC)		50 mA
Weight		0.17 kg

Appearance	Circuit Diagram	Terminal	Signal
		1	X00
		2	X01
		3	X02
		4	X03
		5	X04
		6	X05
		7	X06
		8	X07
		9	X08
		10	X09
		11	X0A
		12	X0B
		13	X0C
		14	X0D
		15	X0E
		16	X0F
		17	COM
		18	Vacant

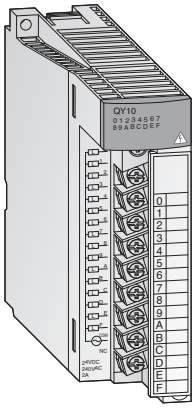


For the module QX10, the number of inputs which can be switched on simultaneously, depends on the ambient temperature.

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 –120 V AC). This minimises the possibility of an incorrect voltage being connected to the inputs.

3.5.2 Digital Output Modules

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



Output type	No. of outputs / Rated output voltage	Output module 8	16	32	64
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		QY40P QY50 QY80	QY41P QY81P	QY42P
	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

Digital output modules for the MELSEC System Q are available in four configurations.

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

Type	Advantages	Advantages
Relay	● One modul can switch mixed voltages ● 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 Output Modules

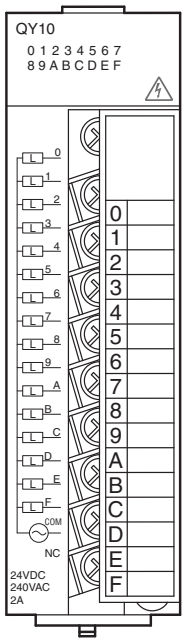
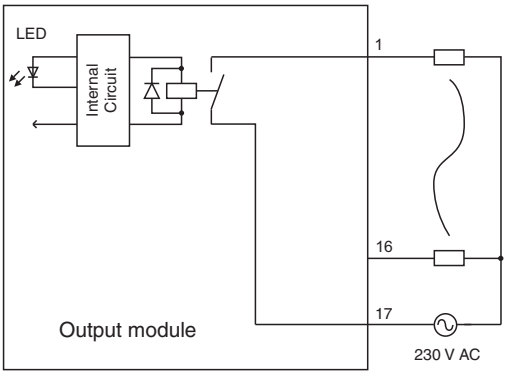
Electrical isolation from the internal and external circuitry is achieved by the coils and the contacts of the output relays.

Modules are available as multiple outputs with isolated grouped commons or as individually isolated "volt-free" outputs (QY18A).

Similar to the other types of output modules, the operation of the output contact is driven by the internal CPU program. At the end of the program 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.

Example for a relay output module

Item		Specifications
Type of module		QY10
Number of output points		16
Isolation method		Relay
Rated switching voltage / current		24 V DC, 2 A (resisitive load) per output 240 V AC, 2 A (cosφ = 1) per output; max. 8 A per group
Minimum switching load		5 V DC, 1 mA
Maximum switching voltage		125 V DC / 264 V AC
Response time	OFF → ON	10 ms or less
	ON → OFF	12 ms or less
Life	Mechanical	20 million times or more
	Electrical	100,000 times or more @ rated switching voltage/current load
		100,000 times or more @ 200 V AC, 1.5 A; 240 V AC 1 A (cos φ = 0.7) 300,000 times or more @ 200 V AC, 0.4 A; 240 V AC 0.3 A (cos φ = 0.7)
		100,000 times or more @ 200 V AC, 1 A; 240 V AC 0.5 A (cos φ = 0.35) 300,000 times or more @ 200 V AC, 0.3 A; 240 V AC 0.15 A (cos φ = 0.35)
		100,000 times or more @ 24 V DC 1 A; 100 V DC 0.1 A (L/R = 0.7 ms) 300,000 times or more @ 24 V DC 0.3 A; 100 V DC 0.03 A (L/R = 0.7ms)
Maximum switching frequency		3600 times/hour
Surge suppressor		—
Fuse		—
Dielectric withstand voltage		2830 V AC rms/3 cycles (altitude: 2000 m)
Insulation resistance		10 MΩ or more (by insulation resistance tester)
Noise immunity		By noise simulator of 1500 V p-p noise voltage, 1μs noise width and 25 to 60 Hz noise frequency
		First transient noise IEC61000-4-4: 1kV
Groups of outputs		1 group with 16 outputs (Common terminal: terminal 17)
Operation indicator		1 LED for each output
External connections		18-point terminal block (M3 x 6 screws)
Applicable wire size		0.3 to 0.75 mm^2, core: 2.8 mm OD max.
Internal current consumption (5 V DC)		430 mA
Weight		0.22 kg

Appearance	Circuit Diagram	Terminal	Signal
		1	Y00
		2	Y01
		3	Y02
		4	Y03
		5	Y04
		6	Y05
		7	Y06
		8	Y07
		9	Y08
		10	Y09
		11	Y0A
		12	Y0B
		13	Y0C
		14	Y0D
		15	Y0E
		16	Y0F
		17	COM
		18	Vacant

Triac Output Modules

Digital output modules can switch voltages of 100 to 240 V AC. As with all other output configurations the physical output is isolated by opto-coupler. The response of the triac is obviously faster than the relay with a response time of 1ms to turn ON and 10 ms to turn OFF again.

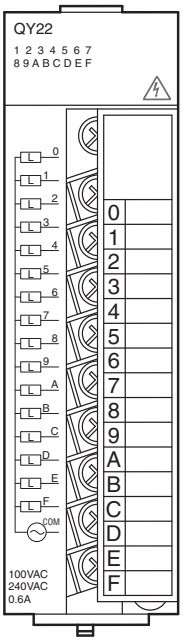
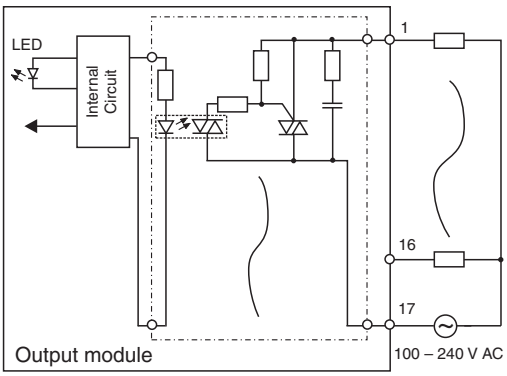
Since the load of an triac output is restricted to 0.6 A, care should be taken when configuring your system so as not to overload the output circuitry.

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.

DANGER:
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!

Example for a triac output module

Item		Specifications
Type of module		QY22
Number of output points		16
Isolation method		Opto-coupler
Rated switching voltage / current		100–240 V AC (+20/-15 %), 0.6 A per output, 4.8 A per module
Minimum switching load		24 V AC, 100 mA; 100 V AC, 25 mA, 240 V AC, 25 mA
Maximum inrush current		20 A
Leakage current at OFF		3 mA or lower @ 120 V AC, 60 Hz 1.5 mA or lower @ 240 V AC, 60 Hz
Maximum voltage drop at ON		1.5 V
Response time	OFF → ON	0.5 x period + max. 1 ms
	ON → OFF	0.5 x period + max. 1 ms
Surge killer		CR absorber
Fuse		—
Dielectric withstand voltage		2830 V AC rms/3 cycles (altitude: 2000 m)
Insulation resistance		10 MΩ or more (by insulation resistance tester)
Noise immunity		By noise simulator of 1500 V p-p noise voltage, 1μs noise width and 25 to 60 Hz noise frequency
		First transient noise IEC61000-4-4: 1kV
Groups of outputs		1 group with 16 outputs (Common terminal: terminal 17)
Operation indicator		1 LED for each output
External connections		18-point terminal block (M3 x 6 screws)
Applicable wire size		0.3 to 0.75 mm^2, core: 2.8 mm OD max.
Internal current consumption (5 V DC)		250 mA (When all outputs are switched ON.)
Weight		0.40 kg

Appearance	Circuit Diagram	Terminal	Signal
		1	Y00
		2	Y01
		3	Y02
		4	Y03
		5	Y04
		6	Y05
		7	Y06
		8	Y07
		9	Y08
		10	Y09
		11	Y0A
		12	Y0B
		13	Y0C
		14	Y0D
		15	Y0E
		16	Y0F
		17	COM
		18	Vacant

Transistor Output Modules

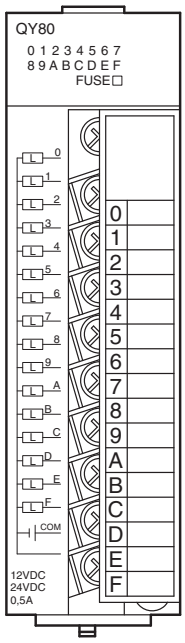
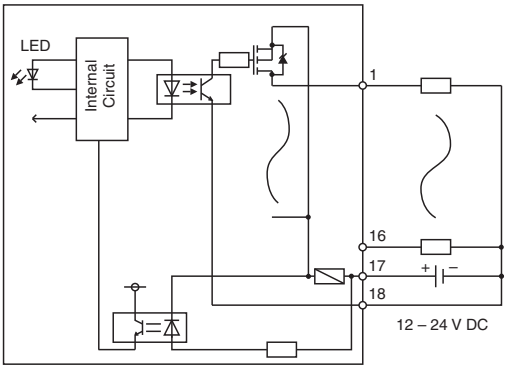
As with all other output configurations the physical outputs of transistor output modules are isolated by opto-coupler.

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

For the MELSEC System Q transistor output modules are available in sink and source configuration.

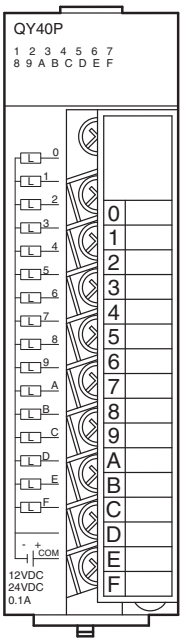
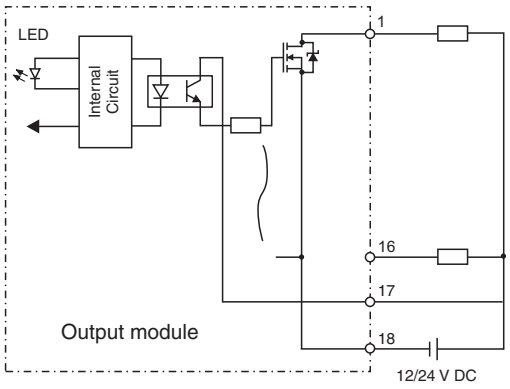
Example for a source transistor output module

Item		Specifications
Type of module		QY80
Number of output points		16
Isolation method		Opto-coupler
Rated switching voltage		12 to 24 V DC (+20/-15%)
Switching voltage range		10.2 to 28.8 V DC
Maximum load current		0.5 A per output, 4 A per group
Maximum inrush current		4 A for 10 ms
Leakage current at OFF		0.1 mA or less
Maximum voltage drop at ON		0.2 V DC @ 0.5 A (TYP), maximum 0.3 V @0.5 A
Response time	OFF → ON	1 ms or less
	ON → OFF	1 ms or less (rated load, resistive load)
Surge suppressor		Zener diode
Fuse		6.7 A (unchangeable)
Fuse blow indication		LED indicates blown fuse and an signal is issued to the PLC CPU
External power supply	Voltage	12 to 24 V DC (+20/-15%, ripple ratio within 5%)
	Current	20 mA (at 24 V DC and when all outputs are ON)
Dielectric withstand voltage		560 V AC rms/3 cycles (altitude: 2000 m)
Insulation resistance		10 MΩ or more (by insulation resistance tester)
Noise immunity		By noise simulator of 500 V p-p noise voltage, 1μs noise width and 25 to 60 Hz noise frequency
		First transient noise IEC61000-4-4: 1kV
Groups of outputs		1 group with 16 outputs (Common terminal: terminal 17)
Operation indicator		1 LED for each output
External connections		18-point terminal block (M3 x 6 screws)
Applicable wire size		0.3 to 0.75 mm^2, core: 2.8 mm OD max.
Internal current consumption (5 V DC)		80 mA (When all outputs are switched ON.)
Weigth		0.17 kg

Appearance	Circuit Diagram	Terminal	Signal
		1	Y00
		2	Y01
		3	Y02
		4	Y03
		5	Y04
		6	Y05
		7	Y06
		8	Y07
		9	Y08
		10	Y09
		11	Y0A
		12	Y0B
		13	Y0C
		14	Y0D
		15	Y0E
		16	Y0F
		17	COM
		18	0 V

Example for a sink transistor output module

Item		Specifications
Type of module		QY40P
Number of output points		16
Isolation method		Opto-coupler
Rated switching voltage		12 to 24 V DC (+20/-15%)
Switching voltage range		10.2 to 28.8 V DC
Maximum load current		0.1 A per output, 1.6 A per group
Maximum inrush current		0.7 A for 10 ms
Leakage current at OFF		0.1 mA or less
Maximum voltage drop at ON		0.1 V DC @ 0.1 A (TYP), maximum 0.2 V @0.1 A
Response time	OFF → ON	1 ms or less
	ON → OFF	1 ms or less (rated load, resistive load)
Surge suppressor		Zener diode
Fuse		—
External power supply	Voltage	12 to 24 V DC (+20/-15%, ripple ratio within 5%)
	Current	10 mA (at 24 V DC and when all outputs are ON)
Dielectric withstand voltage		560 V AC rms/3 cycles (altitude: 2000 m)
Insulation resistance		10 MΩ or more (by insulation resistance tester)
Noise immunity		By noise simulator of 500 V p-p noise voltage, 1μs noise width and 25 to 60 Hz noise frequency
		First transient noise IEC61000-4-4: 1kV
Groups of outputs		1 group with 16 outputs (Common terminal: terminal 18)
Operation indicator		1 LED for each output
External connections		18-point terminal block (M3 x 6 screws)
Applicable wire size		0.3 to 0.75 mm^2, core: 2.8 mm OD max.
Internal current consumption (5 V DC)		65 mA (When all outputs are switched ON.)
Weigth		0.16 kg

Appearance	Circuit Diagram	Terminal	Signal
		1	Y00
		2	Y01
		3	Y02
		4	Y03
		5	Y04
		6	Y05
		7	Y06
		8	Y07
		9	Y08
		10	Y09
		11	Y0A
		12	Y0B
		13	Y0C
		14	Y0D
		15	Y0E
		16	Y0F
		17	12/24 V DC
		18	COM

3.6 Special Function Modules

3.6.1 Analog Modules

When you automate processes you will frequently need to acquire or control analog values such as temperatures, pressures and filling levels. Additional analog modules are thus required for inputting and outputting analog signals.

Basically, there are two different kinds of analog modules:

- Analog input modules and
- Analog output modules.

Analog input modules can acquire current, voltage and temperature values. Analog output modules send current or voltage signals to the module's outputs.

Criteria for selecting analog modules

A wide range of analog modules are available for the MELSEC System Q and you need to choose the correct module for each automation task. The main criteria for selection are as follows:

- Resolution

  The resolution describes the smallest physical value that can be acquired or output by the analog module.

  In the case of analog input modules the resolution is defined as the change in voltage, current or temperature at the input that will increase or decrease the digital output value by 1.

  In the case of analog output modules the resolution is the change in the voltage or current value at the module output caused by increasing or decreasing the digital input value by 1.

  The resolution is restricted by the internal design of the analog modules and depends on the number of bits required to store the digital value. For example, if a 10V voltage is acquired with a 12-bit A/D converter the voltage range is divided into 4,096 steps ($2^{12} = 4096$, see section 4.3). This results in a resolution of 10V/4096 = 2.5mV.

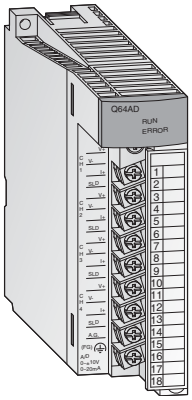
- Number of analog inputs or outputs

  The inputs or outputs of analog modules are also referred to as channels. You can choose analog input modules with 2, 4 or 8 channels, depending on the number of channels you need.

Analog input modules

Analog input modules convert a measured analog value (e.g. 10 V) into a digital value (e.g. 4000) that can be processed by the PLC. This conversion process is known as analog/digital conversion or A/D conversion for short.

Temperatures can be acquired directly by the analog modules of the MELSEC System Q but other physical values like pressures or flow rates must first be converted into current or voltage values before they can be converted into digital values for processing by the PLC. This conversion is performed by sensors that output signals in standardised ranges (for example 0 to 10 V or 4 to 20 mA). The measurement of a current signal has the advantage that the value is not falsified by the length of the cables or contact resistances.



The analog input modules of the MELSEC System Q 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	Module
Voltage	-10 to +10 V	1 to 5 V 0 to 5 V 0 to 10 V -10 to +10 V	8	Q68ADV
Current	0 to 20 mA	0 to 20 mA 4 to 20 mA	8	Q68ADI
Voltage or current (can be selected for each channel)	-10 to +10 V 0 to 20 mA	As for 68ADV and Q68ADI	4	Q64AD

Analog input modules for temperature acquisition

Temperature values can be acquired with two different sensor technologies: Pt100 resistance thermometers and thermocouples.

- Pt100 resistance thermometers

  These devices measure the resistance of a platinum element, which increases with temperature. At 0°C the element has a resistance of 100°C (thus the name Pt100). The resistance sensors are connected in a three-wire configuration, which helps to ensure that the resistance of the connecting cables does not influence the measurement result.

  The maximum measurement range of Pt100 resistance thermometers is from -200°C to +600°C but in practice it also depends on the capabilities of the temperature acquisition module being used.

  Another metal used for resistance thermometers is nickel (Ni100). In this case the measurement range is smaller (-60°C to 180°C).

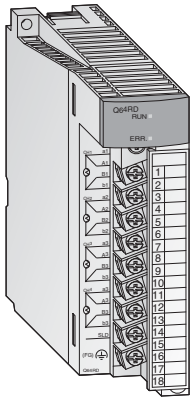
- Thermocouples

  These temperature measurement devices take advantage of the fact that a voltage is generated when heat is applied to an element made of two different metals. This method thus measures a temperature with the help of a voltage signal.

  There are different kinds of thermocouples. They differ in their thermal electromotive force (thermal e.m.f.) and the temperature ranges they can measure. The material combinations used are standardized and are identified by a type code. Types J and K are both commonly used. Type J thermocouples use a combination of iron (Fe) and a copper/nickel alloy (CuNi), type K thermocouples use a NiCr and Ni combination. In addition to their basic construction thermocouples also differ in the temperature range they can measure.

  Thermocouples can be used to measure temperatures from -200°C to +1,200°C.

Special features



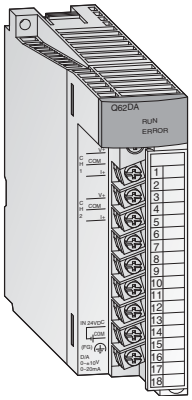
- Up to 4 temperatures can be measured by one module
- The disconnection of the temperature sensor can be detected on each channel
- Selection of sampling processing/time averaging processing/count averaging processing
- Error compensation by offset/gain value setting
- Alarm output when limit value is exceeded
- Potential isolation between process and control by means of an opto-coupler is a standard feature. Additional potential isolation between the channels for Q64TDV-GH and Q64RD-G.

Temperature sensor	Temperature acquisition range	Max. Resolution	Module
Resistance thermometer (Pt100, JPt100)	Pt100: -200 to 850°C, JPt 100: -180 to 600°C	0.025 °C	Q64RD
Resistance thermometer (Pt100, JPt100, Ni100)	Pt100: -200 to 850°C, JPt 100: -180 to 600°C, Ni100: -60 to 180 °C	0.025 °C	Q64RD-G
Thermocouples type K, E, J, T, B, R, S or N	Depends on the thermo-couple used	B, R, S, N: 0.3°C; K, E, J, T: 0.1°C	Q64TD
		B: 0.7°C; R, S: 0.8°C; K, T: 0.3 °C; E,T: 0.2°C; J: 0.1°C; N: 0.4 °C; Voltage: 4 µV	Q64TDV-GH

Analog output modules

Analog output modules convert a digital value from the PLC CPU into an analog voltage or current signal that can be used to control an external device (digital/analog conversion or D/A conversion).

The analog output signals generated by the MELSEC System Q use the standard industrial ranges of 0–10V and 4–20mA.



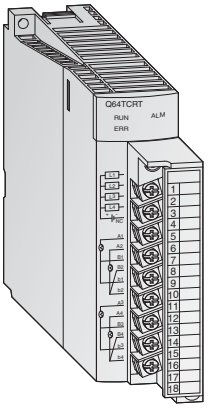
The resolution of 0.333 mV respectively 0.83 µA and the extremly short conversion time of 80 µs per output channel are only two of the many features of this modules. Isolation between process and control by means of opto-couplers is also a standard feature.

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

3.6.2 Temperature Control Modules with PID Algorithm

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

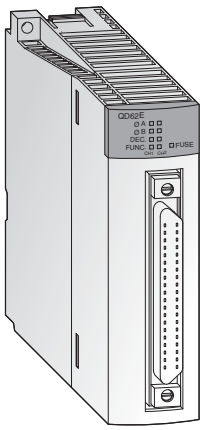


Special features

- Four temperature input channels and four PID control circuits per module
- Temperature acquisition either with Pt100 resistance thermometers (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 four PID control circuits
- Transistor output with pulse train to drive the actuator in the control circuit

3.6.3 High-Speed Counter Modules

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



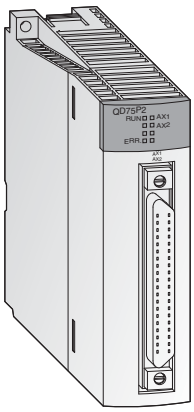
Special features

- 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 40-pin plug.

3.6.4 Positioning Modules

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

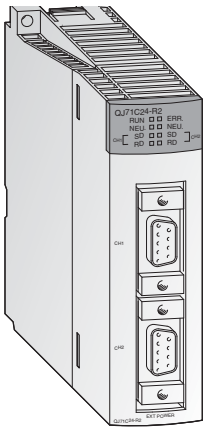


Special features

- 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 defined in 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® software GX Configurator QP.

3.6.5 Serial Communication Modules

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

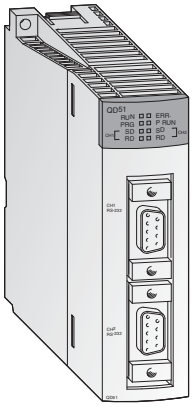


Special features

- 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 System 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.

3.6.6 BASIC Programmable Interface 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 to the buffer memory of intelligent function modules is supported
- Remote RUN/STOP is supported via the serial communication line

3.7 Networks and Network Modules

3.7.1 Networking on all Levels

For complex or widely ramified applications but also for the implementation of remote inputs and outputs or for the visualization of processes the communication between PLCs, production control computers, operator terminals and other devices is very important.

Mitsubishi Electric offers optimal solutions based on a three-level network:

- Production level
- Control level
- Command level

Production level

A field network which links control devices, such as PLC, with remote inputs and outputs, inverters and operator terminals is on the lowest network level in production locations.

While control devices were previously linked with sensors and drive equipment by wires on a point-by-point basis, the field network can connect multiple sensors and drive equipment with a single network cable, reducing the number of wires and wiring processes. When connected with intelligent equipment such as the ID system, bar-code reader, inverter and display, the field network allows production data control at network ends through transfer of various data, in addition to ON/OFF data, and serves for improved maintenance efficiency by centralized control of equipment operating statuses.

The high speed and high performance are further increased when combined with a PLC of the MELSEC System Q while at the same time easy handling is guaranteed.

Control level

A control network which links control devices, e.g. PLC and CNC, is on the middle network level in production sites. Designed to transfer data directly related to the operations and motions of machinery and equipment between the control devices, the control network is required to have excellent real-time capabilities. MELSECNET(10/H), MELSEC's control network, is highly regarded in the market for its excellent real-time capabilities, simple network settings, highly redundant reliability typified by duplex loop.

Command level

On the highest network level in production fields in an information network. Designed to transfer production control information, quality control information, facility operating status and other information between the PLC or facility controller and the production control computer, the information network assumes the use of the most general-purpose Ethernet. Ethernet accepts not only a wide variety of computers such as Windows and UNIX type personal computers but also various Factory Automation equipment. The MELSEC System Q has functions which make the best use of the Ethernet features.

In addition to the above levels networks can be divided into

- Open networks

and

- MELSEC networks.

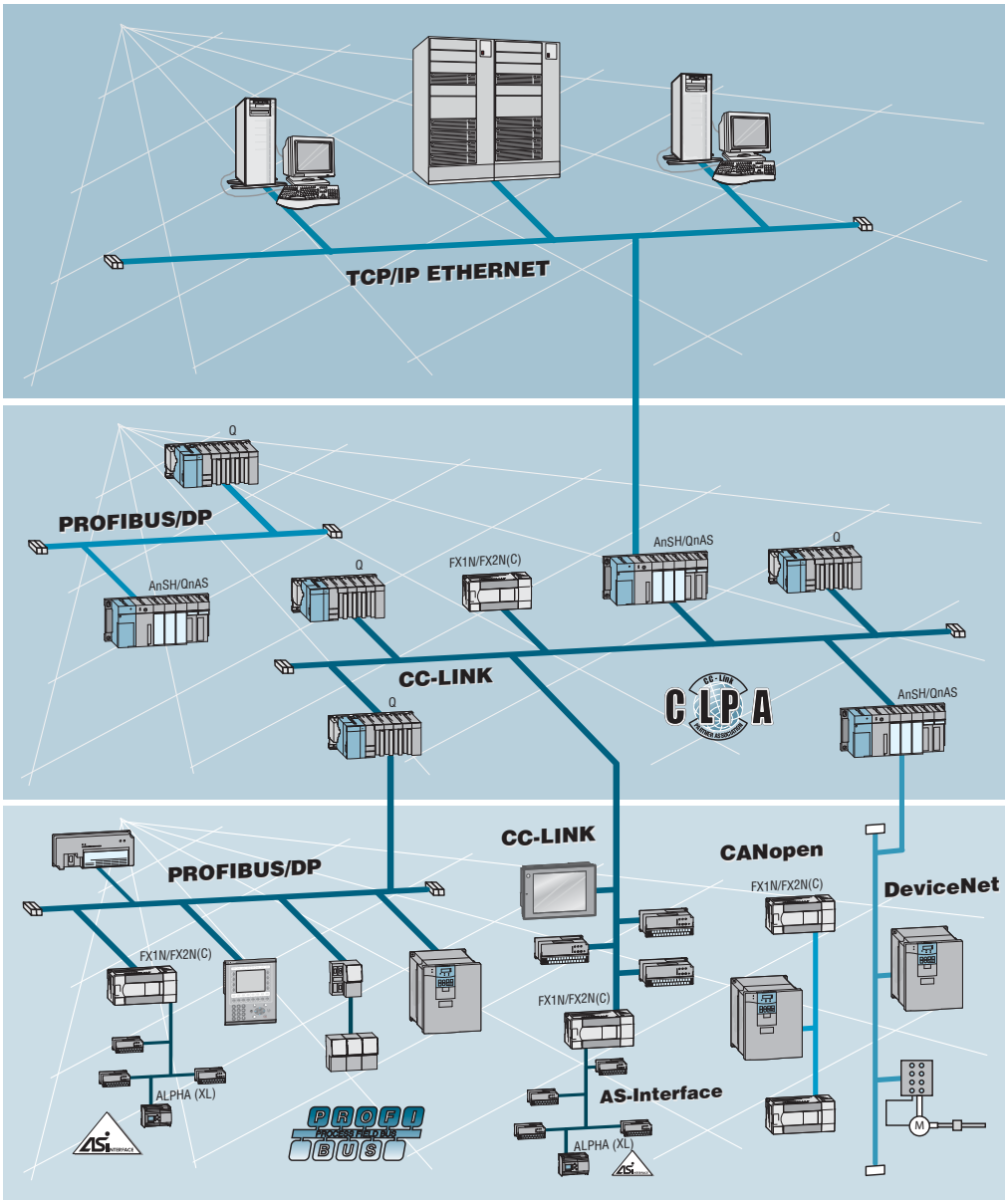
3.7.2 Open Networks

Open networks are manufacturer independent, i. e. those networks are also used by other manufacturers. Thus the communication between a MELSEC PLC and devices of third-party manufacturers is possible.

Command Level
ETHERNET

Control Level
PROFIBUS/DP
CC-Link

Production Level
PROFIBUS/DP
DeviceNet
AS-Interface
CC-Link
CANopen



ETHERNET

ETHERNET is the most widespread network for connection of information processors such as personal computers and work stations.ETHERNET is a platform for a very wide range of data communications protocols. The combination of ETHERNET and the extremely widespread TCP/IP protocol enables high-speed data communications between process supervision systems and the MELSEC PLC series.

TCP/IP provides logical point-to-point links between two ETHERNET stations. Using the TCP/IP protocol a process supervision system can request up to 960 data words per query if the MELSEC System Q module is used.

PROFIBUS/DP

The open PROFIBUS/DP network enables extremely fast data exchange with a very wide variety of slave devices, including:

- Remote digital I/Os and remote analog I/Os
- Frequency inverters
- Operator terminals
- A range of other devices from third-party manufacturers

To help reduce costs PROFIBUS/DP uses RS 485 technology with shielded 2-wire cabling.

CC-Link

The open fieldbus and control network CC-Link provides a fast data communications with different devices. The following components from MITSUBISHI ELECTRIC among others can be integrated:

- MELSEC PLC systems
- Remote digital I/Os and remote analog I/Os
- Positioning modules
- Frequency inverters
- Operator terminals
- Robots
- Third-party devices like bar code readers

Various data like digital and analog data can be exchanged easily. In addition to the cyclic transmission of word data, CC-Link systems handle transient transmission (message transmission) as well. This enables data communications with intelligent devices such as display devices, bar code readers, measuring devices, personal computers and PLC systems (up to 24 CPUs) as well beside the analog and digital devices.

DeviceNet

DeviceNet represents a cost-effective solution for the network integration of low-level terminal equipment. Up to 64 devices including a master can be integrated in one network.

AS Interface

The AS interface is an international standard for the lowest field bus level. The network suits versatile demands, is very flexible and particularly easy to install. Suitable for controlling **A**ctuators, like solenoids or indicators and **s**ensors, hence the name AS-i.

CANopen

CANopen is an “open” implementation of the **C**ontroller **A**rea **N**etwork (CAN),

CANopen networks are used for connecting sensors, actuators and controllers in industrial control systems, medical equipment, maritime electronics, railways, trams and commercial vehicles.

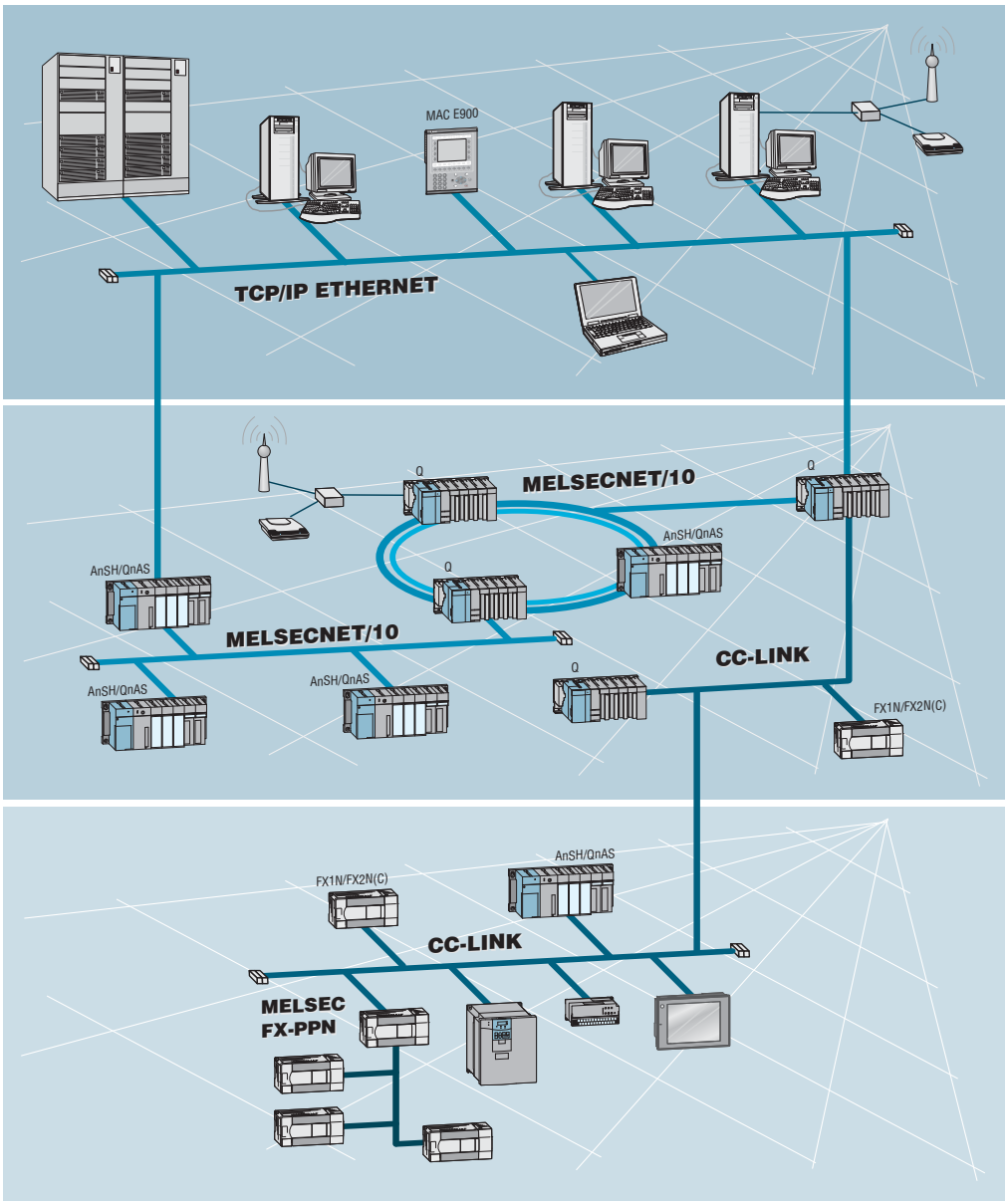
CANopen Network modules are available for the MELSEC FX family of controllers.

3.7.3 MELSEC Networks

Command Level
TCP/IP ETHERNET

Control Level
CC-Link
MELSECNET/10
MELSECNET/H

Production Level
CC-Link
MELSEC FX-PPN



MELSECNET/10/H

MELSECNET/10 and MELSECNET/H are high-speed networks for the data exchange between MELSEC PLCs. Even remote I/O-Stations can be integrated in these networks. MELSECNET/10/H enables you to program and monitor every PLC in the system from any station.

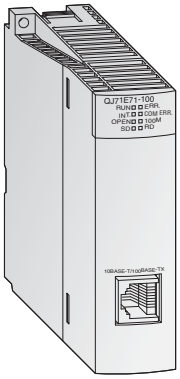
Up to 255 MELSECNET/10/H networks can be linked together. The build-in router functionality allows easy data transfer from one network to another. For the cyclic communication an extraordinary large amount of data (8192 words and 8192 relays) is available. In parallel to the cyclic data exchange it is also possible for any station to send data to and read data from any other station, even across several networks.

MELSECNET/10 gives you a wide choice of cable types and topologies: from coaxial bus (max. 500 m) over a coaxial duplex loop to a fibre-optics duplex loop for distances of up to 30 km(!).

3.7.4 Network Modules

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.

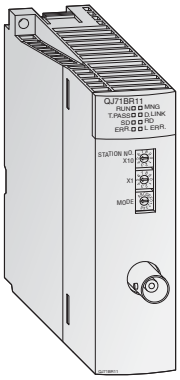


Special features

- 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 from GX Developer or GX IEC Developer on a personal computer via ETHERNET.

MELSECNET Modules

The modules QJ71BR11 and QJ71LP21 are used to connect the MELSEC 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.

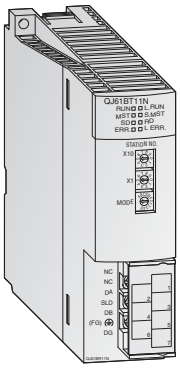


Special features

- Two different topologies are featured: Coaxial bus (QJ71BR11) or redundant optical loop (QJ71LP21).
- High data transfer rates: 10 Mbit/s with coaxial bus systems 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

Master/Local Module for CC-Link

The QJ61BT11N is applicable as a master or local station in a CC-Link system and manages the connection of remote inputs and outputs.

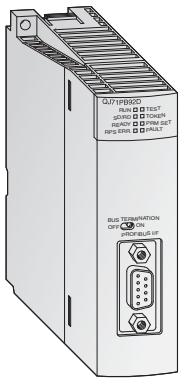


Special features

- 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

PROFIBUS/DP Interface Modules

The PROFIBUS/DP master modules QJ71PB92D and QJ71PB92V as well as the QJ71PB93D PROFIBUS/DP slave module enable PLCs of the System Q to communicate with other PROFIBUS devices.

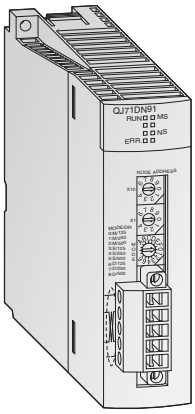


Special features

- 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 exchange with automatic refresh is supported. Batch transfer can be chosen as an option.

DeviceNet Master Module QJ71DN91

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

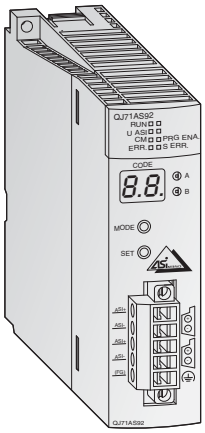
- The positions of master and slave stations are user selectable.
- Transfer rates of 125, 250 and 500 kBaud
- Transmission distances of up to 500 m
- Communication methods
  - Polling
  - Bit strobe
  - Change of state
  - Cyclic

Master Modules for AS Interface

The QJ71AS92 is a master module for connecting System Q to the AS-interface system.

The QJ71AS92 can control up to 62 slave units (group A: 31 / group B: 31) with up to 4 inputs and 4 outputs each per address. The addresses of the slave devices across the AS-interface are assigned automatically by the master.

The maximum transmission distance witout a repeater is 100 m. The maximum transmission distance can be increased up to 300 m by two repeaters.

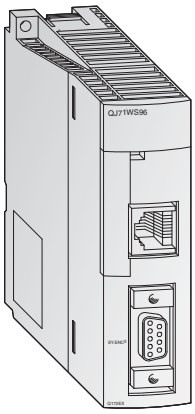


Special features

- Up to 62 slave units can be configured across two networks.
- Up to 496 digital inputs/outputs can be driven via the master.
- Communications via AS-i coded flat or round cable
- Highly efficient error securing system
- Automatic data exchange with the PLC

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

4 An Introduction to Programming

A program consists of a sequence of program instructions. These instructions determine the functionality of the PLC and they are processed sequentially, in the order in which they were entered by the programmer. To create a PLC program you thus need to analyse the process to be controlled and break it up into steps that can be represented by instructions. A program instruction, represented by a line or “rung” in ladder diagram format, is the smallest unit of a PLC application program.

4.1 Structure of a Program Instruction

A program instruction consists of the instruction itself (sometimes referred to as a command) and one or more (in the case of applied instructions) operands, which in a PLC are references to devices. Some instructions are entered on their own without specifying any operands – these are the instructions that control program execution in the PLC.

Every instruction you enter is automatically assigned a step number that uniquely identifies its position in the program. This is important because it is quite possible to enter the same instruction referring to the same device in several places in the program.

The illustrations below show how program instructions are represented in the Ladder Diagram (LD, left) and Instruction List (IL, right) programming language formats:

The instruction describes **what** is to be done, i.e. the function you want the controller to perform. The operand or device is what you want to perform the function **on**. Its designation consists of two parts, the device name and the device address:

Examples of devices:

Device name	Type	Function
X	Input	Input terminal on the PLC (e.g. connected to a switch)
Y	Output	Output terminal on the PLC (e.g. for a contactor or lamp)
M	Relay	A buffer memory in the PLC that can have two states, ON or OFF
T	Timer	A “time relay” that can be used to program timed functions
C	Counter	A counter
D	Data register	Data storage in the PLC in which you can store things like measured values and the results of calculations.

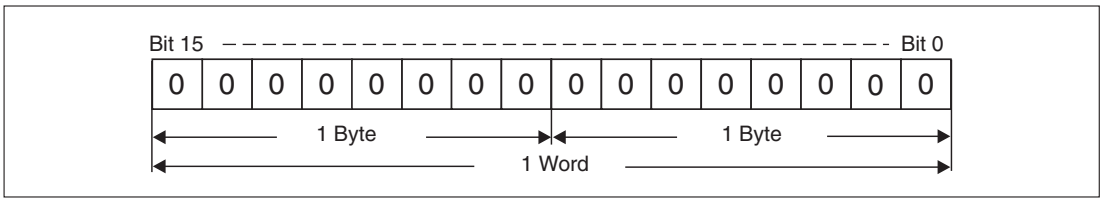
See Chapter 5 for a detailed description of the available devices.

The specific device is identified by its address. For example, since every controller has multiple inputs you need to specify both the device name and the address in order to read a specific input.

4.2 Bits, Bytes and Words

As in all digital technology, the smallest unit of information in a PLC is a “bit”. A bit can only have two states: “0” (OFF or FALSE) and “1” (ON or TRUE). PLCs have a number of so-called **bit devices** that can only have two states, including inputs, outputs and relays.

The next larger information units are the “byte”, which consists of 8 bits, and the “word”, which consists of two bytes. In the MELSEC System Q range of PLCs the data registers are “word devices”, which means that they can store 16-bit values.



Since a data register is 16 bits wide it can store signed values between -32,768 and +32,767 (see next chapter 4.3). When larger values need to be stored two words are combined to form a 32-bit long word, which can store signed values between -2,147,483,648 and +2,147,483,647.

4.3 Number Systems

The PLCs of the MELSEC System Q use several different number systems for inputting and displaying values and for specifying device addresses.

Decimal numbers

The decimal number system is the system we use most commonly in everyday life. It is a “positional base 10” system, in which each digit (position) in a numeral is ten times the value of the digit to its right. After the count reaches 9 in each position the count in the current position is returned to 0 and the next position is incremented by 1 to indicate the next decade (9 → 10, 99 → 100, 199 → 1,000 etc).

– Base: 10

– Digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

In the MELSEC System Q family of PLCs decimal numbers are used for entering constants and the setpoint values for timers and counters. Device addresses are also entered in decimal format, with the exception of the addresses of inputs and outputs.

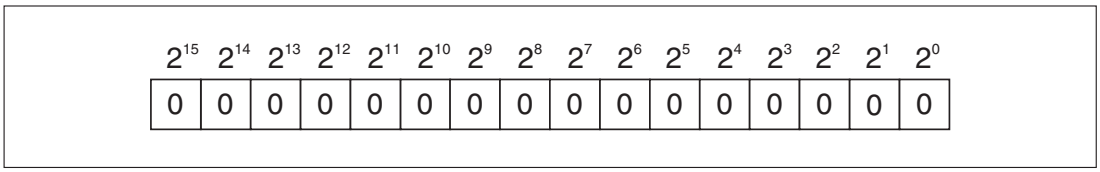
Binary numbers

Like all computers a PLC can only really distinguish between two states, ON/OFF or 0/1. These “binary states” are stored in individual bits. When numbers need to be entered or displayed in other formats the programming software automatically converts the binary numbers into the other number systems.

– Base: 2

– Digits: 0 and 1

When binary numbers are stored in a word (see above) the value of each digit (position) in the word is one power of 2 higher than that of the digit to its right. The principle is exactly the same as in decimal representation, but with increments of 2 instead of 10 (see graphic):



Base 2 Notation	Decimal Value	Base 2 Notation	Decimal Value
2^0	1	2^8	256
2^1	2	2^9	512
2^2	4	2^{10}	1024
2^3	8	2^{11}	2048
2^4	16	2^{12}	4096
2^5	32	2^{13}	8192
2^6	64	2^{14}	16384
2^7	128	2^{15}	32768*

* In binary values bit 15 is used to represent the sign (bit 15=0: positive value, bit 15=1: negative value)

To convert a binary value to a decimal value you just have to multiply each digit with a value of 1 by its corresponding power of 2 and calculate the sum of the results.

Example

00000010 00011001 (binary)

00000010 00011001 (binary) = 1 x 2^9 + 1 x 2^4 + 1 x 2^3 + 1 x 2^0
00000010 00011001 (binary) = 512 + 16 + 8 + 1
00000010 00011001 (binary) = 537 (decimal)

Hexadecimal numbers

Hexadecimal numbers are easier to handle than binary and it is very easy to convert binary numbers to hexadecimal. This is why hexadecimal numbers are used so often in digital technology and programmable logic controllers. In the PLCs of the MELSEC System Q hexadecimal numbers are used for the numbering of inputs and outputs and the representation of constants. In the programming manual and other manuals hexadecimal numbers are always identified with an H after the number to avoid confusion with decimal numbers (e.g. 12345H).

– Base: 16

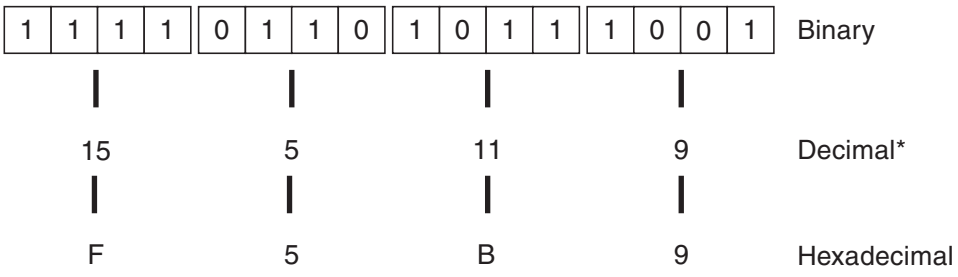
– Digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
(the letters A, B, C, D, E and F represent the decimal values 10, 11, 12, 13, 14 and 15)

The hexadecimal system works in the same way as the decimal system – you just count to FH (15) instead of to 9 before resetting to 0 and incrementing the next digit (FH → 10H, 1FH → 20H, 2FH → 30H, FFH → 100H etc). The value of digit is a power of 16, rather than a power of 10:

1A7FH

16^0 = 1	(in this example: 15 x 1	= 15)
16^1 = 16	(in this example: 7 x 16	= 112)
16^2 = 256	(in this example: 10 x 256	= 2560)
16^3 = 4096	(in this example: 1 x 4096	= 4096)
		6783 (decimal)

The following example illustrates why it is so easy to convert binary values hexadecimal values:



* Converting the 4-bit blocks to decimal values does not directly produce a value that corresponds to the complete 16-bit binary value! In contrast, the binary value can be converted directly to hexadecimal notation with exactly the same value as the binary value.

Octal numbers

Octal numbers are listed here for the sake of completeness only. They are not used in a PLC of the MELSEC System Q. In the octal system the digits 8 and 9 don`t exist. Here, the current digit is reset to 0 and the digit in the next position is incremented after the count reaches 7 (0 – 7, 10 – 17, 70 – 77, 100 – 107 etc).

– Base: 8

– Digits: 0, 1, 2, 3, 4, 5, 6, 7

Summary

The following table provides an overview of the four different number systems:

Decimal notation	Octal notation	Hexadecimal notation	Binary notation
0	0	0	0000 0000 0000 0000
1	1	1	0000 0000 0000 0001
2	2	2	0000 0000 0000 0010
3	3	3	0000 0000 0000 0011
4	4	4	0000 0000 0000 0100
5	5	5	0000 0000 0000 0101
6	6	6	0000 0000 0000 0110
7	7	7	0000 0000 0000 0111
8	10	8	0000 0000 0000 1000
9	11	9	0000 0000 0000 1001
10	12	A	0000 0000 0000 1010
11	13	B	0000 0000 0000 1011
12	14	C	0000 0000 0000 1100
13	15	D	0000 0000 0000 1101
14	16	E	0000 0000 0000 1110
15	17	F	0000 0000 0000 1111
16	20	10	0000 0000 0001 0000
:	:	:	:
99	143	63	0000 0000 0110 0011
:	:	:	:

4.4 Codes

For safe und efficient data communication the letters of the alphabet and the decimal numbers must be converted into a code that a machine can understand.

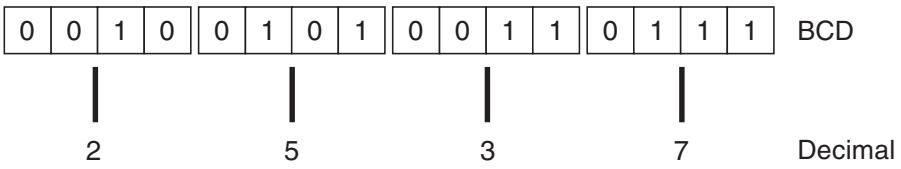
4.4.1 BCD Code

Binary-coded decimal (BCD) is an encoding for decimal numbers in which each digit (0 to 9) is represented by a 4-bit binary number (0000 to 1001, see table below). Thus one byte (8 bits) can store two decimal numbers.

Decimal	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

To convert decimal numbers with more than one digit the BCD expressions of the individual digits are linked together. A four digit number in BCD code occupies one word (16 Bit) and can cover the range from 0000 to 9999.

Example



In the MELSEC System Q the BCD code is not used for internal operations. However, in industrial applications BCD is frequently used to input values or to display numbers on a LED display. For those cases several instructions for converting to and from BCD code are available.

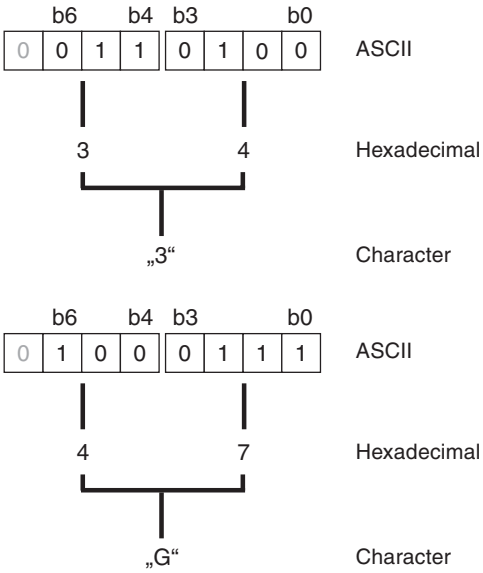
4.4.2 ASCII Code

ASCII is the short form for ***A**merican **S**tandard **C**ode for **I**nformation **I**nterchange*. In ASCII code 7 bits can represent alphanumeric characters and also punctuation marks, miscellaneous symbols and control characters.

Data in ASCII code is used to communicate with peripheral devices.

Bits 3 to 0		Bits 6 to 4							
		0	1	2	3	4	5	6	7
		000	001	010	011	100	101	110	111
0	0000	NUL	DLE	SP	0	@	P	′	p
1	0001	SOH	DC1	!	1	A	Q	a	q
2	0010	STX	DC2	!!	2	B	R	b	r
3	0011	ETX	DC3	#	3	C	S	c	s
4	0100	EOT	DC4	$	4	D	T	d	t
5	0101	ENQ	NAK	%	5	E	U	e	u
6	0110	ACK	SYN	&	6	F	V	f	v
7	0111	BEL	ETB	‘	7	G	W	g	w
8	1000	BS	CAN	(	8	H	X	h	x
9	1001	HT	EM	)	9	I	Y	i	y
A	1010	LF	SUB	*	:	J	Z	j	z
B	1011	VT	ESC	+	;	K	[	k	{
C	1100	FF	FS	,	<	L	\	l	|
D	1101	CR	GS	-	=	M	]	m	}
E	1110	SO	RS	.	>	N	↑	n	~
F	1111	SI	VS	/	?	O	←	o	DEL

Examples



4.5 Programming Languages

GX IEC Developer provides separate editors for programming. You can choose between graphical input and display of the programs and input and display as text. With the exception of the Sequential Function Chart language, all the editors divide PLC programs into sections, referred to as "Networks".

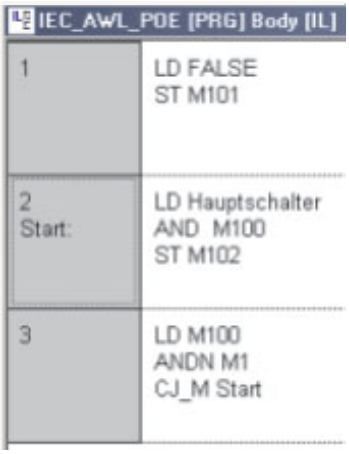
4.5.1 Text Editors

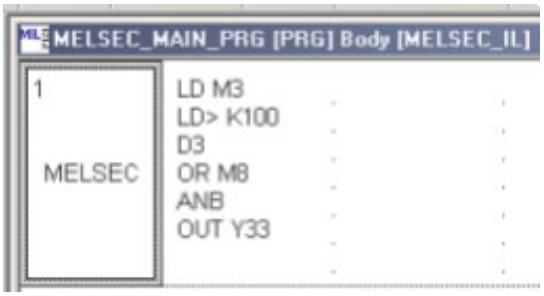
Instruction List (IL)

The Instruction List (IL) work area is a simple text editor with which the instructions are entered directly. Each instruction must contain an operator (function) and one or more operands. Each instruction must begin in a new line.

Two different types of Instruction List are used:

- IEC Instruction List





- MELSEC Instruction List

In a MELSEC Instruction List you can only use the MELSEC instruction set; IEC standard programming is not possible.

Structured Text (ST)

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.

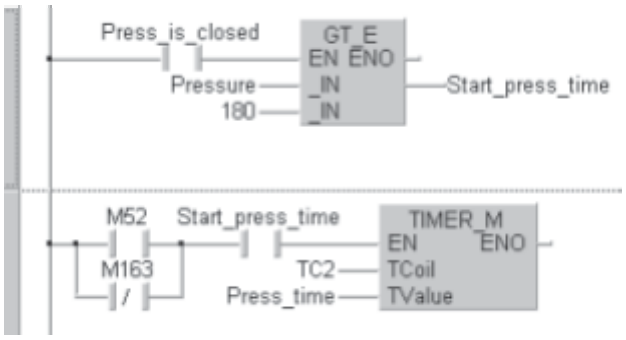
```
(*Example showing Structured Text*)

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

4.5.2 Graphic Editors

Ladder Diagram

Programming in Ladder Diagram is very similar to drawing a circuit diagram for conventional relay systems. A Ladder Diagram consists of input contacts (makers and breakers), output coils but also 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.



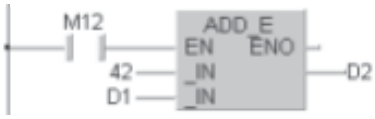
Example for Ladder Diagram

For the most frequently applied instructions in Ladder Diagram buttons are available in the toolbar.

More complex function and function blocks are displayed as boxes in a Ladder Diagram. In addition to the inputs and output necessary for the function function and function blocks have an EN input and an ENO output. The EN input (EN = ENable) controls the execution of the instruction.

This instruction is executed cyclical.

This instruction is only executed when M12 is ON.



The result of the operation is passed to the ENO output (ENO = ENable Out).

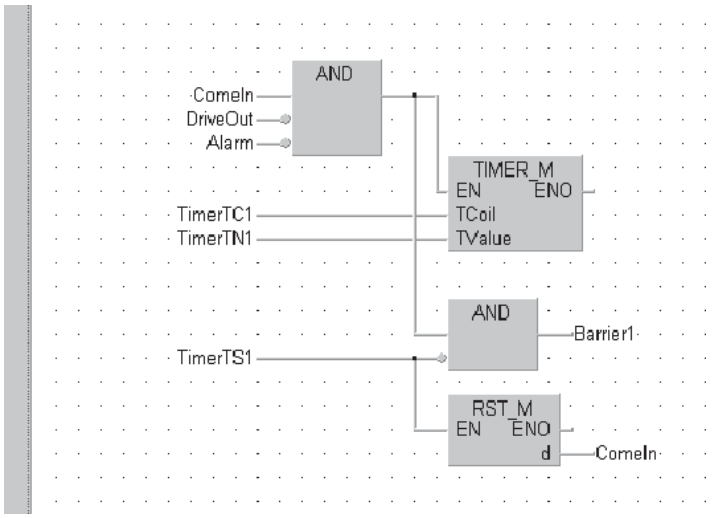
M34 is set when the contents of the two devices compared with the compare instruction are identical.

To control the program flow, ENO outputs and EN inputs can be connected. In the following example the execution of the second instruction depends on the result of the first instruction.

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.

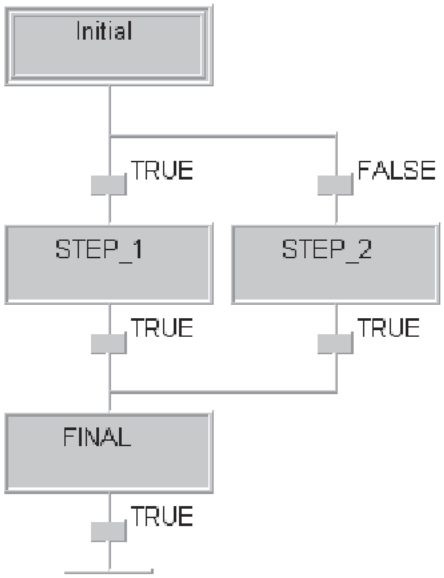
Example for Function Block Diagram:



Sequential Function Chart

Sequential Function Chart (SFC) is a structured language which allows clear representation of complex processes.

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.



4.6 The IEC 61131-3 Standard

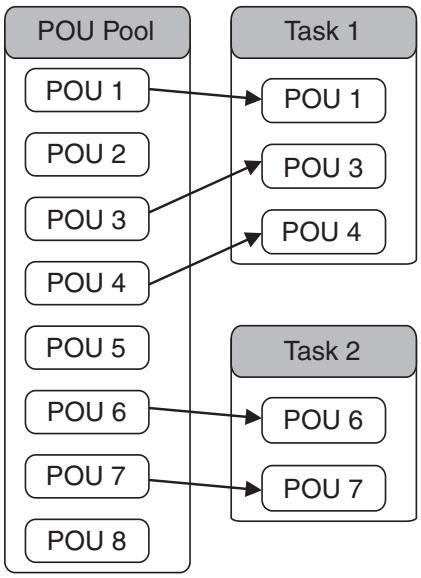
IEC 61131-3 is the international standard for PLC programs, defined by the International Electromechanical Commission (IEC). The IEC 61131-3 not only covers PLC programming languages but also offers guidelines for PLC applications.With the software GX IEC Developer PLCs can be programmed according to the IEC 61131-3 Standard.

In this Beginner's Manual only the terms which are necessary for understanding the program examples are explained. For further information about GX IEC Developer please refer to the Beginner's Manual (art. no. 043596) or the Reference Manual (art. no. 043597) for this software. During programming you can also use the Help function of GX IEC Developer.

4.6.1 Software Structure

Program Organisation Unit (POU)

In IEC 61131-3 a PLC program is divided into individual program modules called Program Organisation Units (POUs). A POU is the smallest independent element of a sequence program.



POUs are stored in a so called POU Pool.

Program POUs are grouped together in Task.

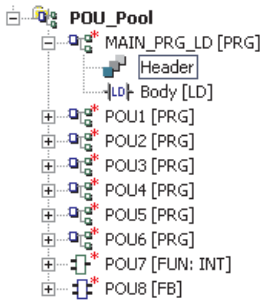
Individual Tasks are grouped together to form the actual PLC program.

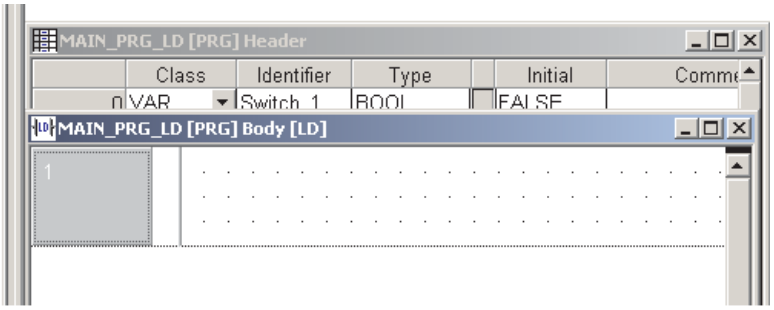
Every POU consists of

- Header
- and Body

In the ***Header*** the variables which are used in this POU are declared.

The ***Body*** is the part of the project where the program is edited.Several languages are available for editing a program.



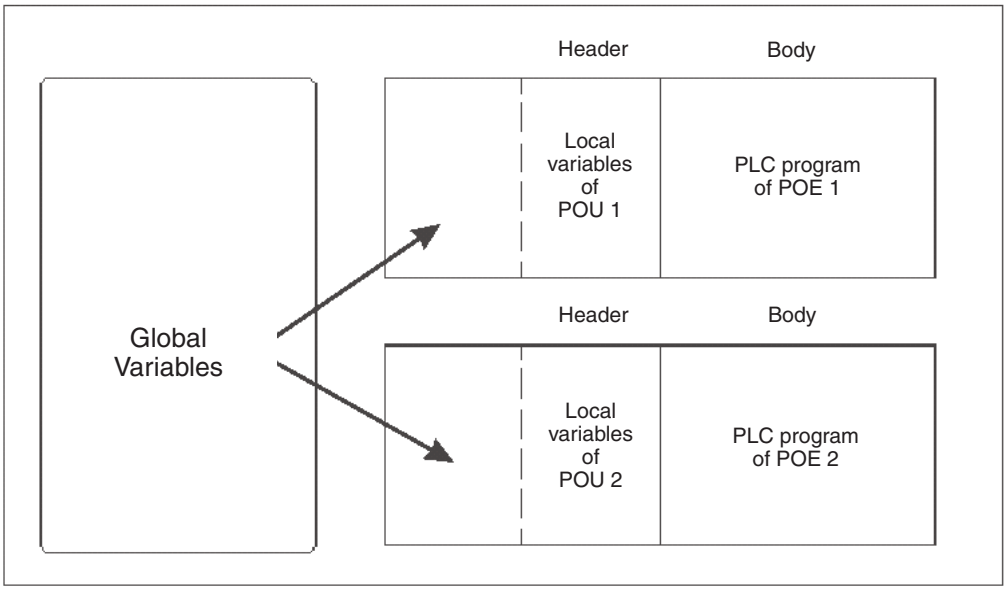


4.6.2 Variables

Variables contain the values of inputs, outputs, or the internal devices of the PLC.A distinction is made between two different types of variables:

- Global variables and
- Local 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. Global Variables make it possible to exchange data between the individual POUs.



For a particular POU to access a Global Variable, it must be declared in the Header of this POU. The Header can be made up of both Global Variables and Local Variables.

A **Local Variable** can be thought of as an intermediate result. A POUs can not access Local Variables of other POUs.

Declaring Variables

At the start of each POE the variables are declared, i.e. they are assigned to a certain data type like INT or BOOL.

Each variable has the following elements:

- Class
- Identifier, the name of the variable
- Absolute address (optional for global variables)
- Data type
- Inital value (is specified automatically)
- Comment (optional)

Global Variable List

	Class	Identifier	MIT-Addr.	IEC-Addr.	Type		Initial	Comment
0	VAR_GLOBAL	START_MOTOR	X10	%IX16	BOOL	...	FALSE	
1	VAR_GLOBAL	STOP_MOTOR	X11	%IX17	BOOL	...	FALSE	
2	VAR_GLOBAL	MOTOR_1_ON	Y20	%QX32	BOOL	...	FALSE	

Class

The class keyword assigns the variable a property that defines how it is to be used in the project. Some examples:

– VAR: Local variable for use within the POU

– VAR_EXTERNAL: An external global variable is declared in the Global Variable List and can be read and written by all POUs.

– VAR_CONSTANT: Local variable with constant value for use within the POU.

Identifier

Each variable is given a symbolic address. This individual name (identifier) can be chosen freely but must always begin with a letter or a single underline character. Spaces and mathematical operator characters (e.g. +, - ,*) are not permitted.

Examples for identifiers:

– S02.3

– Drive_2_ready

– _Open_Valve

– Motor_M1_ON

The use of symbolic declarations complies with IEC 61131.3.

Absolute addresses

When global variables are declared they should also be assigned absolute addresses. If you do not assign the absolute addresses manually, they are assigned automatically. An absolute address specifies the memory location of the variable in the CPU or an input or output.

Absolute addresses can be assigned using either the IEC syntax (IEC-Adr.) or the MELSEC syntax (MIT-Addr.). Some examples for absolute addresses:

Input X0F = X0F (MELSEC syntax) = %IX15 (IEC syntax)

Output Y03 = Y03 (MELSEC syntax) = %QX3 (IEC syntax)

Elementary Data Types

The data type defines the characteristics of a variable like value range or number of bits.

Data type		Value range	Size
BOOL	Boolean	0 (FALSE), 1 (TRUE)	1 Bit
INT	Integer	-32768 to +32767	16 Bits
DINT	Double Integer	-2,147,483,648 to 2,147,483,647	32 Bits
WORD	Bit string 16	0 to 65535	16 Bits
DWORD	Bit string 32	0 to 4.294.967.295	32 Bits
REAL	Floating point value	3.4E +/-38 (7 digits)	
TIME	Time value	-T#24d0h31m23s64800ms to T#24d20h31m23s64700ms	
STRING	Character string	Character strings are limited to16 characters	

4.7 The Basic Instruction Set

The instructions of the PLCs of the MELSEC System Q can be divided into two basic categories, basic instructions and applied instructions, which are sometimes referred to as “application instructions”.

The functions performed by the basic instructions are comparable to the functions achieved by the physical wiring of a hard-wired controller.

Basic instruction set quick reference

Instruction	Function	Description	Reference
LD	Load	Initial logic operation, polls for signal state “1” (normally open)	Chapter 4.7.1
LDI	Load invers	Initial logic operation, polls for signal state “0” (normally closed)	
OUT	Output instruction	Assigns the result of a logic operation to a device	Chapter 4.7.2
AND	Logical AND	Logical AND operation, polls for signal state “1”	Chapter 4.7.4
ANI	AND NOT	Logical AND NOT operation, polls for signal state “0”	
OR	Logical OR	Logical OR operation, polls for signal state “1”	Chapter 4.7.5
ORI	OR NOT	Logical OR NOT operation, polls for signal state “0"	
ANB	AND Block	Connects a parallel branch circuit block to the preceding parallel block, in series.	Chapter 4.7.6
ORB	OR Block	Connects a serial block of circuits to the preceding serial block, in parallel.	
LDP	Pulse signal instructions	Load Pulse, load on detection of rising edge of device signal pulse	Chapter 4.7.7
LDF		Load Falling Pulse, load on falling device signal pulse	
ANDP		AND Pulse, logical AND on rising device signal pulse	
ANDF		AND Falling Pulse, logical AND on falling device signal pulse	
ORP		OR Pulse, logical OR on rising device signal pulse	
ORF		OR Falling Pulse, logical OR on falling device signal pulse	
SET	Set device	Assigns a signal state that is retained even if after input condition is no longer true	Chapter 4.7.8
RST	Reset device		
PLS	Pulse instructions	Pulse, sets a device for one operation cycle on the rising pulse of the input condition (input turns ON)	Chapter 4.7.9
PLF		Pulse Falling, sets a device for one operation cycle on the falling pulse of the input condition (input turns OFF)	
INV	Invert	Inverts the result of an operation	Chapter 4.7.10
FF	Inversion of bit	Inversion of bit output device	Chapter 4.7.11
MEP	Result into pulse conversion	Pulse generation at the rising edge of the operation result.	Chapter 4.7.12
MEF		Pulse generation at the falling edge of the operation result.	

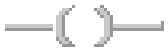
4.7.1 Starting logic operations

Instruction	Function	Ladder Diagram	IEC Instruction List
LD	Load instruction, starts a logic operation and polls the specified device for signal state “1”		LD
LDI	Load instruction, starts a logic operation and polls the specified device for signal state “0”		LDN

A circuit in a program always begins with an LD- or LDI instruction. These instructions can be performed on inputs, relays, timers and counters.

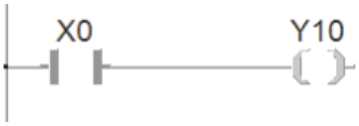
For examples of using these instructions see the description of the OUT instruction in the next section.

4.7.2 Outputting the result of a logic operation

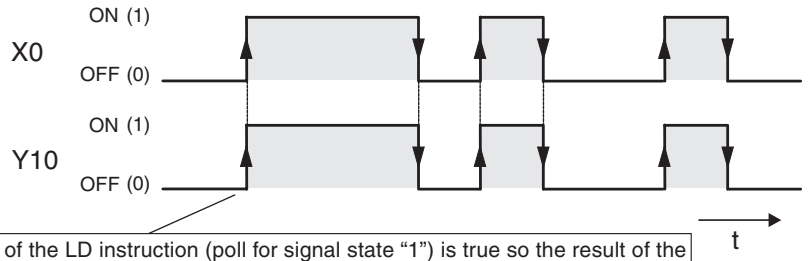
Instruction	Function	Ladder Diagram	IEC Instruction List
OUT	Output instruction, assigns the result of an operation to a device		ST

The OUT instruction can be used to terminate a circuit. You can also program circuits that use multiple OUT instructions as their result. This is not necessarily the end of the program, however. The device set with the result of the operation using OUT can then be used as an input signal state in subsequent steps of the program.

Example (LD and OUT instructions)

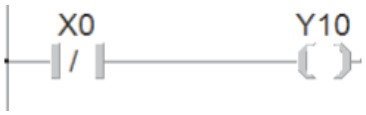
Ladder Diagram	MELSEC Instruction List		IEC Instruction List	
	LD	X0	LD	X0
	OUT	Y10	ST	Y10

These two instructions result in the following signal sequence:



The condition of the LD instruction (poll for signal state “1”) is true so the result of the operation is also true (“1”) and the output is set.

Example (LDI and OUT instructions)

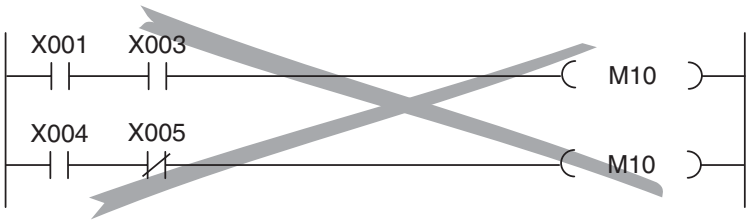
Ladder Diagram	MELSEC Instruction List		IEC Instruction List	
	LDI OUT	X0 Y10	LDI ST	X0 Y10

The condition of the LDI instruction (poll for signal state “0”) is no longer true so the output is reset.

NOTE **Double assignment of relays or outputs**

The result of an operation should not assigned to the same device in more than one place in the program!

The program is executed sequentially from top to bottom, so in this example the second assignment of M10 would simply overwrite the result of the first assignment.



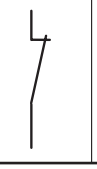
You can solve this problem with modification shown on the right. This takes all the required input conditions into account and sets the result correctly.

But there is an exception for every rule! You can take advantage of the top-to-bottom-execution of a PLC program and place instructions with high priority at the end of the program to intentionally overwrite previous results. An example is shown in chapter 4.9.1. Here safety facilities are used to reset internal devices of the PLC and bring a motor to a stop. But the outputs for the motor are only assigned once in the whole program!

4.7.3 Using switches and sensors

Before we continue with the description of the rest of the instructions we should first describe how signals from switches, sensors and so on can be used in your programs.

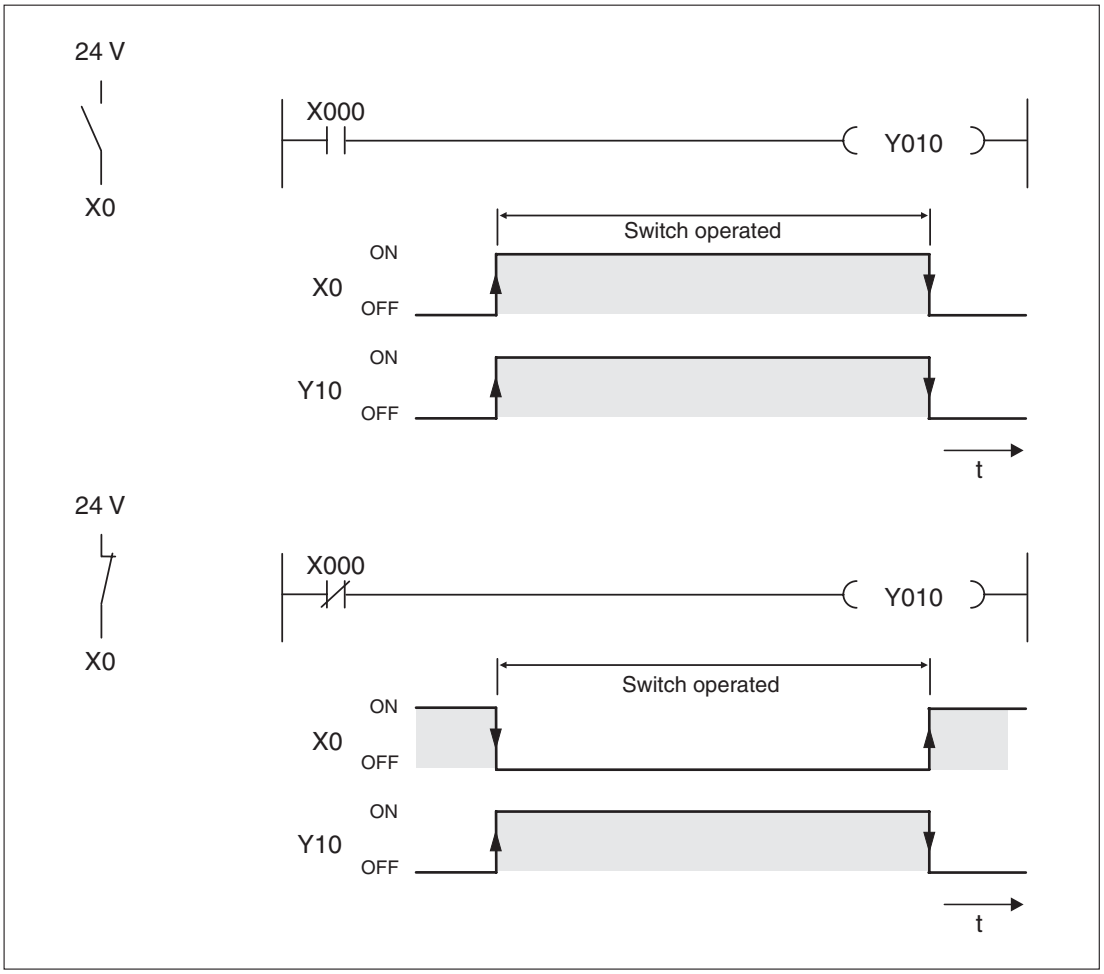
PLC programs need to be able respond to signals from switches, buttons and sensors to perform the correct functions. It is important to understand that program instructions can only poll the binary signal state of the specified input – irrespective of the type of input and how it is controlled.

	Normally open contact (make contact)	When a make normally open contact is operated the input is set (ON, signal state “1”)
	Normally closed contact (break contact)	When a normally closed contact is operated the input is reset (OFF, signal state “0”)

As you can imagine, this means that when you are writing your program you need to be aware whether the element connected to the input of your PLC is a make or a break device. An input connected to a make device must be treated differently to an input connected to a break device. The following example illustrates this.

Usually, switches with make contacts are used. Sometimes, however, break contacts are used for safety reasons – for example for switching off drives (see section 4.8).

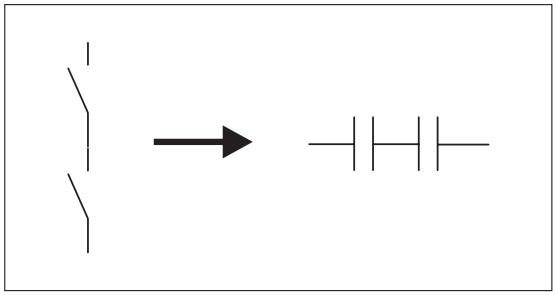
The illustration below shows two program sequences in which the result is exactly the same, even though different switch types are used: When the switch is operated the output is set (switched on).



4.7.4 AND operations

Instruction	Function	Ladder Diagram	IEC Instruction List
AND	Logical AND (AND operation with poll for signal state “1” or ON)		AND
ANI	Logical AND NOT (AND operation with poll for signal state “0” or OFF)		ANDN

An AND operation is logically the same as a serial connection of two or more switches in an electrical circuit. Current will only flow if all the switches are closed. If one or more of the switches are open no current flows – the AND condition is false.

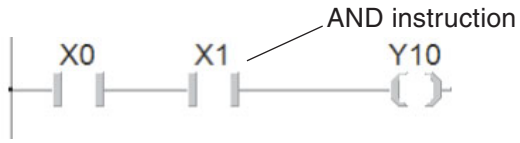


Note that the programming software uses the same icons and function keys for the AND and ANI instructions as for the LD and LDI instructions. When you program in Ladder Diagram format the software automatically assigns the correct instructions on the basis of the insertion position.

When you program in Instruction List format remember that you can’t use the AND and ANI instructions at the beginning of circuit (a program line in ladder diagram format)! Circuits must begin with an LD or LDI instruction (see Chapter 4.7.1).

Example of an AND instruction

Ladder Diagram



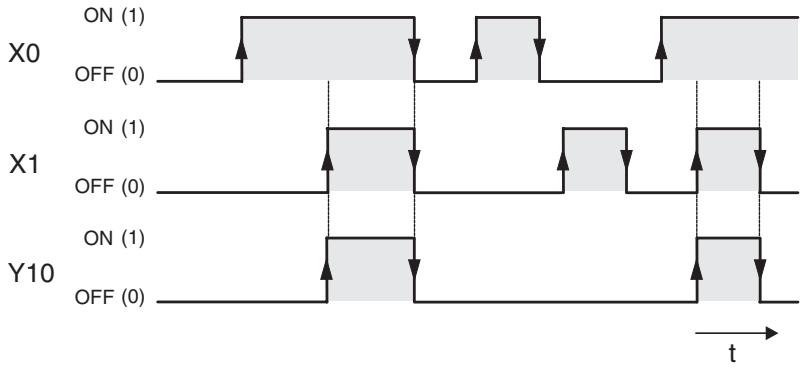
MELSEC Instruction List

```
LD    X0
AND   X1
OUT   Y10
```

IEC Instruction List

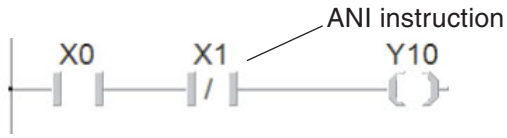
```
LD    X0
AND   X1
ST    Y10
```

In the example output Y10 is only switched on when inputs X0 **and** X1 are **both** on:



Example of an ANI instruction

Ladder Diagram



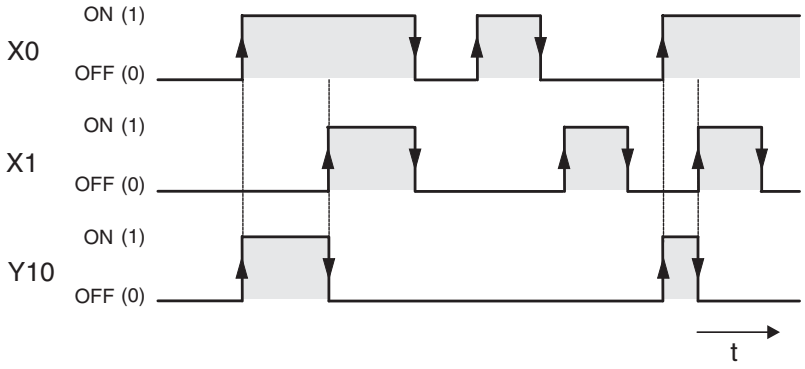
MELSEC Instruction List

```
LD    X0
ANI   X1
OUT   Y10
```

IEC Instruction List

```
LD    X0
ANDN  X1
ST    Y10
```

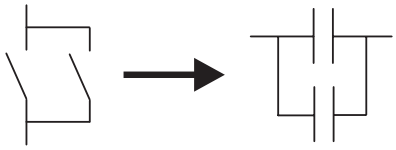
In the example output Y10 is only switched on when input X0 is on **and** input X1 is off:



4.7.5 OR operations

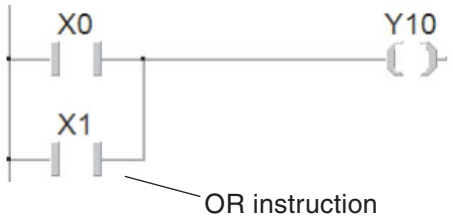
Instruction	Function	Ladder Diagram	IEC Instruction List
OR	Logical OR (OR operation with poll for signal state “1” or ON)		OR
ORI	Logical OR NOT (OR operation with poll for signal state “0” or OFF)		ORN

An OR operation is logically the same as the parallel connection of multiple switches in an electrical circuit. As soon as any of the switches is closed current will flow. Current will only stop flowing when **all** the switches are open.



Example of an OR instruction

Ladder Diagram



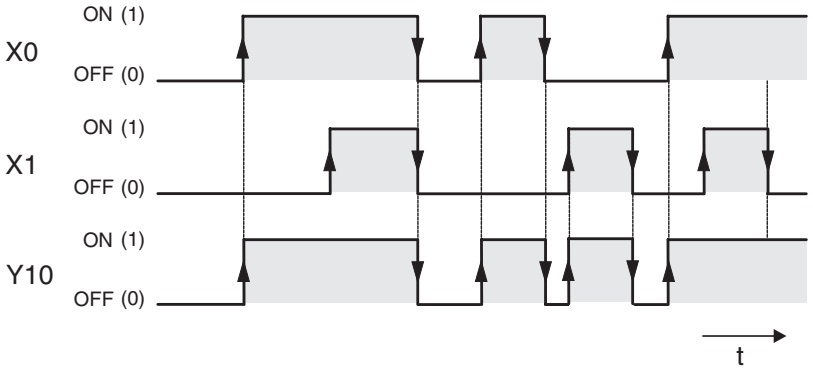
MELSEC Instruction List

LD	X0
OR	X1
OUT	Y10

IEC Instruction List

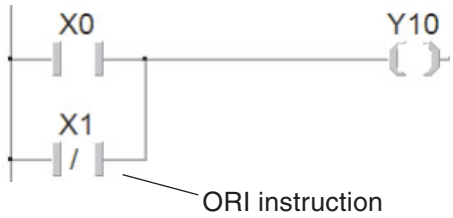
LD	X0
OR	X1
ST	Y10

In the example output Y10 is switched on when **either** input X0 **or** input X1 is on:



Example of an ORI instruction

Ladder Diagram



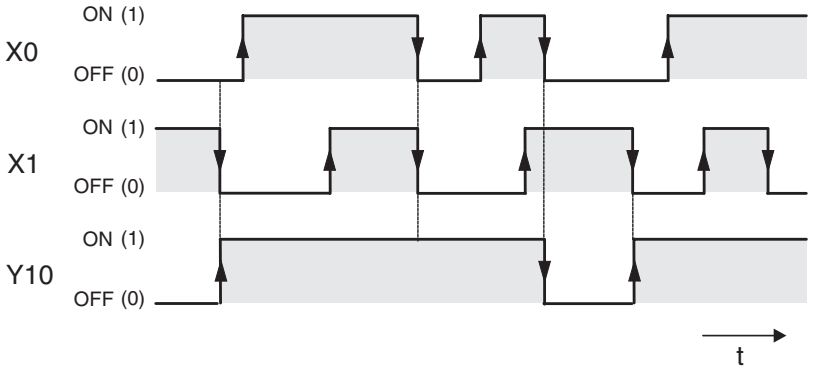
MELSEC Instruction List

LD	X0
ORI	X1
OUT	Y10

IEC Instruction List

LD	X0
ORN	X1
ST	Y10

In the example output Y10 is switched on when **either** input X0 is on **or** input X1 is off:



4.7.6 Instructions for connecting operation blocks

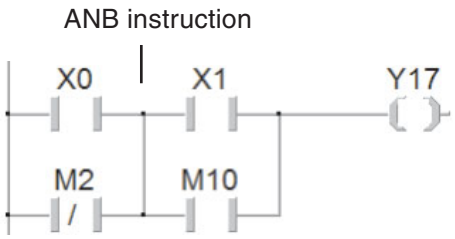
Instruction	Function	Ladder Diagram	IEC Instruction List
ANB	AND Block (serial connection of blocks of parallel operations/circuits)	———	AND (...)
ORB	OR Block (parallel connection of blocks of serial operations/circuits)	|	OR (...)

Although ANB- and ORB are PLC instructions they are only displayed and entered as connecting lines in the Ladder Diagram display. They are only shown as instructions in Instruction List format, where you must enter them with their acronyms ANB and ORB.

Both instructions are entered without devices and can be used as often as you like in a program. However, the maximum number of LD and LDI instructions is restricted to 15, which effectively also limits the number of ORB or ANB instructions you can use before an output instruction to 15 as well.

Example of an ANB instruction

Ladder Diagram



MELSEC Instruction List

```
LD      X0
ORI     M2      ← 1st parallel connection (OR operation)
LD      X1
OR      M10     ← 2nd parallel connection (OR operation)
ANB             ← ANB instruction connecting both OR operations
OUT     Y17
```

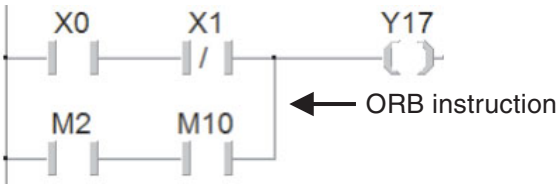
IEC Instruction List

```
LD      X0
ORN     M2      ← 1st parallel connection (OR operation)
AND(            ← ANB instruction connecting both OR operations
        X1
OR      M10     ← 2nd parallel connection (OR operation)
)
ST      Y017
```

In this example output Y17 is switched on if input X00 is “1” **and** input X01 is “0”, **or** if relay M2 is “0” **and** relay M10 is “1”.

Example of an ORB instruction

Ladder Diagram



MELSEC Instruction List

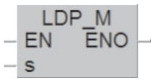
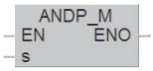
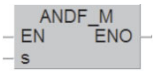
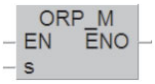
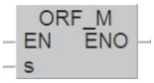
LD	X0	
ANI	X1	← 1st serial connection (AND operation)
LD	M2	
AND	M10	← 2nd serial connection (AND operation)
ORB		← ORB instruction connecting both AND operations
OUT	Y17	

IEC Instruction List

LD	X0	
ANDN	X1	← 1st serial connection (AND operation)
OR(		← ORB instruction connecting both AND operations
	M2	
AND	M10	← 2nd serial connection (AND operation)
)		
ST	Y17	

In this example output Y17 is switched on if input X00 is “1” **and** input X01 is “0”, **or** if relay M2 is “0” **and** relay M10 is “1”.

4.7.7 Pulse-triggered execution of operations

Instruction	Function	Ladder Diagram	IEC Instruction List
LDP	Load Pulse, loads on the rising edge of the device's signal		—
LDF	Load Falling Pulse, loads on the falling edge of the device's signal		—
ANDP	AND Pulse, logical AND operation on the rising edge of the device's signal		ANDP_M
ANDF	AND Falling Pulse, logical AND operation on the falling edge of the device's signal		ANDF_M
ORP	OR Pulse, logical OR operation on the rising edge of the device's signal		ORP_M
ORF	OR Falling Pulse, logical OR operation on the falling edge of the device's signal		ORF_M

In PLC programs you will often need to detect and respond to the rising or falling edge of a bit device's switching signal. A rising edge indicates a switch of the device value from "0" to "1", a falling edge indicates a switch from "1" to "0".

During program execution operations that respond to rising and falling pulses only deliver a value of "1" when the signal state of the referenced device changes.

When do you need to use this? For example, suppose you have a conveyor belt with a sensor switch that activates to increment a counter every time a package passes it on the belt. If you don't use a pulse-triggered function you will get incorrect results because the counter will increment by 1 in every program cycle in which the switch registers as set. If you only register the rising pulse of the switch signal the counter will be incremented correctly, increasing by 1 for each package.

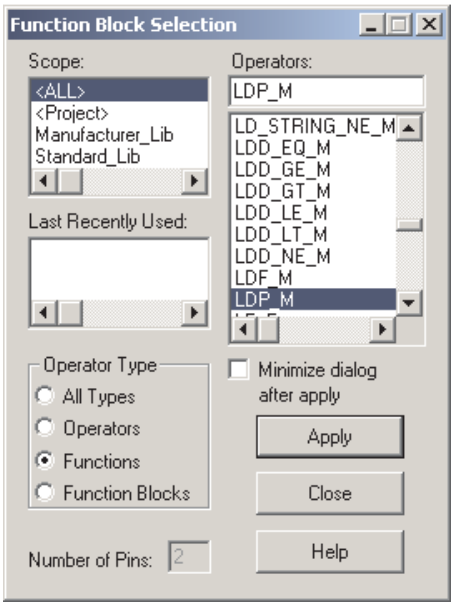
NOTE Most applied instructions can also be executed by pulse signals. For details see chapter 6.

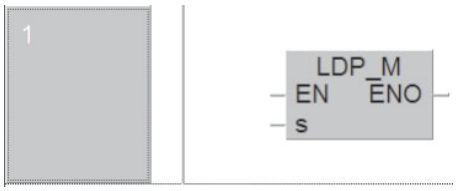
Entering of functions and function blocks into Ladder Diagram

The pulse-triggered instructions and other more complex instructions can not be entered with buttons in the toolbar of GX IEC Developer. These instructions are entered by selecting them in the function block selection window.

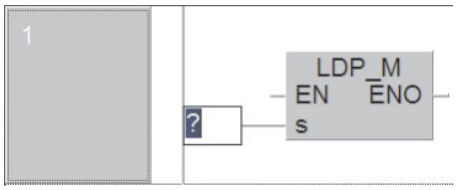
Click on the Function / Function block  selection button on the tool bar. This opens the function block selection window shown below.

On the ***Operator type*** click ***Functions*** and choose for example the instruction **LDP_M** from the list.

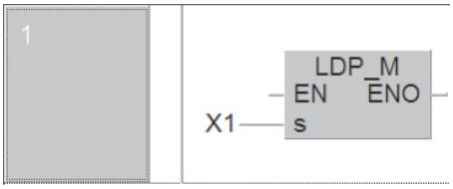




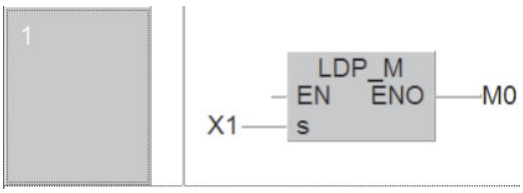
Click on ***Apply*** or double click on the selected object and then click in the body of the POU to place the function.



Click on the button  (Input Variable) in the tool bar and afterwards on the input of the function where a device should be entered.



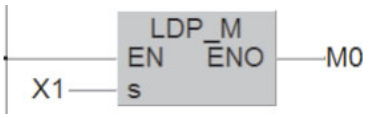
Type the input device and press the ENTER key.



To enter an variable to the output of the function click on the button  in the tool bar and then on the ENO output.

Evaluating a rising signal pulse

Ladder Diagram

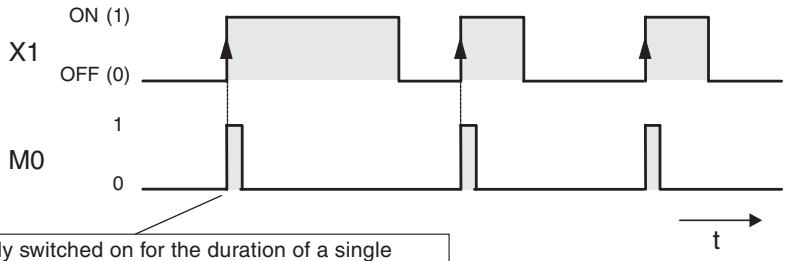


MELSEC Instruction List

```
LDP   X1
OUT   M0
```

IEC Instruction List

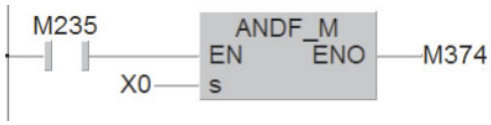
```
LD    X1
PLS_M M0
```



Relay M0 is only switched on for the duration of a single program cycle

Evaluating a falling signal pulse

Ladder Diagram

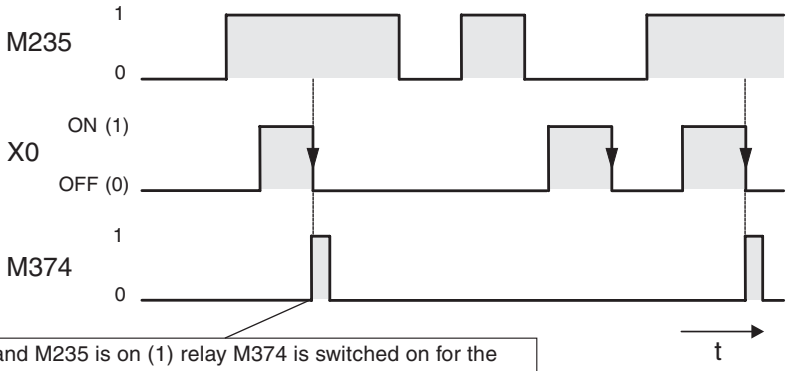


MELSEC Instruction List

```
LD    M235
ANDF  X0
OUT   M374
```

IEC Instruction List

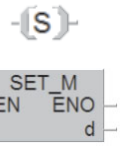
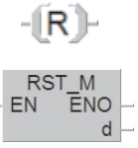
```
LD    M235
ANDF_M X0
ST    M374
```



If X0 is off (0) and M235 is on (1) relay M374 is switched on for the duration of a single program cycle.

With the exception of the pulse trigger characteristic the functions of the LDP, LDF, ANDP, ANDF, ORP and ORF instructions are identical to those of the LD, AND and OR instructions. This means that you can use pulse-trigger operations in your programs in exactly the same way as the conventional versions.

4.7.8 Setting and resetting devices

Instruction	Function	Ladder Diagram	IEC Instruction List
SET	Set a device①, (assign signal state "1")		S
RST	Reset a device②, (assign signal state "0")		R

① The SET instruction can be used to set outputs (Y), relays (M) and state relays (S).

② The RST instruction can be used to reset outputs (Y), relays (M), state relays (S), timers (T), counters (C) and registers (D, V, Z).

The signal state of an OUT instruction will normally only remain "1" as long as the result of the operation connected to the OUT instruction evaluates to "1". For example, if you connect a pushbutton to an input and a lamp to the corresponding output and connect them with an LD and an OUT instruction the lamp will only remain on while the button remains pressed.

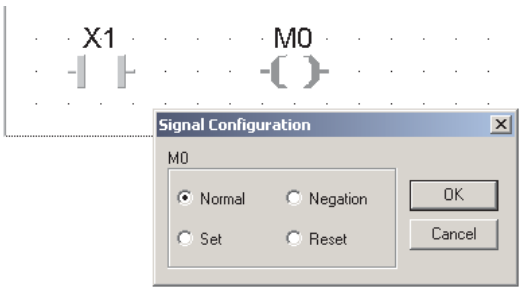
The SET instruction can be used to use a brief switching pulse to switch an output or relay on (set) and leave them on. The device will then remain on until you switch it off (reset) with a RST instruction. This enables you to implement latched functions or switch drives on and off with pushbuttons. (Outputs are generally also switched off when the PLC is stopped or the power supply is turned off. However, some relays also retain their last signal state under these conditions – for example a set relay would then remain set.)

In Ladder Diagram SET and RST instructions can be programmed within an output operation or as a function.

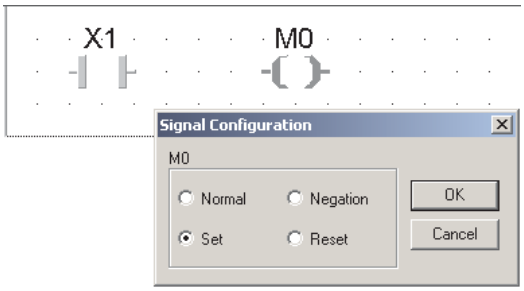
OUT instruction with SET or RST functionality

Program an OUT instruction and enter the device to be set or reset.

Double click on the OUT instruction. The ***Signal configuration*** window is displayed.



For a SET instruction click on ***Set***. If you want a RST instruction click on ***Reset***. Then click on ***OK*** to close the window.

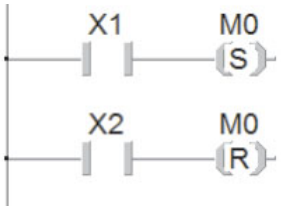


This concludes the conversion of an OUT instruction to a SET instruction.

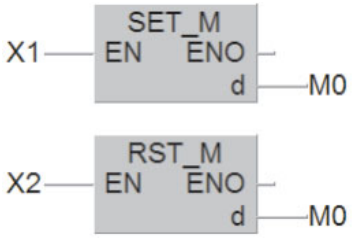
Examples for setting and resetting devices

Ladder Diagram

1st alternative



2nd alternative



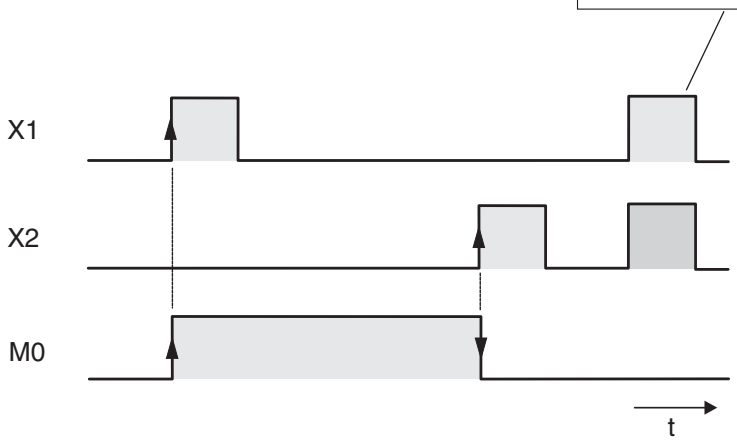
MELSEC Instruction List

LD	X1
SET	M0
LD	X2
RST	M0

IEC Instruction List

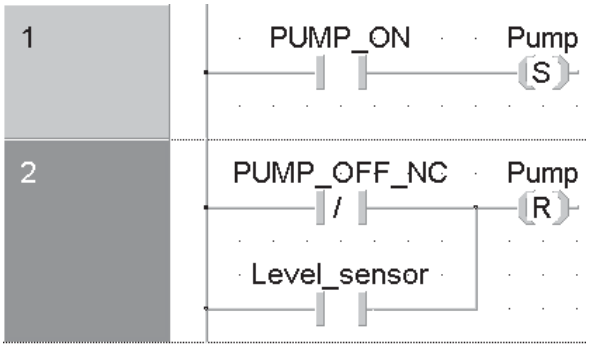
LD	X1
S	M0
LD	X2
R	M0

If the set and reset instructions for the same device both evaluate to “1” the last operation performed has priority. In this example that is the RST instruction, and so M0 remains off.



This example is a program for controlling a pump to fill a container. The pump is controlled manually with two pushbuttons, ON and OFF. For safety reasons a break contact is used for the OFF function. When the container is full a level sensor automatically switches the pump off.

Ladder Diagram



MELSEC Instruction List

```
LD      Pump_ON
SET     Pump
LDI     Pump_OFF_NC
OR      Level_sensor
RST     Pump
```

IEC Instruction List

```
LD      Pump_ON
S       Pump
LDN     Pump_OFF_NC
OR      Level_sensor
R       Pump
```

NOTE

To display the devices with there identifiers it is necessary to declare them as variables in the Global Variable List. Below the Global Variable List for this program example is shown:

	Class	Identifier	MIT-Addr.	IEC-Addr.	Type
0	VAR_GLOBAL	PUMP_ON	X1	%IX1	BOOL
1	VAR_GLOBAL	PUMP_OFF_NC	X2	%IX2	BOOL
2	VAR_GLOBAL	Level_sensor	X3	%IX3	BOOL
3	VAR_GLOBAL	Pump	Y10	%QX16	BOOL

For further information about the Global Variable List please refer to chapter 4.6.2.

4.7.9 Generating pulses

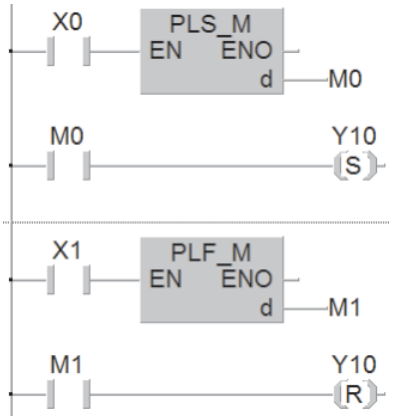
Instruction	Function	Ladder Diagram	IEC Instruction List
PLS	Pulse, sets an device* for the duration of a single program cycle on the rising edge of the switching pulse of the input condition / device	PLS_M EN ENO d	PLS_M
PLF	Pulse Falling, sets a device* for the duration of a single program cycle on the falling edge of the switching pulse of the input condition / device	PLF_M EN ENO d	PLF_M

* PLC and PLF instructions can be used to set outputs (Y) and relays (M).

These instructions effectively convert a static signal into a brief pulse, the duration of which depends on the length of the program cycle. If you use PLS instead of an OUT instruction the signal state of the specified device will only be set to “1” for a single program cycle, specifically during the cycle in which the signal state of the device before the PLS instruction in the circuit switches from “0” to “1” (rising edge pulse).

The PLF instruction responds to a falling edge pulse and sets the specified device to “1” for a single program cycle, during the cycle in which the signal state of the device before the PLF instruction in the circuit switches from “1” to “0” (falling edge pulse).

Ladder Diagram

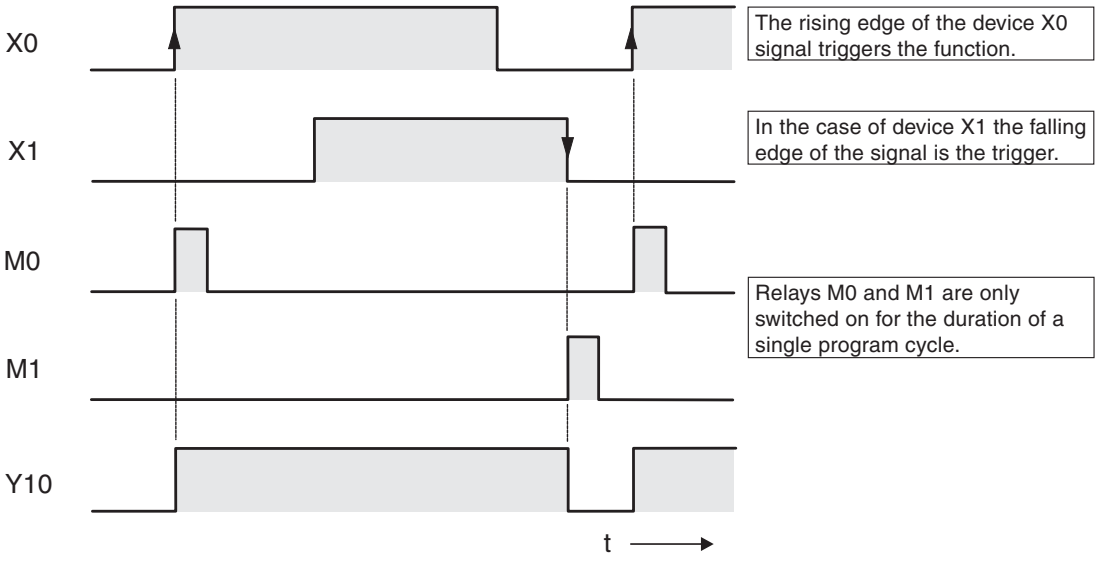


MELSEC Instruction List

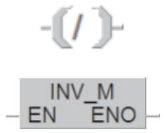
LD	X0
PLS	M0
LD	M0
SET	Y10
LD	X1
PLF	M1
LD	M1
RST	Y10

IEC Instruction List

LD	X0
PLS_M	M0
LD	M0
S	Y10
LD	X1
PLF_M	M1
LD	M1
R	Y10



4.7.10 Inverting the result of an operation

Instruction	Function	Ladder Diagram	IEC Instruction List
INV	Invert, reverses the result of an operation		NOT

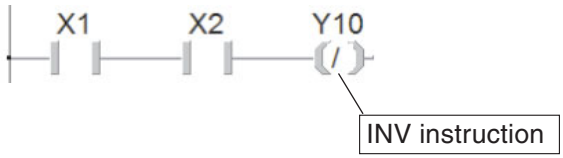
The INV instruction is used on its own without any operands. It inverts the result of the operation that comes directly before it:

– If the operation result was “1” it is inverted to “0”

– If the operation result was “0” it is inverted to “1”.

Ladder Diagram

1st alternative



2nd alternative

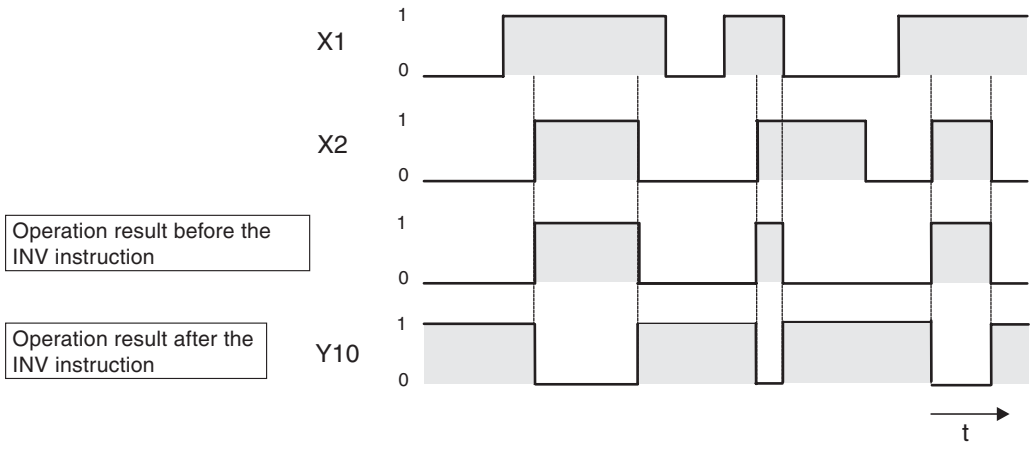
MELSEC Instruction List

```
LD    X1
AND   X2
INV
OUT   Y10
```

IEC Instruction List

```
LD    X1
AND   X2
NOT
ST    Y10
```

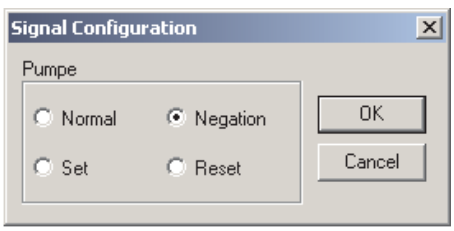
The above example produces the following signal sequence:



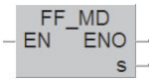
The INV instruction can be used when you need to invert the result of a complex operation. It can be used in the same position as the AND and ANI instructions.

NOTE

To program an INV instruction in Ladder Diagram within an OUT instruction, please double-click on the OUT instruction to display the ***Signal Configuration*** window.Select ***Negation*** and confirm with ***OK*** (see also chapter 4.7.8)



4.7.11 Inversion of bit output device

Instruction	Function	Ladder Diagram	IEC Instruction List
FF	Inversion of bit output device		FF_MD

* The FF instruction can be used to set outputs (Y), relays (M) and individual bits of word devices.

The FF instruction inverts the operation condition of the device designated at the output with the rising edge at the input of the FF instruction.

– If the condition of the output device is set (1) it will be reset (0) after inversion.

– If the condition of the output device is reset (0), it will be set (1) after inversion.

Ladder Diagram

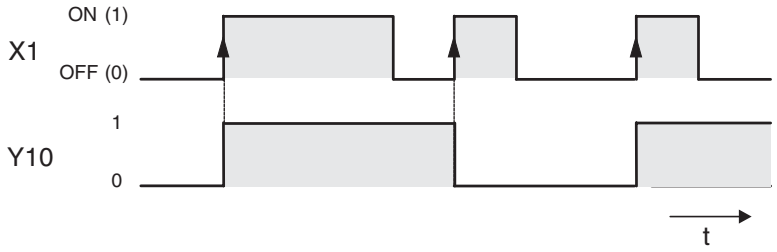
MELSEC Instruction List

```
LD      X1
FF      Y10
```

IEC Instruction List

```
LD      X1
FF_MD   Y10
```

The above program inverts the output condition of Y10 with the rising edge from input X1:

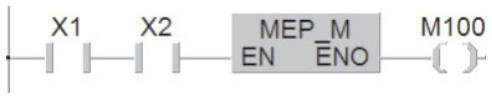


4.7.12 Operation result into pulse conversion

Instruction	Function	Ladder Diagram	IEC Instruction List
MEP	Pulse generation at the rising edge of operation result	MEP_M EN ENO	MEP_M
MEF	Pulse generation at the falling edge of operation result	MEF_M EN ENO	MEF_M

The MEP and MEF instructions are used without devices. They generate one output pulse with the rising respectively falling edge of the input signal i.e. operation result which was valid before the execution of these instructions. The next pulse is generated with the next rising resp. falling edge.

Ladder Diagram



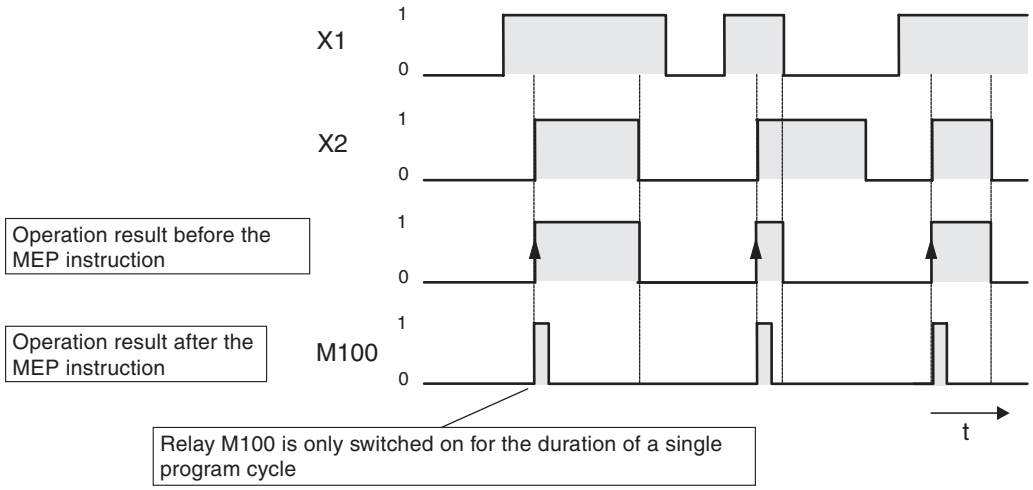
MELSEC Instruction List

```
LD      X1
AND     X2
MEP
OUT     M100
```

IEC Instruction List

```
LD      X1
AND     X2
MEP_M
ST      M100
```

The following figure shows the signal sequence for the above example:



These two instructions are especially suitable for multiple contacts connections. For example, multiple normally open contacts connected in series would maintain the operation result 1 if they were all closed. If a relay was set by this operation result, it could not be reset. With a MEP instruction connected in series with these NO contacts the relay could be reset because the instruction outputs one pulse only, if the series connection result of all contacts changes from 0 to 1.

4.8 Safety First!

PLCs have many advantages over hard-wired controllers. However, when it comes to safety it is important to understand that you cannot trust a PLC blindly.

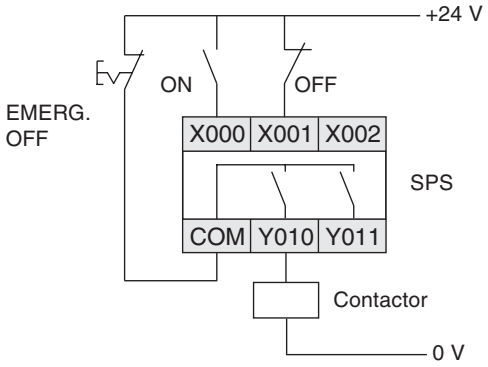
Emergency STOP devices

It is essential to ensure that errors in the control system or program cannot cause hazards for staff or machines. Emergency STOP devices must remain fully functional even when the PLC is not working properly – for example to switch off the power to the PLC outputs if necessary.

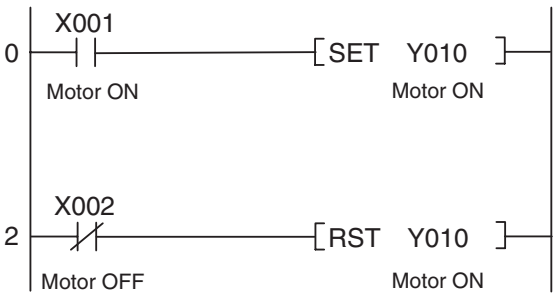
Never implement an Emergency STOP switch **solely** as an input that is processed by the PLC, with the PLC program activating the shutdown. This would be much too risky.

Safety precautions for cable breaks

You must also take steps to ensure safety in the event that the transmission of signals from the switches to the PLC are interrupted by cable breaks. When switching equipment on and off via the PLC always use switches or pushbuttons with make contacts for switching on and with break contacts for switching off.



In this example the contactor for a drive system can also be switched off manually with an Emergency OFF switch.



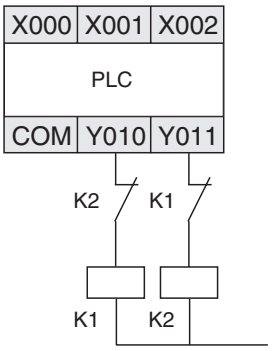
In the program for this installation the make contact of the ON switch is polled with an LD instruction, the break contact of the OFF switch with an LDI instruction. The output, and thus also the drive, is switched off when the input X002 has a signal state of “0”. This is the case when the OFF switch is operated or when the connection between the switch and input X002 is interrupted.

This ensures that if there is a cable break the drive is switched off automatically and it is not possible to activate the drive. In addition to this, switching off has priority because it is processed by the program after the switch on instruction.

Interlock contacts

If you have two outputs that should never both be switched on at the same time – for example outputs for selecting forward or reverse operation for a motor – the interlock for the outputs must also be implemented with physical contacts in the contactors controlled by the PLC. This is necessary because only an internal interlock is possible in the program and an error in the PLC could cause both outputs to be activated at the same time.

The example on the right shows such an interlock with contactor contacts. Here it is physically impossible for contactors K1 and K2 to be switched on at the same time.



Automatic shutdown

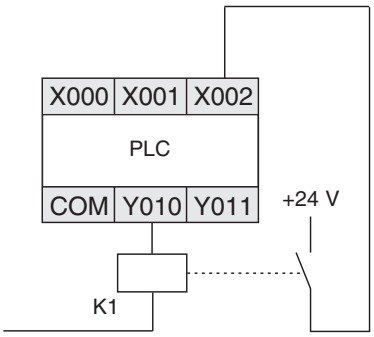
When a PLC is used to control motion sequences in which hazards can arise when components move past certain points additional limit switches must be installed to interrupt the movement automatically. These switches must function directly and independently of the PLC. See Chapter 4.9.1 for an example of such an automatic shutdown facility.

Output signal feedback

Generally, the outputs of PLCs are not monitored. When an output is activated the program assumes that the correct response has taken place outside the PLC. In most cases no additional facilities are required. However, in critical applications you should also monitor the output signals with the PLC – for example when errors in the output circuit (wire breaks, seized contacts) could have serious consequences for safety or system functioning.

In the example on the right a make contact in contactor K1 switches input X002 on when output Y10 is switched on. This allows the program to monitor whether the output and the connected contactor are functioning properly.

Note that this simple solution does not check whether the switched equipment is functioning properly (for example if a motor is really turning). Additional functions would be necessary to check this, for example a speed sensor or a voltage load monitor.



4.9 Programming PLC Applications

Programmable logic controllers provide an almost unlimited number of ways to link inputs with outputs. Your task is to choose the right instructions from the many supported by the controllers of the MELSEC System Q to program a suitable solution for your application.

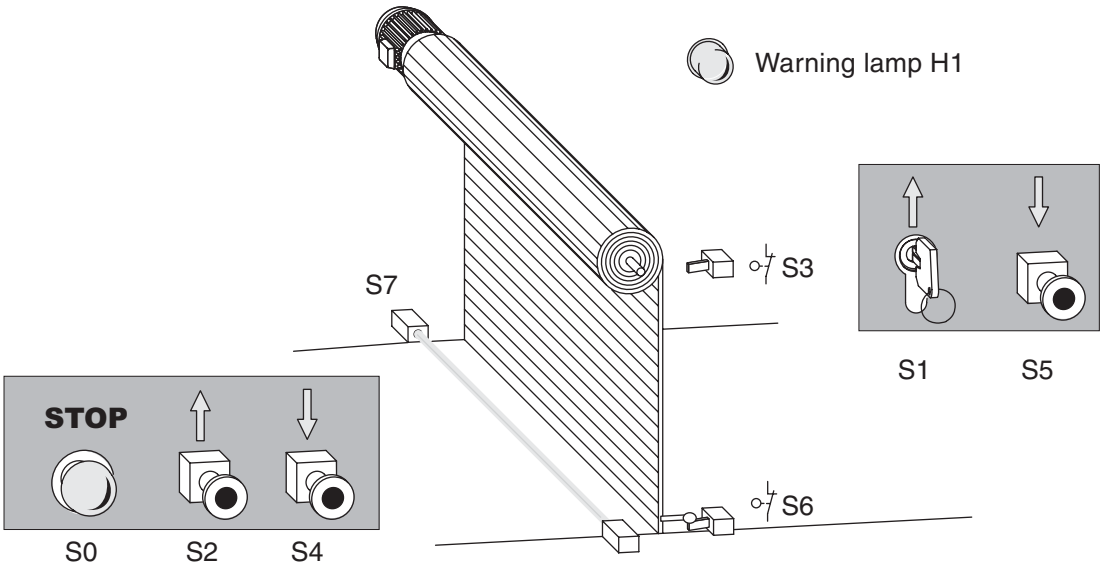
This chapter provides a simple example that demonstrate the development of a PLC application from the definition of the task to the finished program.

4.9.1 A rolling shutter gate

The first step is to have a clear concept of what you want to do. This means that you need to take a "bottom-up" approach and write a clear description of what it is you want the PLC to do.

Task description

We want to implement a control system for a warehouse's rolling shutter gate that will enable easy operation from both outside and inside. Safety facilities must also be integrated in the system.



- Operation
  - It must be possible to open the gate from outside with the key-operated switch S1 and to close it with pushbutton S5. Inside the hall it should be possible to open the gate with pushbutton S2 and to close it with S4.
  - An additional time switch must close the gate automatically if it is open for longer than 20 s.
  - The states "gate in motion" and "gate in undefined position" must be indicated by a blinking warning lamp.

- Safety facilities

– A stop button (S0) must be installed that can halt the motion of the gate immediately at any time, stopping the gate in its current position. This Stop switch is not an Emergency OFF function, however! The switch signal is only processed by the PLC and does not switch any external power connections.

– A photoelectric barrier (S7) must be installed to identify obstacles in the gateway. If it registers an obstacle while the gate is closing the gate must open automatically.

– Two limit switches must be installed to stop the gate motor when the gate reaches the fully open (S3) and fully closed (S6) positions.

Assignment of the input and output signals

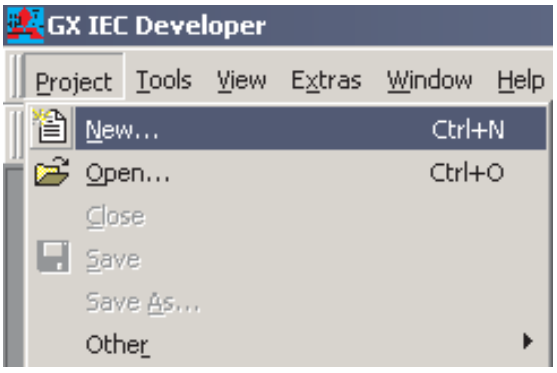
The task description clearly defines the number of inputs and outputs needed. The gate drive motor is controlled with two outputs. The signals required are assigned to the PLC inputs and outputs as follows:

Function		Name	Address	Remarks
Inputs	STOP button	S0	X0	Break contact (when the switch is operated X0 = “0” and the gate stops)
	OPEN key-operated switch (outside)	S1	X1	Make contacts
	OPEN button (inside)	S2	X2	
	Upper limit switch (gate open)	S3	X3	Break contact (X2 =”0” when the gate is up and S3 is activated)
	CLOSE button (inside)	S4	X4	Make contacts
	CLOSE button (outside)	S5	X5	
	Lower limit switch (gate closed)	S6	X6	Break contact (X6 = “0” when the gate is down and S6 is activated)
	Photoelectric barrier	S7	X7	X7 is set to “1” when an obstacle is registered
Outputs	Warning lamp	H1	Y10	—
	Motor contactor (motor reverse)	K1	Y11	Reverse = OPEN gate
	Motor contactor (motor forward)	K2	Y12	Forward = CLOSE gate
Timer	Delay for automatic close	—	T0	Time: 20 seconds

4.9.2 Programming

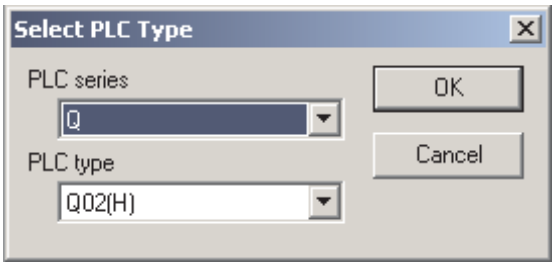
Creating a new project

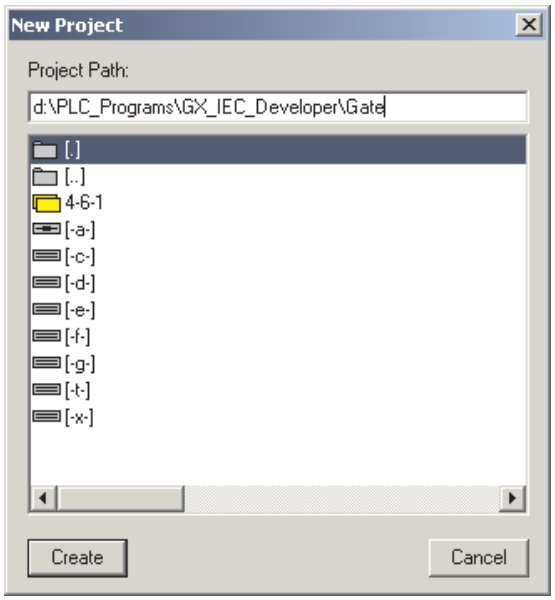
After the start of GX IEC Developer select ***New*** in the ***Project*** menu.



Choose the appropriate PLC type from the selection.

Click on ***OK*** to confirm your selection

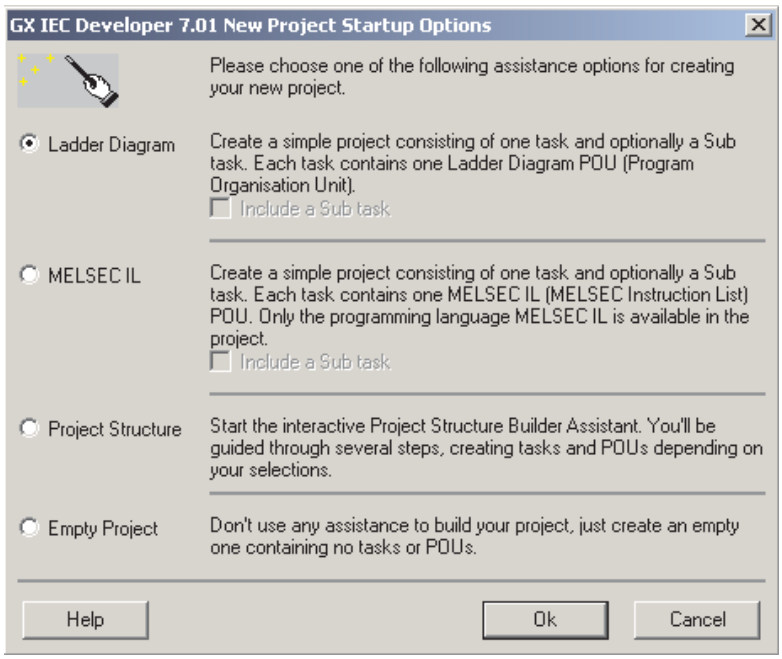




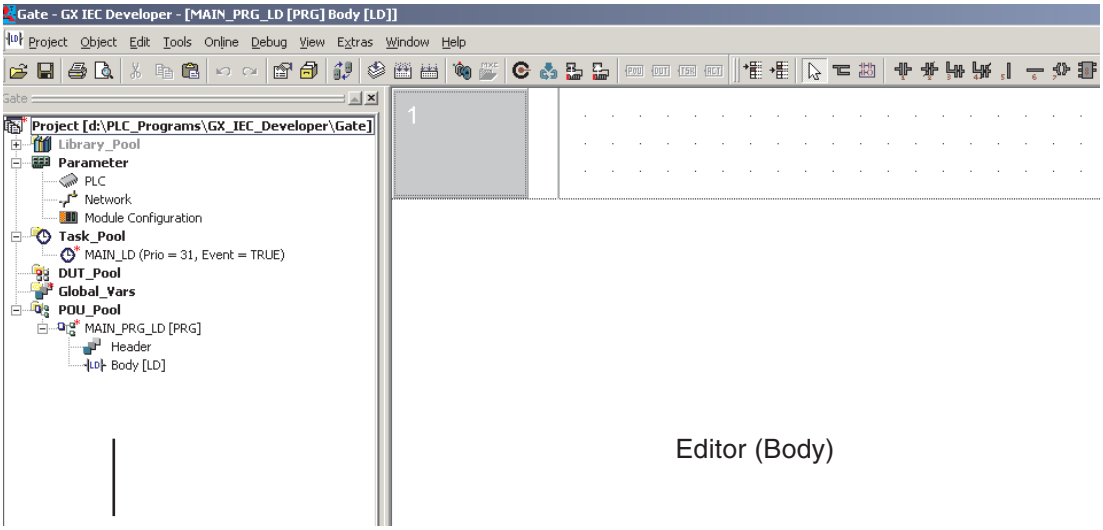
Now the ***New Project*** dialog box is displayed. Select or enter the path under which you wish to save the new project. Enter also a name for the new project at the end of the path.

After clicking on the ***Create*** button a subdirectory with the specified name of the new project will be created by GX IEC Developer.

Select ***Startup Options***. For this example ***Ladder Diagram*** is chosen.



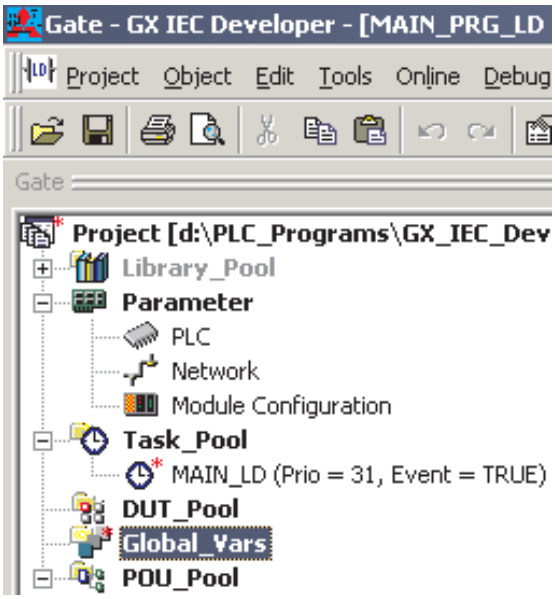
After confirming with ***OK*** the programming can begin. The project display screen with the empty body of the POU MAIN is shown as illustrated on the next page.



Assigning the Global Variables

NOTE

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. However the program will no longer be truly IEC61131-3 compliant.



Double-click ***Global_Vars*** in the Project Navigator Window.

The Global Variable List (GVL) is opened.

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

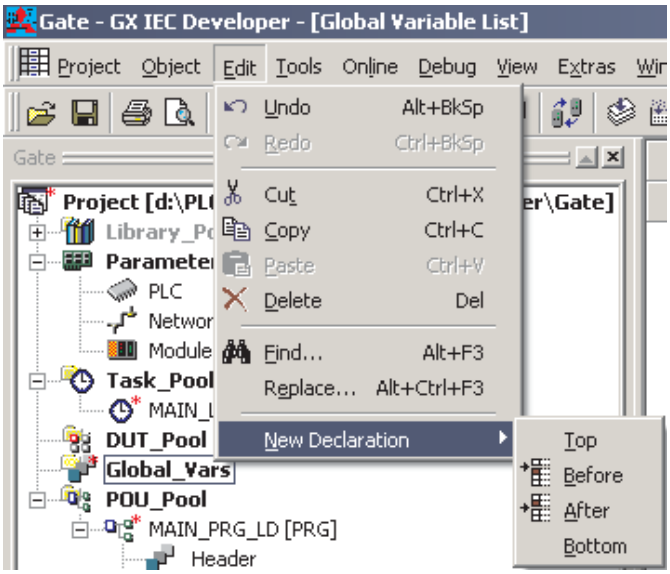
Enter the identifier and the absolute address of the first Global Variable.You do not need to enter both the MITSUBISHI address and the IEC address. If one address is entered, the other one is automatically added by GX IEC Developer.

	Class	Identifier	MIT-Addr.	IEC-Addr.	Type		Initial	Comment
0	VAR_GLOBAL	S0_STOP_Gate	X0	%IX0	BOOL	...	FALSE	Normally Closed

If a physical input address is entered BOOL is selected as type automatically.

To declare further variables the list has to be expanded. There are several ways to do this:

- If the cursor is active in any column of the last variable declaration row press the SHIFT and the ENTER key simultaneously.
- You can also select ***New Declaration*** in the ***Edit*** menu.



- or just click on the button "Insert before" or "Insert after" in the tool bar. 

These are the inputs and outputs specified as Global Variables for the project:

	Class	Identifier	MIT-Addr.	IEC-Addr.	Type		Initial	Comment
0	VAR_GLOBAL	S0_STOP_GATE	X0	%IX0	BOOL	...	FALSE	Normally closed
1	VAR_GLOBAL	S1_OPEN_GATE_Switch	X1	%IX1	BOOL	...	FALSE	
2	VAR_GLOBAL	S2_OPEN_GATE_PB	X2	%IX2	BOOL	...	FALSE	
3	VAR_GLOBAL	S3_Upper_limit_switch	X3	%IX3	BOOL	...	FALSE	Normally closed
4	VAR_GLOBAL	S4_CLOSE_GATE_PB	X4	%IX4	BOOL	...	FALSE	Inside
5	VAR_GLOBAL	S5_CLOSE_GATE_PB	X5	%IX5	BOOL	...	FALSE	Outside
6	VAR_GLOBAL	S6_Lower_limit_switch	X6	%IX6	BOOL	...	FALSE	Normally closed
7	VAR_GLOBAL	S7_Photoelectric_barrier	X7	%IX7	BOOL	...	FALSE	
8	VAR_GLOBAL	H1_Warning_lamp	Y10	%QX16	BOOL	...	FALSE	
9	VAR_GLOBAL	K1_Motor_open_gate	Y11	%QX17	BOOL	...	FALSE	
10	VAR_GLOBAL	K2_Motor_close_gate	Y12	%QX18	BOOL	...	FALSE	

Entering of the Program

Next you can program the individual control tasks:

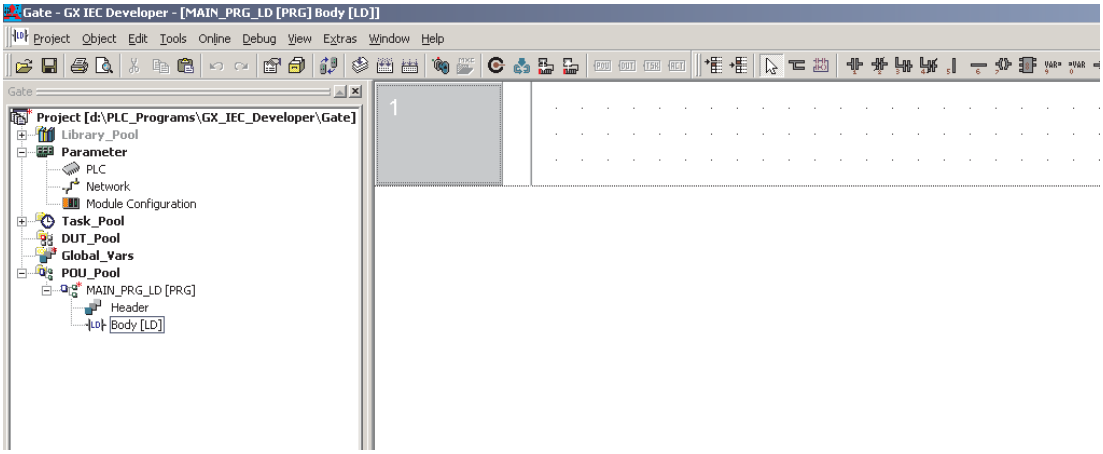
- Operation of the rolling shutter gate with the pushbuttons

The program must convert the input signals for the operation of the gate into two commands for the drive motor: “Open Gate” and “Close Gate”. Since these are signals from pushbuttons that are only available briefly at the inputs they need to be stored. To do this we use two variables to represent the inputs in the program and set and reset them as required:

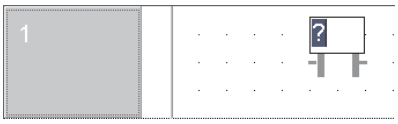
– OPEN_GATE

– CLOSE_GATE

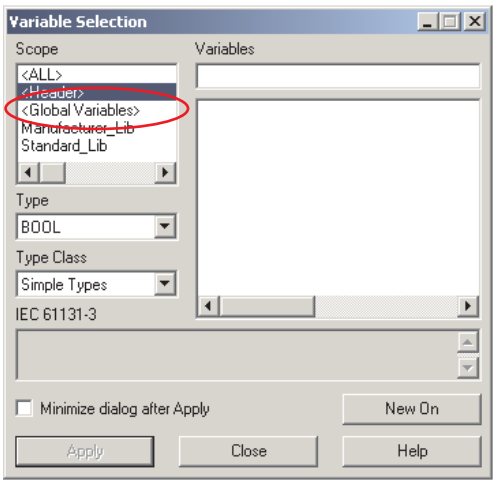
When the body of the POU MAIN is not already displayed double-click on ***Body [LD]*** in the Project Navigator.



Select the ‘Normally Open’ contact from the toolbar.

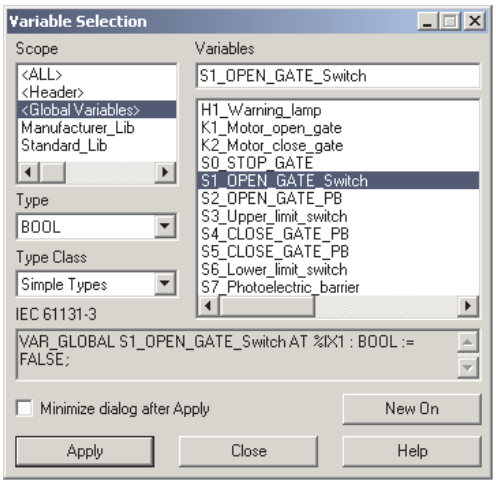


Move the mouse pointer over the work area and click to fix the drop position on the window.



Right click on the question mark to call up the ***Variables Selection*** window.

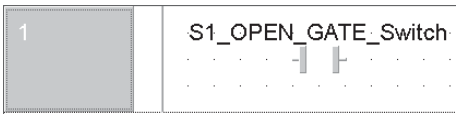
Click on ***Global Variables*** in the ***Scope*** dialogue area.



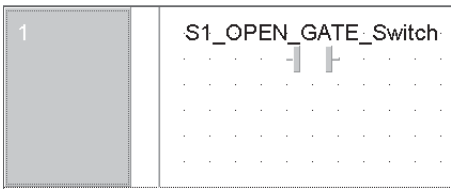
Click the desired variable (in this case “S1_OPEN_GATE_Switch”) to highlight that variable

The thus selected variable is entered by a click on the ***Apply*** button or by double-clicking on the variable itself.

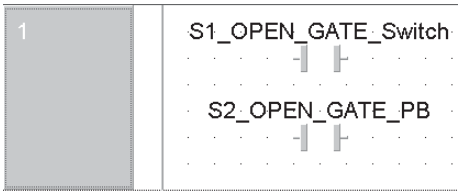
The variable is entered.



Click on the editor to display the full identifier of the variable.

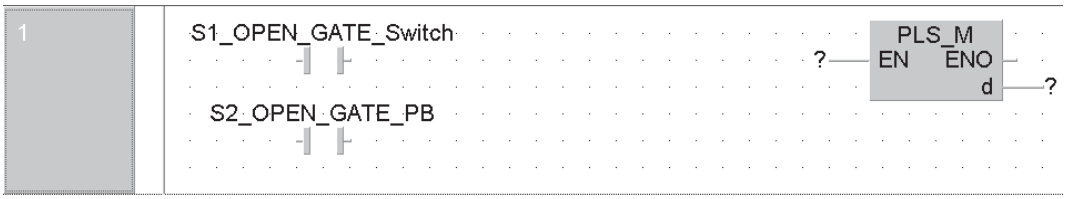


The ladder network may be re-sized by moving the mouse pointer to the lower boundary of the network header and ‘click-hold’ dragging downward to increase the vertical size.

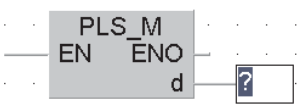


Enter also the push-button for opening the gate.

Any actuation of these switches has to be converted into a pulse. For this the function PLS_M is used. How a function is enterd into the ladder program is covered in chapter 4.7.7.



Click on the  button (output variable) in the tool bar.



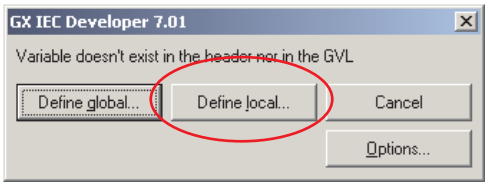
Then click on the output of the PLS_M function to display the variable prompt field.

– Assigning of local variables

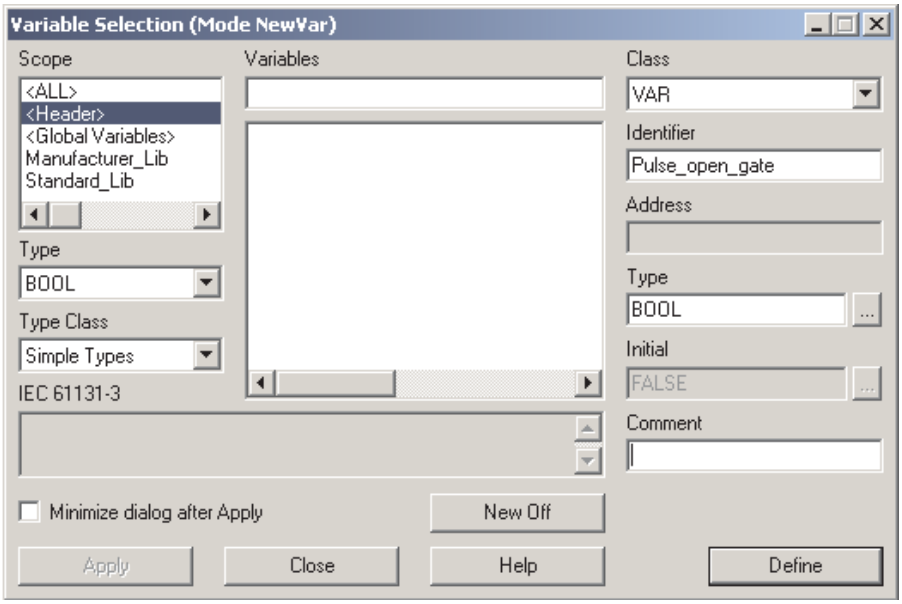
The output variable of the PLS_M function is only used in this POU. Therefore it can be a local variable. No local variables were assigned for this project yet because this can be done during programming.

Enter the variable name Pulse_open_gate into the empty '?' box.

The following prompt is displayed because the variable does not exist in the Local Variable List or the Global Variable List:



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

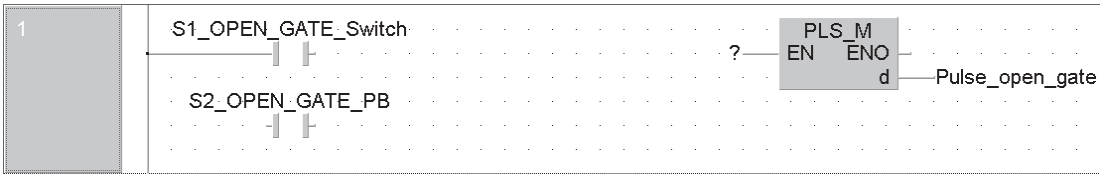


Click ***Define*** to enter the new variable into the Local Variable List (Local Header of the POU).

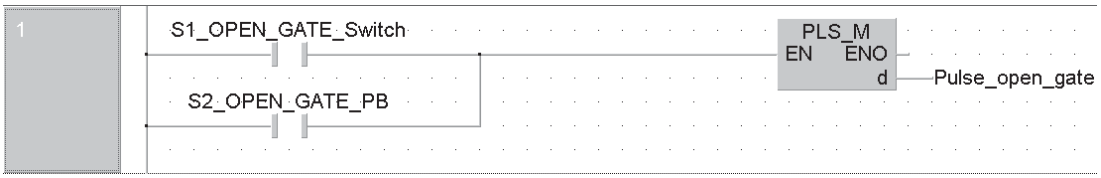
Finally, the ladder network must be finalised by connecting up the elements.

For this purpose the tool bar contains the icon "Line mode". Note that the pointer now changes to a small pencil icon.

Click on the bus bar at the left on the ladder diagram and "Click – Drag" across the diagram and release on the contact. Release the left mouse button at this point.



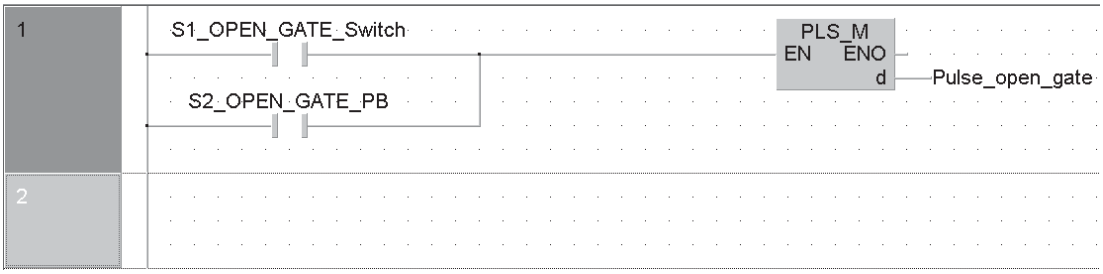
Connect all other elements in this network in the same way.



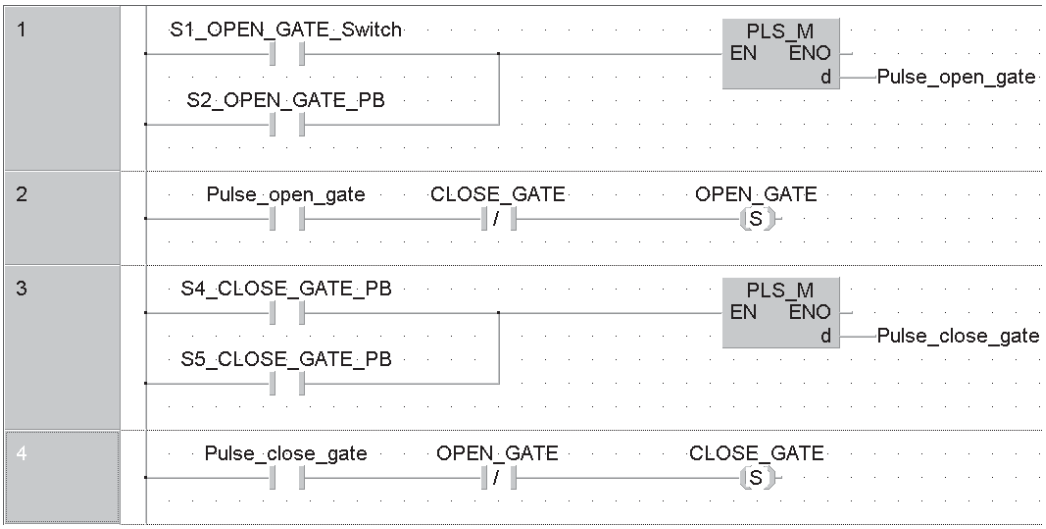
– Creating a new program network

To create a network below the current one, click in the toolbar on this button:

A blank network space will appear:



Enter the following elements in this network and further networks.



All variables except the switches and push buttons are local variables. This already shows the advantage of using variables with identifiers: Even without entering device comments this program is easier to understand than a program with absolute addresses (e.g. X1, X2 etc.).

● Description of networks 1 to 4

The signals for opening the gate are processed first: When key-operated switch S1 or button S2 are operated a signal is generated and the variable "Pulse_open_gate" is set to a signal state of “1” for just one program cycle. This ensures that the gate cannot be blocked if the button sticks or of the operator does not release it. A similar approach is used to process the signals from buttons S4 and S5 for closing the gate. It must be ensured that the drive can only be switched on when it is not already turning in the opposite direction. This is implemented by programming the PLC so that the gate can only be closed when it is not opening and vice versa.

NOTE

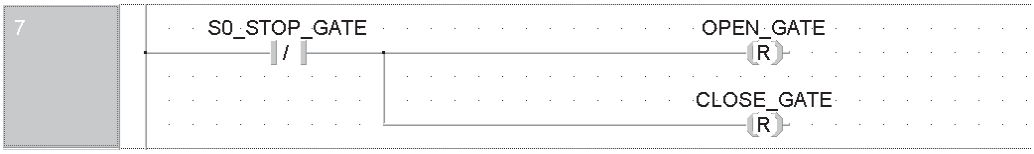
The motor direction interlock must also be complemented by an additional interlock with physical contactors outside the PLC (see wiring diagram in chapter 4.9.3.).

- Close gate automatically after 20 seconds

When the gate is open limit switch S3 activates and input X3 is switched off. (For safety reasons S3 is a break contact.) When this happens timer T0 starts the 20s delay (K200 = 200 x 0.1s = 20s). When the timer reaches 20s the gate is closed.

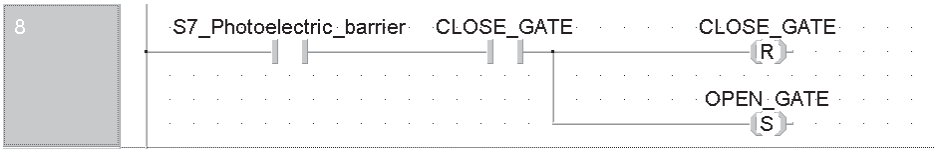
NOTE Timer are described in detail in the next chapter.

- Stop gate with STOP switch



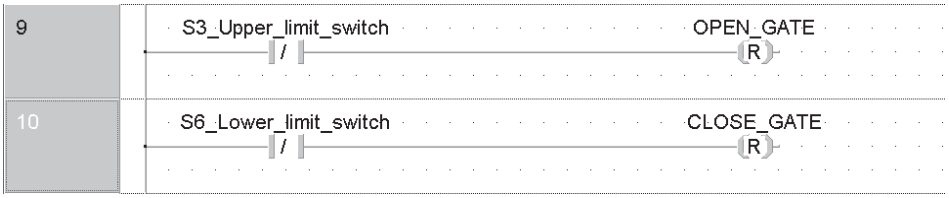
Pressing the STOP button resets the two local variables, stopping the gate motor.

- Identifying obstacles with the photoelectric barrier



If an obstacle is registered by the photoelectric barrier while the gate is closing the close operation is halted and the gate is opened again.

- Switching the motor off with the limit switches



When the gate is open limit switch S3 is activated and input X3 is switched off. This resets the local variable OPEN_GATE and stops the motor.

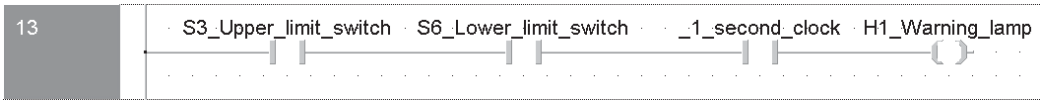
When the gate is fully closed S6 is activated, turning off the motor also. For safety reasons the limit switches are break contacts. This ensures that the motor is also switched off automatically (or cannot be switched on) if the connection between the switch and the input is interrupted.

NOTE The limit switches must be wired so that they also switch off the motor automatically without support from the PLC (see wiring diagram in chapter 4.9.3).

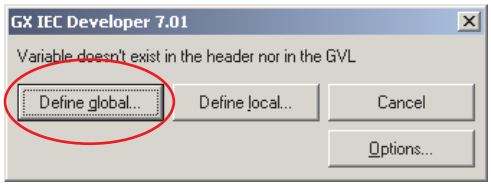
- Controlling the motor

At the end of the program the signal states of the local variables for opening and closing are transferred to outputs Y11 and Y12.

- Warning lamp: “Gate in Motion” and “Gate in Undefined Position”

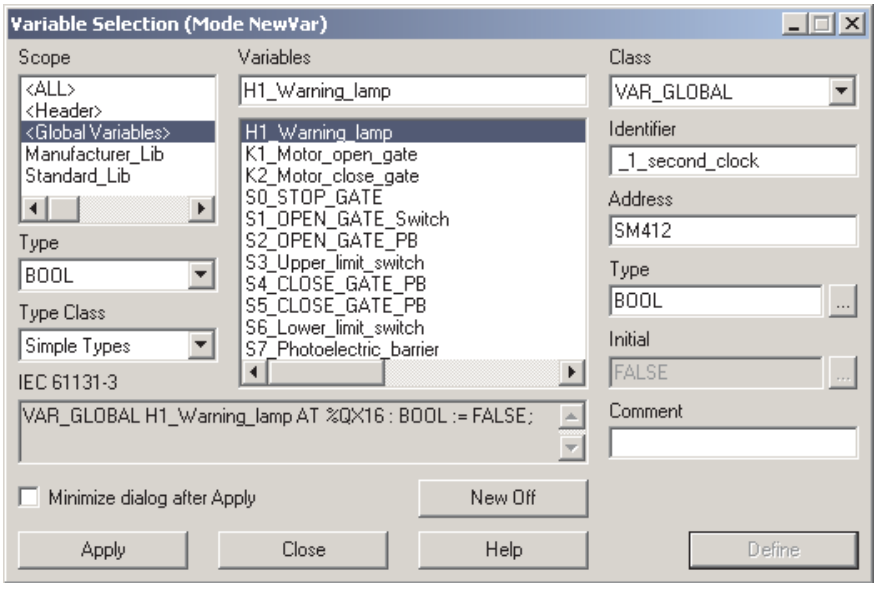


If neither of the limit switches is activated this means that the gate is being opened or closed or has been stopped in an intermediate position. In all these situations the warning lamp blinks. The blink speed is controlled with special relay SM412, which is automatically set and reset at 1s intervals (see chapter 5.2). SM412 is assigned as Global Variable when entering the program:

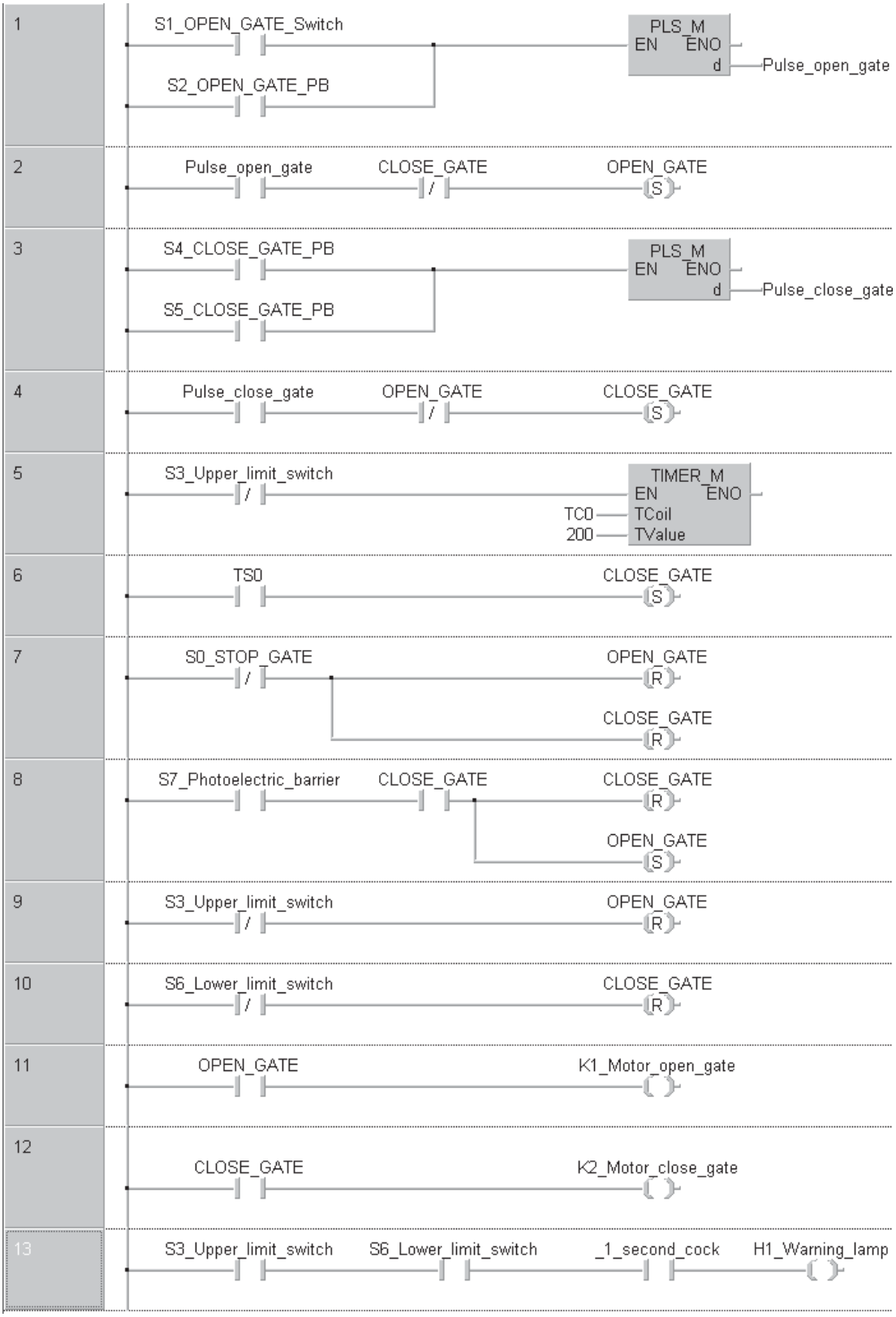


After entering the variable name _1_second_clock the message shown on the left appears because this variable doesn't exist yet. Click on ***Define global***.

Enter SM412 into the ***Address*** field of the ***Variable Selection*** window. Then click on ***Define***.



The illustration on the next page shows the entire Ladder Diagram for the control of the rolling shutter gate.



NOTE

Very important is the order of the instructions, especially the reset of the variables OPEN_GATE and CLOSE_GATE by the safety facilities at the end of the program **after** the setting of these variables.
Because of the execution of the program from top to bottom (see chapter 2.2), reset has a higher priority than set and therefore safety is ensured.

4.9.3 The Hardware

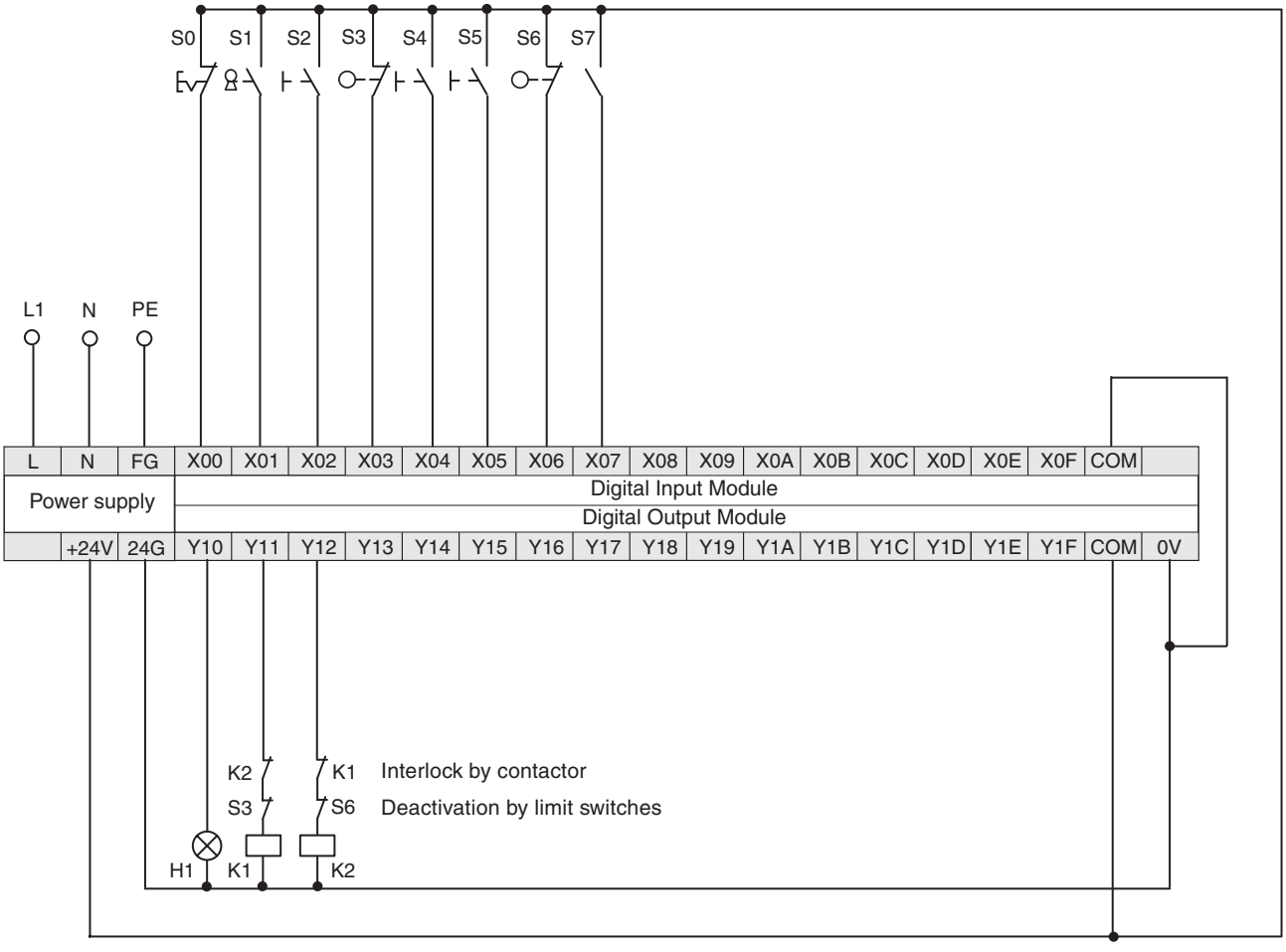
The rolling shutter gate can be implemented with the following components of the MELSEC System Q:

- Main base unit with at least two slots for I/O modules, e. g. Q33B
- Power Supply Q62P

  This power supply module provides 24 DC for the supply of sensors and indicator lamps. Please note that this output can be loaded with a maximum current of 0.6 A.
- CPU module (as required)*
- 1 Digital input module QX80 with 16 inputs (negative common)
- 1 Digital output module QY80 with 16 transistor outputs (source)

* Of course it's a little bit exaggerated to use a PLC of the MELSEC System O solely for the control of a roller shutter gate. The CPU would be unchallenged with this task. But as part of a complex application, e. g. for the control of a production line, this application is conceivable.

Connection of the PLC



A reference for the names of the electrical equipment and the function is shown on the next page.

Name	Function	Adress	Remarks
S0	STOP button	X0	Break contact (Normally closed, NC)
S1	OPEN key-operated switch (outside)	X1	Make contacts (Normally open, NO)
S2	OPEN button (inside)	X2	
S3	Upper limit switch (gate open)	X3	Break contact (NC)
S4	CLOSE button (inside)	X4	Make contacts (NO)
S5	CLOSE button (outside)	X5	
S6	Lower limit switch (gate closed)	X6	Break contact (NC)
S7	Photoelectric barrier	X7	X7 is set to “1” when an obstacle is registered
H1	Warning lamp	Y10	—
K1	Motor contactor (motor reverse)	Y11	Reverse = OPEN gate
K2	Motor contactor (motor forward)	Y12	Forward = CLOSE gate

5 Devices in Detail

The devices in PLCs are used directly in control program instructions. Their signal states can be both read and changed by the PLC program. A device reference has two parts:

– the device name and

– the device address.

Example of a device reference (e.g. input 0):

5.1 Inputs and Outputs

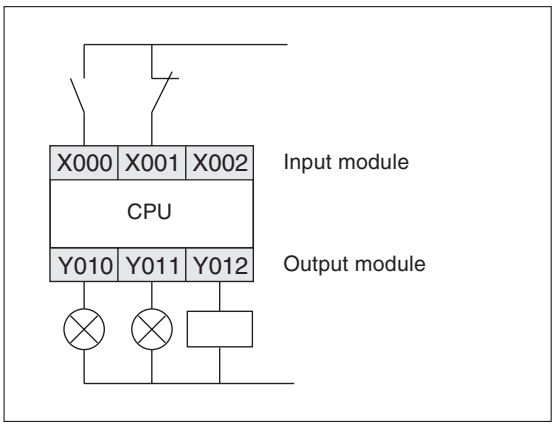
The PLC's inputs and outputs connect it to the process that it is controlling. When an input is polled by the PLC program the voltage on the input terminal of an input module is measured. Since these inputs are digital they can only have two signal states, ON or OFF. When the voltage at the input terminal reaches the rated voltage (e. g. 24V) the input is on (state "1"). If the voltage is lower the input evaluates as off (signal state "0").

In MELSEC PLCs the identifier "**X**" is used for inputs. The same input can be polled as often as necessary in the same program.

NOTE

The PLC cannot change the state of inputs. For example, it is not possible to execute an OUT instruction on an input device.

If an output instruction is executed on an output the result of the current operation (the signal state) is applied to the output terminal of an output module. If it is a relay output the relay closes (all relays have make contacts). If it is a transistor output the transistor makes the connection and activates the connected circuit.



The illustration on the left shows an example of how you can connect switches to the inputs and lamps and contactors to the outputs of a MELSEC PLC.

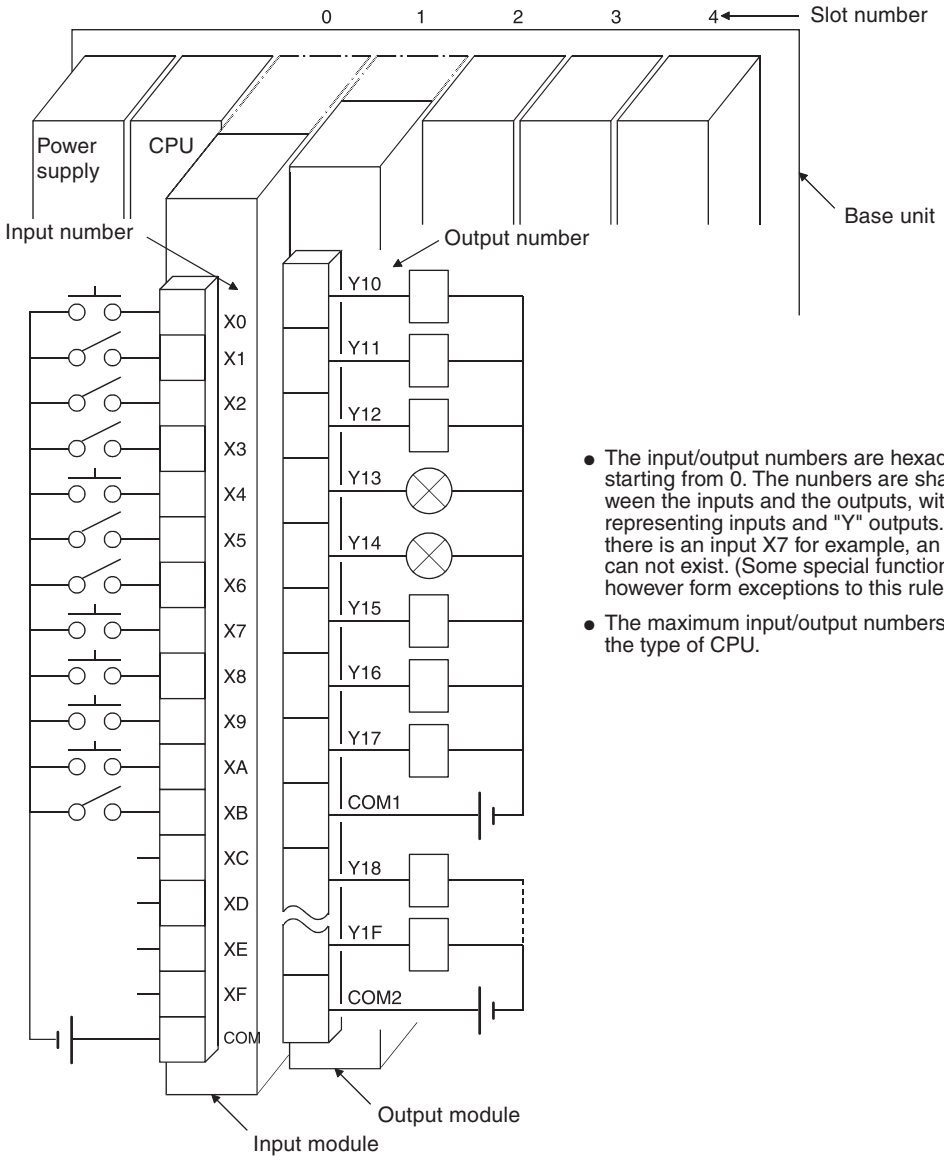
The identifier for output devices is "**Y**". Outputs can be used in logic operation instructions as well as with output instructions. However, it is important to remember that you can never use an output instruction on the same output more than once (see also section 4.7.2).

5.1.1 External I/O Signals and I/O Numbers

Signals from external input devices are replaced by input numbers, which are determined by the mounting position (see chapter 3.2.2) 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 also determined by the mounting position and terminal numbers of the output module with which external output devices are connected.

The input and outputs are numbered hexadecimal (0, 1, 2 ...9, A, B, C, D, E, F; 10, 11, 12 ...). Thus I/O signals are grouped into groups with 16 input or outputs



- The input/output numbers are hexadecimal starting from 0. The nunbers are shared between the inputs and the outputs, with "X" representing inputs and "Y" outputs. When there is an input X7 for example, an output Y7 can not exist. (Some special function modules however form exceptions to this rule.)
- The maximum input/output numbers vary with the type of CPU.

5.1.2 Inputs and Outputs of the MELSEC System Q

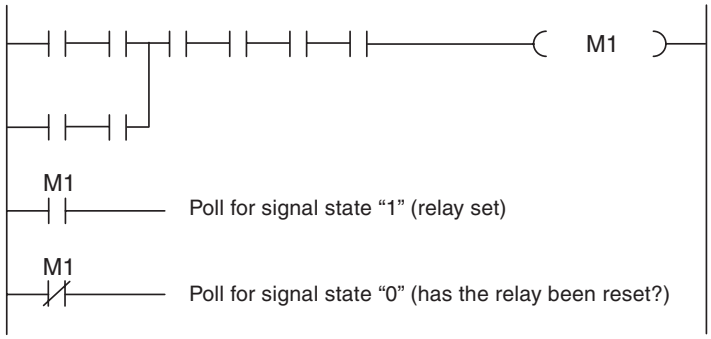
The following table provides a general overview of the inputs and outputs of the controllers of the MELSEC System Q.

Device		Inputs and Outputs	
		I/O in Main and Extension Base Units	I/O in Main and Extension Base Units and remote I/O
Device identifier		X (Inputs), Y (Outputs)	
Device type		Bit device	
Possible values		0 or 1	
Device address format		Hexadecimal	
Number of devices and addresses (depends on type of CPU)	Q00J	256 (X/Y000 to X/Y00FF)	2048 (X/Y000 to X/Y07FF)
	Q00	1024 (X/Y000 to X/Y03FF)	2048 (X/Y000 to X/Y07FF)
	Q01		
	Q02	4096 (X/Y000 to X/Y0FFF)	8192 (X/Y000 to X/Y1FFF)
	Q02H		
	Q06H		
	Q12H		
	Q25H		
	Q12PH		
	Q25PH		

5.2 Relays

In your PLC programs you will often need to store intermediate binary results (a signal state of “0” or “1”) temporarily for future reference. The PLC has special memory cells available for this purpose known as “auxiliary relays”, or “relays” for short (device identifier: "**M**").

You can store the binary result of an operation in a relay, for example with an OUT instruction, and then use the result in future operations. Relays help to make programs easier to read and also reduce the number of program steps: You can store the results of operations that need to be used more than once in a relay and then poll it is often as you like in the rest of the program.



In addition to normal relays the controllers of the MELSEC System Q also have retentive or “latched” relays. The normal unlatched relays are all reset to a signal state of “0” when the PLC power supply is switched off, and this is also their standard state when the controller is switched on. In contrast to this, latched relays retain their current states when the power is switched off and on again.

Device		**Relay types**	
		Unlatched relays	**Latched relays**
Device identifier		M	L
Device type		Bit device	
Possible values for a device		0 or 1	
Device address format		Decimal	
Number of devices and addresses	Q00J	8192 (M0–M8191)*	8192 (L0–L8191)*
	Q00		
	Q01		
	Q02		
	Q02H		
	Q06H		
	Q12H		
	Q25H		
	Q12PH		
	Q25PH		

* You can set the number of latched and unlatched relays with the PLC parameters. The values shown above are the initial settings.

5.2.1 Special relays

In addition to the relays that you can switch on and off with the PLC program there is also another class of relays known as special or diagnostic relays with the device identifier "SM". These relays contain information on system status or can be used to influence program execution. The following table shows a few examples of the many special relays available.

Special relay	Function	Program processing options
SM0	PLC error	Poll signal state
SM51	Low battery voltage	
SM400	When the PLC is in RUN mode this relay is always set to “1”.	
SM401	When the PLC is in Run mode this relay is always set to “0”.	
SM402	Initialisation pulse (following activation of RUN mode this relay is set to “1” for the duration of one program cycle.)	
SM411	Clock signal pulse: 0.2 second (0.1 s ON, 0.1s OFF)	
SM412	Clock signal pulse: 1 second (0.5 s ON, 0.5 s OFF)	
SM413	Clock signal pulse: 2 second (1 s ON, 1 s OFF)	
SM414	Variable clock signal pulse	

NOTE

For an overview of all special relays please refer to the Programming Manual for the A/Q series and the MELSEC System Q, art. no. 87431.

5.3 Timers

When you are controlling processes you will often want to program a specific delay before starting and stopping certain operations. In hard-wired controllers this is achieved with timer relays. In PLCs this is achieved with programmable internal timers.

Timers are really just counters that count the PLCs internal clock signals (e.g. 0.1s pulses). When the counter value reaches the setpoint value the timer's output is switched on.

A timer is represented by four elements:

– Set value (***TValue***)

– Actual value (***TN***)

– Timer coil (***TCoil***, ***TC***)

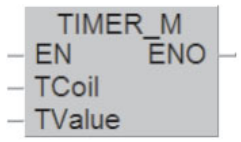
– Timer contact (***TS***)

All timers function as make delay switches and are activated with a "1" signal. To start and reset timers you program them in the same way as outputs. You can poll the outputs of timers (TS) as often as you like in your program.

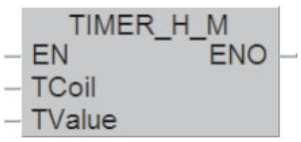
The timer of the MELSEC System Q are distinguished between low speed and high speed timer. With the GX IEC Developer the time base for the timer (i. e. the frequency of the clock signal counted by the timer) can be set in the range from 1 ms to 1000 ms in the PLC parameters. The initial value is 100 ms. The time base for high speed timer can be set in the range from 0.1 ms to 100 ms. The initial value in this case is 100 ms.

Whether a timer functions as low or as high speed timer is determined by the instruction which starts the timer.

Operation as low speed timer

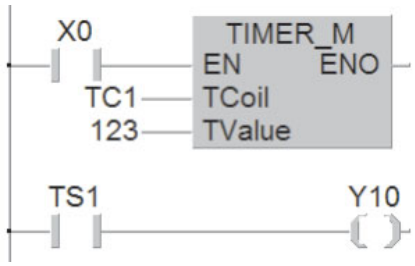


Operation as high speed timer



Example of a program using low speed timer

Ladder Diagram



As variable for the TCoil input of the TIMER_M instruction the device address of the timer is specified (in this example **TC1**).

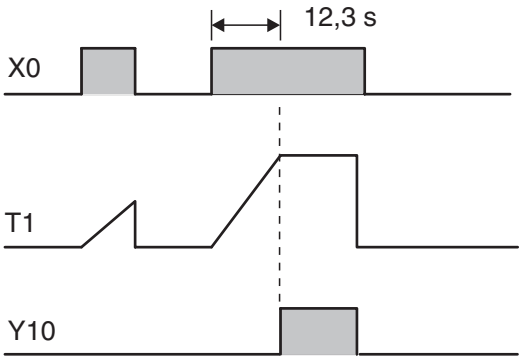
MELSEC Instruction List

```
LD      X0
OUT     T1
        K123
LD      T1
OUT     Y10
```

IEC Instruction List

```
LD          X0
TIMER_M     TC1,    123
LD          TS1
ST          Y10
```

In the above example timer T1 is started when input X0 is switched on. The setpoint value is 123 x 100 ms = 12,3 s, so T200 switches on output Y10 after a delay of 1.23 s. The signal sequence generated by the following program example is as follows:



The timer continues to count the internal 100ms pulses as long as X0 remains on. When the setpoint value is reached the output of T1 is switched on.

If input X0 or the power supply of the PLC are switched off the timer is reset and its output is also switched off.

You can also specify the timer setpoint value with a decimal value stored in a data register. See section 5.7.1 for details.

Retentive timers

In addition to the normal timers described above the controllers of the MELSEC System Q also have retentive timers that retain their current time counter value even if the device controlling them is switched off.

The current timer counter value is stored in a memory that is retained even in the event of a power failure.

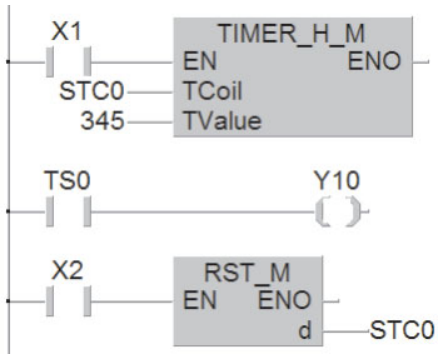
The device identifier for retentive timer is "ST". Similar to "normal" timers, retentive timers can be programmed as low speed or high speed timers.

NOTE

When shipped, 2048 (2k) normal timers are set in the parameters of an PLC CPU and no retentive timers. In order to use retentive timers, the number of these timers must be set in the PLC parameters.

Example of a program using a retentive timer as high speed timer:

Ladder Diagram



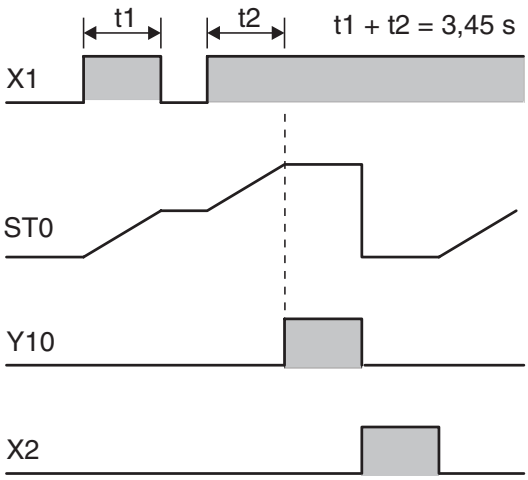
MELSEC Instruction List

LD	X1
OUTH	ST0
	K345
LD	ST0
OUT	Y10
LD	X2
RST	ST0

IEC Instruction List

LD	X1
TIMER_H_M	STC0, 345
LD	STS0
OUT	Y10
LD	X2
R	STC0

Timer T0 is started when input X1 is switched on. The setpoint value is 345 x 10ms = 3.45s. When the setpoint value is reached T0 switches output Y10 on. Input X2 resets the timer and switches its output off.



When X1 is on the timer counts the internal 10ms pulses. When X1 is switched off the current time counter value is retained. The timer's output is switched on when the current value reaches the setpoint value of the timer.

A separate instruction must be programmed to reset the timer since it is not reset by switching off input X1 or the PLC's power. Input X2 resets timer ST0 and switches off its output.

Timers of the MELSEC System Q PLC-CPUs

Device		Timer types	
		Normal Timer	**Retentive Timer**
Device identifier		T	ST
Device type (for setting and polling)		Bit device	
Possible values (timer output)		0 or 1	
Device address format		Decimal	
Timer setpoint value entry		As a decimal integer constant. The setpoint can be set either directly in the instruction or indirectly in a data register.	
Number of devices and addresses	Q00J	512 (T0 to T511)*	0*
	Q00		
	Q01		
	Q02	2048 (T0 to T2047)*	0*
	Q02H		
	Q06H		
	Q12H		
	Q25H		
	Q12PH		
	Q25PH		

* Initial values, the number of timers can be set in the PLC parameters.

5.4 Counters

The programmers of the MELSEC System Q also have internal counters that you can use for programming counting operations.

Counters count signal pulses that are applied to their inputs by the program. The counter output is switched on when the current counter value reaches the setpoint value defined by the program. Like timers, counter outputs can also be polled as often as you like in the program.

A counter is represented by four elements:

– Set value (***CValue***)

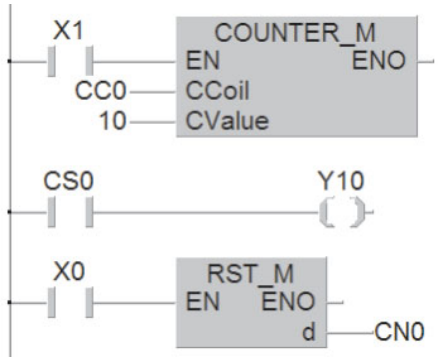
– Actual value(***CN***)

– Counter coil (***CCoil, CC***)

– Counter contact (***CS***)

Example of a program using a counter:

Ladder Diagram



As variable for the CCoil input of the COUNTER_M instruction the device address of the counter is specified (in this example **CC0**).

MELSEC Instruction List

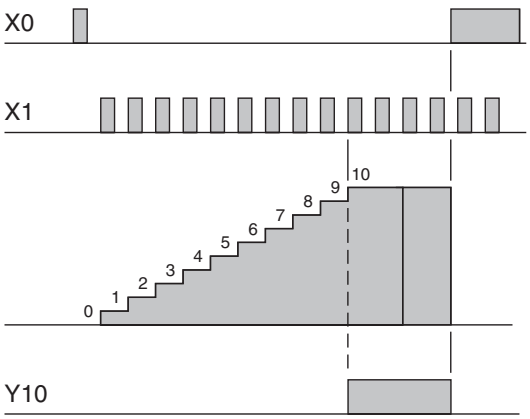
```
LD      X1
OUT     C0
        K10
LD      C0
OUT     Y10
LD      X0
RST     C0
```

IEC Instruction List

```
LD      X1
COUNTER_M   CC0, 10
LD      CS0
ST      Y10
LD      X0
R       CN0
```

Whenever input X1 is switched on the value of counter C0 is incremented by 1. Output Y10 is set when X1 has been switched on and off ten times (the counter setpoint is K10).

The signal sequence generated by this program is as follows:



First the counter is reset with input X0 and a RST instruction. This resets the counter value to 0 and switches off the counter output.

Once the counter value has reached the setpoint value any additional pulses on input X1 no longer have any effect on the counter.

The following table shows the key features of these counters.

Feature	Counter
Function	With each rising edge of the signal at the counter input the actual value is incremented by 1. (It's not necessary to feed the counter input with a pulse signal.)
Count direction	Incrementing
Setpoint value range	1 to 32767
Setpoint value entry	Directly as a decimal constant (K) in the instruction, or indirectly in a data register
Counter overflow behaviour	Counts to a maximum of 32,767, after which the counter value no longer changes.
Counter output	Once the setpoint value has been reached the output remains on.
Resetting	An RST instruction is used to delete the current value of the counter and turn off its output.

Counter overview

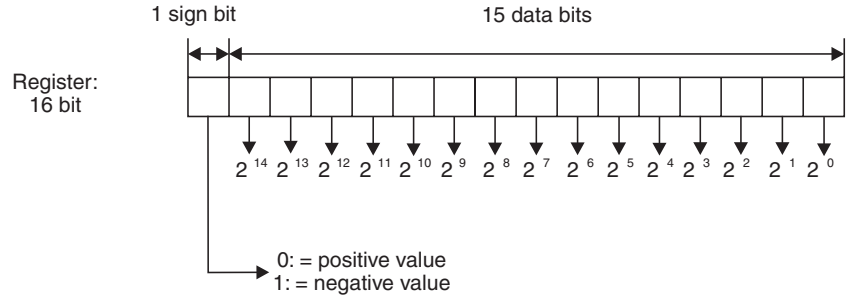
Device		Counter
Device identifier		C
Device type (for setting and polling)		Bit-Operand
Possible device values (counter output)		0 or 1
Device address format		Decimal
Counter setpoint value entry		As a decimal integer constant. The setpoint can be set either directly in the instruction or indirectly in a data register .
Number of devices and addresses	Q00J	512* (C0 to C511)
	Q00	
	Q01	
	Q02	1024* (C0 to C1023)
	Q02H	
	Q06H	
	Q12H	
	Q25H	
	Q12PH	
	Q25PH	

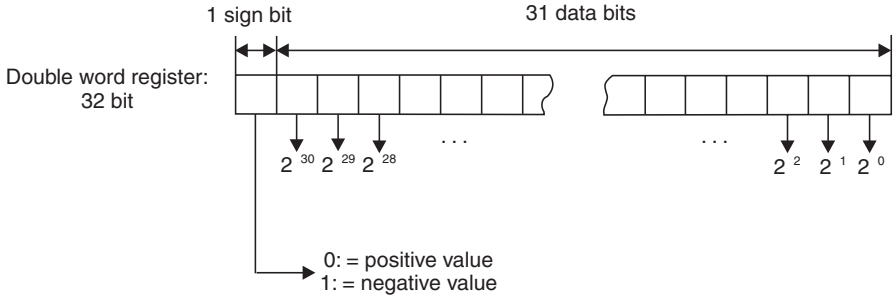
* Initial values, the number of counters can be set in the PLC parameters.

5.5 Registers

The PLC's relays are used to store the results of operations temporarily. However, relays can only store values of On/Off or 1/0, which means that they are not suitable for storing measurements or the results of calculations. Values like this can be stored in the "registers" of the controllers of the MELSEC System Q.

Registers are 16 bits or one word wide (see section 4.2). You can create "double word" registers capable of storing 32-bit values by combining two consecutive data registers.





A normal register can store values from 0000H to FFFFH (-32,768 to 32,767). Double-word registers can store values from 00000000H to FFFFFFFFH (-2,147,483,648 to 2,147,483,647).

The controllers of the MELSEC System Q have a large number of instructions for using and manipulating registers. You can write and read values to and from registers, copy the contents of registers, compare them and perform math functions on their contents (see chapter 6).

5.5.1 Data registers

Data registers can be used as memory in your PLC programs. A value that the program writes to a data register remains stored there until the program overwrites it with another value.

When you use instructions for manipulating 32-bit data you only need to specify the address of a 16-bit register. The more significant part of the 32-bit data is automatically written to the next consecutive register. For example, if you specify register D0 to store a 32-bit value D0 will contain bits 0 through 15 and D1 will contain bits 16 through 31.

What happens when the PLC is switched off or stopped

In addition to the normal registers whose contents are lost when the PLC is stopped or the power supply is turned off, the MELSEC System Q CPUs also have latched registers, whose contents are retained in these situations.

Data register overview

Device		Data register
Device identifier		D
Device type (for setting and polling)		Word device (two registers can be combined to store double-word values)
Possible device values		16 bit registers: 0000H to FFFFH (-32768 to 32767) 32 bit register: 00000000H to FFFFFFFFH (-2 147 483 648 to 2 147 483 647)
Device address format		Decimal
Number of devices and addresses	Q00J	11136* (D0 to D11135)
	Q00	
	Q01	
	Q02	12288* (D0 to D12287)
	Q02H	
	Q06H	
	Q12H	
	Q25H	
	Q12PH	
	Q25PH	

* Initial values, the number of data registers can be set in the PLC parameters.

5.5.2 Special registers

Just like the special relays (Chapter 5.2.1) the controllers of the MELSEC System Q also have special registers. The device identifier of these registers is "**SD**". Often there is also a direct connection between the special relays and special registers. For example, special relay SM51 shows that the voltage of the PLC's battery is too low, and the contents of special register SM51 provides information about which battery (CPU or memory card) is empty. The following table shows a small selection of the available special registers as examples.

Special register	Function	Program processing options
SD0	Error code	Read register contents
SD392	Software version	
SD520, SD521	Current program cycle time	
SD210–SD213	Time and date of the integrated real-time clock (BCD format)	Read register contents Change register contents

NOTE

For an overview of all special registers please refer to the Programming Manual for the A/Q series and the MELSEC System Q, art. no. 87431.

5.5.3 File registers

The contents of file registers are also not lost when the power supply is switched off. File registers can thus be used for storing values that you need to transfer to data registers when the PLC is switched on, so that they can be used by the program for calculations, comparisons or as setpoints for timers.

File registers have the same structure as data registers.

Device		File registers
Device identifier		R
Device type (for setting and polling)		Word device (two registers can be combined to store double-word values)
Possible device values		16 bit register: 0000_H to $FFFF_H$ (-32768 to 32767) 32 bit register: 00000000_H to $FFFFFFFF_H$ (-2 147 483 648 to 2 147 483 647)
Device address format		Decimal
Number of devices and addresses	Q00J	0
	Q00	32767 (R0 to R32766)
	Q01	
	Q02	32767 in each block (R0 to R32766) When a memory card is used, up to 1 million additional file registers can be stored.
	Q02H	
	Q06H	
	Q12H	
	Q25H	
	Q12PH	
	Q25PH	

5.6 Constants

5.6.1 Decimal and Hexadecimal constants

Decimal and hexadecimal constants are devices which designate decimal resp. hexadecimal data in sequence programs (e. g. setpoints for Timer and Counter). The constant is converted by the PLC CPU into a binary number.

Decimal constants are not particularly denoted in the Ladder Diagram or the IEC instruction list. Hexadecimal constants are prefixed with "16#". For example the notation 16#12 is interpreted by the PLC CPU as hexadecimal value 12.

In the MELSEC instruction list decimal constants are prefixed with the letter "K" and hexadecimal constants are prefixed with the letter "H". Examples: K100 = decimal value 100; K64 = hexadecimal value 64.

The following table shows the value ranges of the decimal and hexadecimal constants

Constants	16 bit	32 bit
Decimal	-32 768 to +32 767	-2 147 483 648 to +2 147 483 647
Hexadecimal	0 to FFFF	0 to FFFFFFFF

5.6.2 Floating decimal point constants

Decimal constants are integer values. Floating decimal point values (or real numbers) however have decimal places and therefore offer advantages for arithmetical operations.

Floating decimal point constants are prefixed by an "E" in the sequence program (for example E1.234 or E1.234 + 3). As you see, these constants can be designated in the program by an expression with or without exponent.

– Designation of a constant without exponent

  The specified value is designated in the "normal" way. For example 10.2345 becomes "E10.2345".

– Designation of a constant with exponent

  The value is divided into a base and an exponent. The base of the exponent is 10 (10^n). For example, 1234 can be expressed as 1.234 x 1000 or – in exponential expression – as 1.234×10^3. In the sequence program this value becomes E1.234 + 3. (+3 represents 10^3).

The value ranges for floating decimal point constants are:
-1.0×2^{128} to -1.0×2^{-126},
0
and 1.0×2^{-126} to $1.0 \times 2^{+128}$

5.6.3 Character string constants

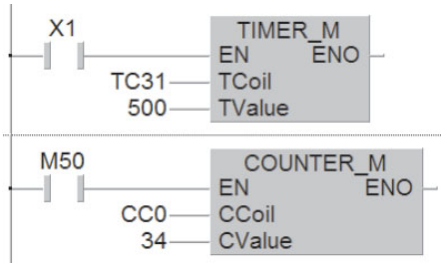
When characters are designated in the sequence program by quotation marks they are interpreted as ASCII code (e. g. "MOTOR12"). One character occupies 1 byte. You can use up to 32 characters for a character string.

5.7 Programming Tips for Timers and Counters

5.7.1 Specifying timer and counter setpoints indirectly

The usual way to specify timer and counter setpoint values is directly, in an output instruction:

Ladder Diagram



MELSEC Instruction List

```
LD      X1
OUT     T31
        K500
LD      M50
OUT     C0
        K34
```

IEC Instruction List

```
LD              X1
TIMER_M         TC31, 500
LD              M50
COUNTER_M       CC0, K34
```

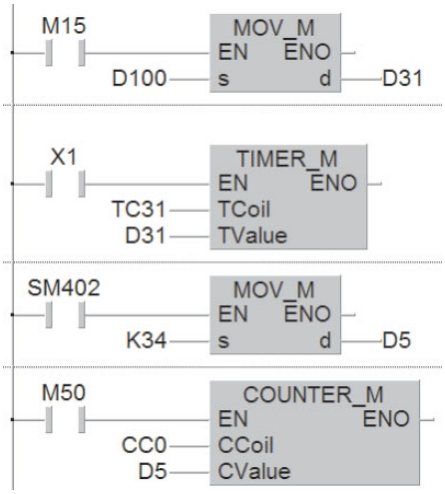
In the example above T31 is a 100ms timer. The constant K500 sets the delay to 500 x 0.1s = 50s. The setpoint for counter C0 is also set directly, to a value of 34 with the constant K34.

The advantage of specifying setpoints like this is that you don't have to concern yourself with the setpoint value once you have set it. The values you use in the program are always valid, even after power failures and directly after switching the controller on. However, there is also a disadvantage: If you want to change the setpoint you need to edit the program. This applies particularly for timer setpoint values, which are often adjusted during controller configuration and program tests.

You can also store setpoint values for timers and counters in data registers and have the program read them from the registers. It is then possible to change the values quickly with a programming unit if necessary, or to specify setpoint values with switches on a control console or a HMI control panel.

The listing shown on the next page is an example of how to specify setpoint values indirectly:

Ladder Diagram



MELSEC Instruction List

```
LD      M15
MOV     D100
        D31
LD      X1
OUT     T31
        D131
LD      SM402
MOV     K34
        D5
LD      M50
OUT     C0
        D5
```

IEC Instruction List

```
LD          M15
MOV_M       D100, D31
LD          X1
TIMER_M     TC31, D31
LD          SM402
MOV_M       K34, D5
LD          M50
COUNTER_M   CC0, D5
```

– When relay M15 is on the contents of data register D100 is copied to D31. This register contains the setpoint value for T31. You could use a programming or control unit to adjust the contents of D100.

– The special relay SM402 is only set for a single program cycle directly after the PLC is switched on. This is used to copy the constant value 34 into data register D5, which is then used as the setpoint value for counter C0.

You don't have to write program instructions to copy the setpoint values to the data registers. You could also use a programming unit to set them before the program is started, for example.

WARNING:
If you use normal registers the setpoint values will be lost when the power supply is switched off and when the RUN/STOP switch is set to the STOP position. If this happens hazardous conditions may be created next time the power is switched on and/or when the PLC is started again, because all the set points will have a value of "0".

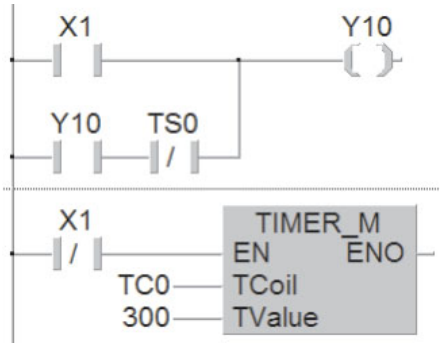
If you don't configure your program to copy the values automatically you should always use latched data registers for storing the setpoint values for timers and counters. Also, remember that even the contents of these registers will also be lost when the PLC is switched off if the backup battery is empty.

5.7.2 Switch-off delay

By default, all the timers in MELSEC PLCs are delayed make timers, i.e. the output is switched ON after the defined delay period. However, you will often also want to program a delayed break operation (switch OFF after a delay). A typical example of this is a ventilation fan in a bathroom that needs to continue running for several minutes after the lights are switched off.

Program version 1 (latching)

Ladder Diagram



MELSEC Instruction List

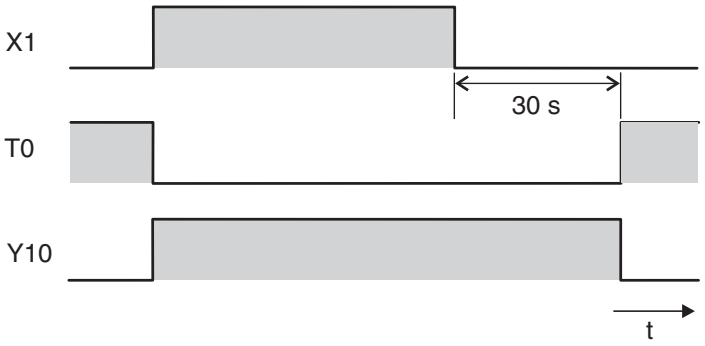
```
LD      X1
LD      Y10
ANI     T0
ORB
OUT     Y10
LDI     X1
OUT     T0
        K300
```

IEC Instruction List

```
LD      X1
OR(
        Y10
ANDN    TS0
)
ST      Y10
LDN     X1
TIMER_M TC0, 300
```

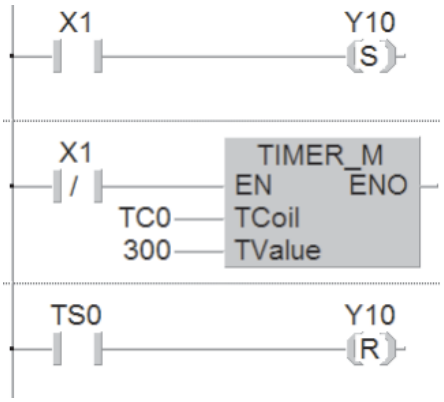
As long as input X1 (e.g. a light switch) is on output Y10 (fan) is also on. However, the latching function ensures that Y10 also remains on after X1 has been switched off, because timer T0 is still running. T0 is started when X1 is switched off. At the end of the delay period (300 x 0.1s = 30s in the example) T0 interrupts the Y10 latch and switches the output off.

Signal sequence



Program version 2 (set/reset)

Ladder Diagram



MELSEC Instruction List

```
LD      X1
SET     Y10
LDI     X1
OUT     T0
        K300
LD      T0
RST     Y000
```

IEC Instruction List

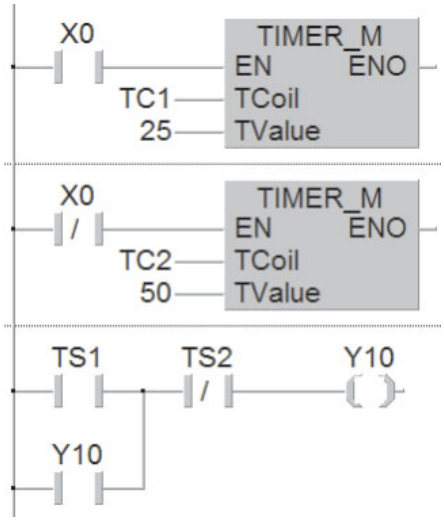
```
LD      X1
S       Y10
LDN     X1
TIMER_M TC0, 300
LD      TS0
R       Y10
```

When X1 is switched on output Y10 is set (switched on). When X1 is switched off timer T0 is started. After the delay period T0 then resets output Y10. The resulting signal sequence is identical with that produced by program version 1.

5.7.3 Delayed make and break

Sometimes you will want to switch an output on after a delay and then switch it off again after another delay. This is very easy to implement with the controller's basic logical instructions.

Ladder Diagram



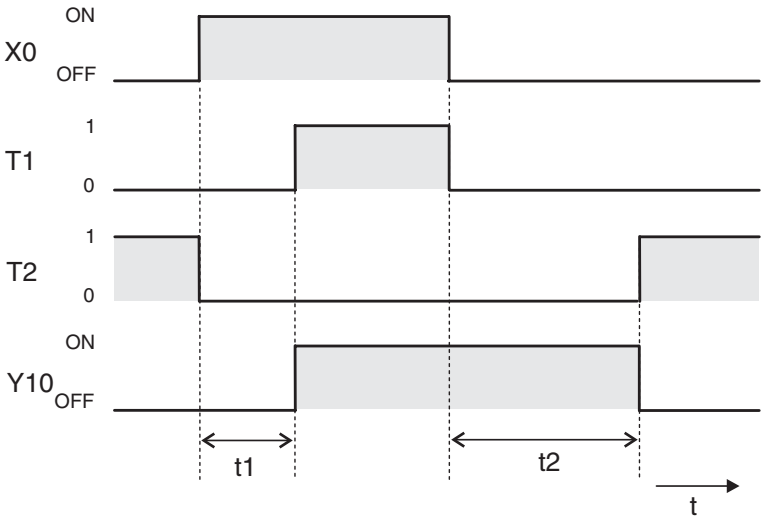
MELSEC Instruction List

```
LD      X0
OUT     T1
        K25
LDI     X0
OUT     T2
        K50
LD      T1
OR      Y10
ANI     T2
OUT     Y10
```

IEC Instruction List

```
LD      X0
TIMER_M TC1, 25
LDN     X0
TIMER_M TC2, 50
LD      TS1
OR      Y10
ANDN    TS2
ST      Y10
```

Signal sequence



When X0 is switched on T1 is started and T2 reset. At the end of the delay period t1 the output Y10 is switched on. It keeps on as long as X0 is on.

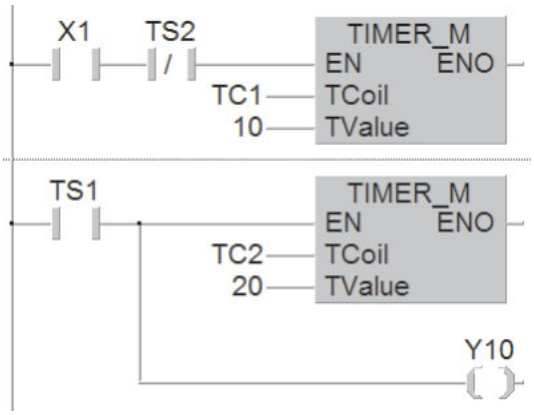
But even when X0 is switched off und T1 is reset Y10 remains on because of the latching function. When X1 is switched off timer T2 is started. After the delay period t2 the output Y10 is reset.

5.7.4 Clock signal generators

The controllers have special relays that make it very easy to program tasks requiring a regular clock signal (for example for controlling a blinking error indicator light). Relay SM413 switches on and off at 1-second intervals, for example. For full details on all special relays see the Programming Manual for the A/Q series and the MELSEC System Q, Art. no. 87431.

If you need a different clock frequency or different on and off times you can program your own clock signal generator with two timers, like this:

Ladder Diagram



MELSEC Instruction List

```
LD      X1
ANI     T2
OUT     T1
        K10
LD      T1
OUT     T2
        K20
OUT     Y10
```

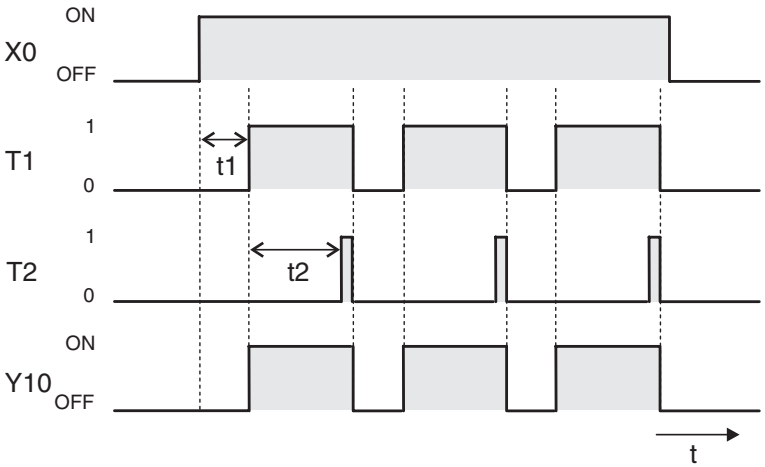
IEC Instruction List

```
LD      X1
ANDN    TS2
TIMER_M TC1, 10
LD      TS1
TIMER_M TC2, 20
ST      Y10
```

Input X1 starts the clock generator. If you want, you can omit this input – then the clock generator is always on. In the program you could use the output of T1 to control a blinking warning light. The on period is determined by T2, the off period by T1.

The output of timer T2 is only switched on for a single program cycle. This time is shown much longer than it really is in the signal sequence illustration below. T2 switches T1 off and immediately after this T2 itself is also switched off. In effect this means that the duration of the on period is increased by the time that it takes to execute a program cycle. However, since the cycle is only a few milliseconds long it can usually be ignored.

Signal sequence



6 More Advanced Programming

The basic logic instructions listed in Chapter 3 can be used to emulate the functions of a hard-wired contactor controller with a programmable logic controller. However, this only scratches the surface of the capabilities of modern PLCs. Since every PLC is built around a microprocessor they can also easily perform operations like mathematical calculations, comparing numbers, converting from one number system to another or processing analog values.

Functions like these that go beyond the capabilities of logic operations are performed with special instructions, which are referred to as *applied* or *application instructions*.

6.1 Applied Instructions Reference

Applied instructions have short names that are based on the English names of their functions. For example, the instruction for moving 16-bit data is called MOV.

When you program an applied instruction you enter the instruction name followed by the device name. The following table shows all the applied instructions currently supported by the MELSEC System Q family of controllers. This list may look a little overwhelming at first, but don't worry – you don't have to memorise them all! When you are programming you can use the powerful Help functions of GX Developer and GX IEC Developer to find the instructions you need. In this chapter we will only cover the more frequently-used instructions, which are shown with a grey shaded background in the reference table. For full documentation of all the instructions with examples please refer to the Programming Manual for the A/Q series and the MELSEC System Q, art. no. 87431.

NOTE

Many applied instructions can be executed cyclical or pulse triggered (with the rising edge of the input condition). In this case the name of the instruction is suffixed by the letter "P". For example: **MOV** -> execution in each program cycle as long as the input condition is true; **MOVP** -> one-time execution on the rising edge of the signal pulse generated by the input condition.

Category		Instruction	Function
Comparison operations	Compare 16-bit data within operations	**LD=**	Compare for „equals“
		LD>	Compare for „greater than“
		LD<	Compare for „less than“
		LD<>	Compare for „not equal“
		LD<=	Compare for „less than or equal to“
		LD>=	Compare for „greater than or equal to“
		AND=	Compare for „equals“
		AND>	Compare for „greater than“
		AND<	Compare for „less than“
		AND<>	Compare for „not equal“
		AND<=	Compare for „less than or equal to“
		AND>=	Compare for „greater than or equal to“
		OR=	Compare for „equals“
		OR>	Compare for „greater than“
		OR<	Compare for „less than“
		OR<>	Compare for „not equal“
		OR<=	Compare for „less than or equal to“
		OR>=	Compare for „greater than or equal to“

Category		Instruction	Function
Comparison operations	32-bit data comparisons	LDD=	Compare 16-bit data within operations
		LDD>	
		LDD<	
		LDD<>	
		LDD<=	
		LDD>=	
		ANDD=	
		ANDD>	
		ANDD<	
		ANDD<>	
		ANDD>=	
		ANDD<=	
		ORD=	
		ORD>	
		ORD<	
		ORD<>	
		ORD<=	
		ORD>=	
	Real number data comparisons	LDE=	Compare data within operations
		LDE>	
		LDE<	
		LDE<>	
		LDE<=	
		LDE>=	
		ANDE=	
		ANDE>	
		ANDE<	
		ANDE<>	
		ANDE>=	
		ANDE<=	
		ORE=	
		ORE>	
		ORE<	
		ORE<>	
		ORE<=	
		ORE>=	
	Character string data comparisons	LD$=	Compare of two character strings (one character at a time) within operations
		LD$>	
		LD$<	
		LD$<>	
		LD$<=	
		LD$>=	
		AND$=	
		AND$>	
		AND$<	
		AND$<>	
		AND$>=	
		AND$<=	
		OR$=	
		OR$>	
		OR$<	

Category		Instruction	Function
Comparison operations	Character string data comparisons	**OR$<>**	Compare of two character strings (one character at a time) within operations
		OR$<=	
		OR$>=	
	Block data comparison	**BKCMP=**	Compares BIN 16-bit data stored in consecutive devices (data blocks). The number of data blocks is specified with the instruction. The result is stored in a separate area.
		BKCMP>	
		BKCMP<	
		BKCMP<>	
		BKCMP<=	
		BKCMP>=	
Math instructions	Addition and Subtraction	**+**	Add 16-bit binary data
		-	Subtract 16-bit binary data
		D+	Add 32-bit binary data
		D-	Subtract 32-bit binary data
		B+	Add 4-digit BCD values
		B-	Subtract 4-digit BCD values
		DB+	Add 8-digit BCD values
		DB-	Subtract 8-digit BCD values
		E+	Add floating decimal point values
		E-	Subtract floating decimal point values
		BK+	Add BIN 16-bit data stored in data blocks
		BK-	Subtract BIN 16-bit data stored in data blocks
	Multiplication and division	**x**	Multiply 16-bit binary data
		/	Divide 16-bit binary data
		Dx	Multiply 32-bit binary data
		D/	Divide 32-bit binary data
		Bx	Multiply 4-digit BCD values
		B/	Divide 4-digit BCD values
		DBx	Multiply 8-digit BCD values
		DB/	Division 8-digit BCD values
		Ex	Multiply floating decimal point values
		E/	Divide floating decimal point values
	Character string data combination	**S+**	Link one character string to another
	Increment and Decrement	**INC**	Increment (add "1" to the actual value) of 16-bit binary data
		DINC	Increment of 32-bit binary data
		DEC	Decrement (subtract "1" from the actual value) of 16-bit binary data
		DDEC	Decrement of 32-bit binary data
Data conversion instructions	Binary data -> BCD	**BCD**	Converts 16-bit binary data to BCD
		DBCD	Converts 32-bit binary data to BCD
		BKBCD	Converts 16-bit binary data blocks to BCD
	BCD-> Binary data	**BIN**	Converts 4-digit BCD values to binary data
		DBIN	Converts 8-digit BCD values to binary data
		BKBIN	Converts block 4-digit BCD values to block binary data
	Binary data -> floating decimal point	**FLT**	Converts 16-bit binary data to floating decimal point value
		DFLT	Converts 32-bit binary data to floating decimal point value
	Floating decimal point -> Binary data	**INT**	Converts a floating decimal point value to 16-bit binary
		DINT	Converts a floating decimal point value to 32-bit binary
	Binary data -> Binary data	**DBL**	Converts 16-bit binary data to 32-bit binary data
		WORD	Converts 32-bit binary data to 16-bit binary data

Category		Instruction	Function
Data conversion instructions	Binary data -> Gray code	**GRY**	Converts 16-Bit binary data to Gray code
		DGRY	Converts 16-Bit binary data to Gray code
	Gray code -> binary data	**GBIN**	Converts Gray code to 16-Bit binary data
		DGBIN	Converts Gray code to 32-Bit binary data
	Sign reversal	**NEG**	Complement of 2 (sign reversal) of 16-Bit binary data
		DNEG	Complement of 2 (sign reversal) of 32-Bit binary data
		ENEG	Sign reversal for floating decimal point values
Move functions	for 16-bit binary data	**MOV**	Move single 16-bit data from one storage area to another
		BMOV	Block data transfer
		FMOV	Fill Move, copy to a range of devices
		XCH	Exchange data in specified devices
		BXCH	Exchange block data
		SWAP	Exchange upper and lower byte of a word
		EROMWR	Write data to a EEPROM register
	for 32-bit binary data	**DMOV**	Move single 32-bit data from one storage area to another
		DXCH	Exchange data in specified devices
	for floating decimal point values	**EMOV**	Move floating decimal point values
	for character strings	**$MOV**	Move character string
	Negation, logical inversion of device contents	**CML**	Inverts 16-bit binary data bit by bit
		DCML	Inverts 32-bit binary data bit by bit
	for files	**SP.FWRITE**	Write to file
		SP.FREAD	Read from file
	for block data	**RBMOV**	High-speed block transfer of file registers
Program branch instructions		**CJ**	Conditional Jump to a program position
		SCJ	Conditional Jump from next program scan on
		JMP	Jump instruction
		GOEND	Jump to the end of a program
Program execution control instructions	Enable interrupts	**EI**	Enable the call of an interrupt program
	Disable interrupts	**DI**	Disable the execution of an interrupt program
	Enable/Disable individual interrupts	**IMASK**	Control of the execution conditions of interrupt programs
	End of an interrupt program	**IRET**	Return from an interrupt programm to the main program
Data refresh instructions	Inputs and outputs	**RFS**	Refresh the inputs and outputs of an designated range of I/O devices during one program scan.
	Link and interface data	**COM**	Refresh of link and interface data
	Execution condition for a link refresh	**DI**	Disable link refresh execution
		EI	Enable link refresh execution
Logical operation instructions	Logical AND	**WAND**	Combination of two 16-bit devices
		DAND	Combination of two 32-bit devices
		BKAND	Combination of 16-bit devices in data blocks
	Logical OR	**WOR**	Combination of two 16-bit devices
		DOR	Combination of two 32-bit devices
		BKOR	Combination of 16-bit devices in data blocks
	Logical exklusive OR (XOR)	**WXOR**	Combination of two 16-bit devices
		DXOR	Combination of two 32-bit devices
		BKXOR	Combination of 16-bit devices in data blocks
	Logical exklusive NOR (XNR)	**WNXR**	Combination of two 16-bit devices
		DNXR	Combination of two 32-bit devices
		BKXNR	Combination of 16-bit devices in data blocks

Category		Instruction	Function
Data rotation instructions	16-bit data	**ROR**	Rotation of bits to the right
		RCR	Rotation of bits to the right with carry flag
		ROL	Rotation of bits to the left
		RCL	Rotation of bits to the left with carry flag
	32-bit data	**DROR**	Rotation of bits to the right
		DRCR	Rotation of bits to the right with carry flag
		DROL	Rotation of bits to the left
		DRCL	Rotation of bits to the left with carry flag
Data shift instructions	16-bit data	**SFR**	Shifting a 16-bit data word n bits to the right (n: 0 to 15)
		SFL	Shifting a 16-bit data word n bits to the left (n: 0 to 15)
	Bit devices	**BSFR**	Shifting of a number of bit devices 1 bit to the right
		BSFL	Shifting of a number of bit devices 1 bit to the left
	Word devices	**DSFR**	Shifting of a number of word devices 1 bit to the right or the left
		DSFL	
Bit processing instructions	Setting/resetting	**BSET**	Setting of single bits
		BRST	Resetting of single bits
		BKRST	Batch reset of bits
	Bit test	**TEST**	Test of single bits in 16-/32-Bit data words
		DTEST	
Data processing instructions	Search data	**SER**	Search 16-bit data
		DSER	Search 32-bit data
	Check data bits	**SUM**	Determine the number of bits set in a 16-/32-bit data word
		DSUM	
	Decoding data	**DECO**	Decode from 8 to 256 bit (binary to decimal)
	Encoding data	**ENCO**	Encode from 256 to 8 bit (decimal to binary)
	7-segment decoding	**SEG**	Converts a 4-digit binary value into 7-segment code in order to display the values 0 to F
	Disunite/unite 16-bit data words	**DIS**	Disunite 16-bit data words into groups of 4 bit
		UNI	Store each 4 lowest bits of up to four 16-bit data values in a 16-bit data value
		NDIS	Disunite data in random bit units
		NUNI	Unite data in random bit units
		WTOB	Disunite data in byte units
		BTOW	Unite data in byte units
	Search maximum values	**MAX**	Search maximum value in 16-bit data blocks
		DMAX	Search maximum value in 32-bit data blocks
	Search minimum values	**MIN**	Search minimum value in 16-bit data blocks
		DMIN	Search minimum value in 32-bit data blocks
	Sort	**SORT**	Sort 16-bit data
		DSORT	Sort 32-bit data
	Calculation of totals	**WSUM**	Calculate totals of 16-bit binary data blocks
		DWSUM	Calculate totals of 32-bit binary data blocks
Structured program instructions	Repetition instructions	**FOR**	Start of a program repetition
		NEXT	End of a program repetition
		BREAK	Terminate the FOR/NEXT loop

Category		Instruction	Function
Structured program instructions	Subroutine programs	**CALL**	Subroutine program call
		RET	End of subroutine
		FCALL*	Reset outputs in subroutines
		ECALL*	Calling a subroutine program in a program file
		EFCALL*	Reset outputs in subroutine programs in program files
	Index qualification	**IX**	Index qualification of an entire program parts
		IXEND	
		IXDEV	Store indexed device numbers in an index qualification list
		IXSET	
Data table operation instructions	Write data	**FIFW**	Write data to a data table
	Read data	**FIFR**	Read data entered first from data table
		FPOP	Read data entered last from data table
	Delete data	**FDEL**	Delete specified data blocks from data table
	Insert data	**FINS**	Insert specified data blocks in data table
Buffer memory access instructions	Read	**FROM**	Read 16-bit data from special function module
		DFRO	Read 32-bit data from special function module
	Write	**TO**	Write 16-bit data to special function module
		DTO	Write 32-bit data to special function module
Display instructions	ASCII character output	**PR**	Output of an ASCII character string to a peripheral device
		PRC	Output of a comment (in ASCII code) to a peripheral device
	Clear display	**LEDR**	Reset annunciators and LED display
Failure diagnosis and debugging	Failure check	**CHKST**	Start instruction for the CHK instruction
		CHK	Failure check
		CHKCIR	Generate check circuits for the CHK instruction
		CHKEND	End instruction for a program part with generated check circuits
	Store device status	**SLT**	Set status latch (Store device status)
		SLTR	Reset status latch (clear device status)
	Sampling trace	**STRA**	Set sampling trace
		STRAR	Reset sampling trace
	Program trace	**PTRA**	Set program trace
		PTRAR	Reset program trace
		PTRAEXE	Execute program trace
	Trace	**TRACE**	Set trace
		TRACER	Clear data stored by the trace instruction
Character string processing instructions	Binary -> Decimal (ASCII)	**BINDA**	Convert 16-/32-bit binary data into decimal values in ASCII code
		DBINDA	
	Binary -> Hexadecimal (ASCII)	**BINHA**	Convert 16-/32-bit binary data into hexadecimal values in ASCII code
		DBINHA	
	BCD -> ASCII	**BCDDA**	Convert 4-digit BCD data into ASCII-Code
		DBCDDA	Convert 8-digit BCD data into ASCII-Code
	Decimal (ASCII) -> Binary	**DABIN**	Convert decimal ASCII into 16-/32-bit binary data
		DDABIN	
	Hexadecimal (ASCII) -> Binary	**HABIN**	Convert decimal ASCII into 16-/32-bit binary data
		DHABIN	

* It is not possible to program the instructions FCALL, ECALL and EFCALL with the GX IEC Developer.

Category		Instruction	Function
Character string processing instructions	Decimal (ASCII) -> BCD	**DABCD**	Convert decimal ASCII into 4-digit BCD data
		DDABCD	Convert decimal ASCII into 8-digit BCD data
	Read device comment data	**COMRD**	Read device comment data and store as ASCII
	Length detection	**LEN**	Detect the length of character strings
	Binary data -> character string	**STR**	Insert a decimal point and convert 16-/32-bit binary data into a character string
		DSTR	
	Character string -> Binary data	**VAL**	Convert character strings into 16-/32-bit binary data
		DVAL	
	Floating point data -> Character string	**ESTR**	Convert floating point data into character strings
	Character string -> Floating point data	**EVAL**	Convert character strings into floating point data
	Floating point data -> BCD	**EMOD**	Convert floating point data into BCD data
	Floating point data -> Deicmal	**EREXP**	Convert floating point data in BCD format into decimal format
	BIN 16-bit data -> ASCII	**ASC**	Convert 16-bit binary data into ASCII code
	ASCII -> Binary	**HEX**	Convert hexadecimal ASCII characters into binary values
	Extraction of character string data	**RIGHT**	Extract substring from right
		LEFT	Extract substring from left
	Storing	**MIDR**	Store specified parts of character strings
	Move	**MIDW**	Move parts of character strings to a defined area
	Search	**INSTR**	Search for a character string
Instructions for floating point numbers	Trigonometry instructions	**SIN**	Calculate the sine
		COS	Calculate the cosine
		TAN	Calculate the tangent
		ASIN	Calculate the arc sine
		ACOS	Calculate the arc cosine
		ATAN	Calculate the arc tangent
		RAD	Convert degrees to radians
		DEG	Convert radians to degrees
	Math instructions	**SQR**	Calculate square root
		EXP	Floating point value as exponent of the base e
		LOG	Logarithm calculation
Special functions	Randomizing values	**RND**	Generate a random value
		SRND	Update series of random values
Instructions for BCD data	Trigonometry instructions	**BSIN**	Calculate the sine
		BCOS	Calculate the cosine
		BTAN	Calculate the tangent
		BASIN	Calculate the arc sine
		BACOS	Calculate the arc cosine
		BATAN	Calculate the arc tangent
	Math instructions	**BSQR**	Calculate square root from 4-digit BCD data
		BDSQR	Calculate square root from 8-digit BCD data
Data control instructions	Limit control	**LIMIT**	Limitation of output values for 16-/32-bit binary data
		DLIMIT	
	Dead band control	**BAND**	Dead band control of 16-/32-bit binary data
		DBAND	
	Zone control	**ZONE**	Zone control of 16-/32-bit binary data
		DZONE	

Category		Instruction	Function
Instructions for file register	Switching instructions	**RSET**	Switch between file register blocks
		QDRET	Switch between files in file registers
		QCDSET	Switch between comment files in file registers
	Read	**ZRRDB**	Read from byte in file register directly
	Write	**ZRWRB**	Write directly in a byte in a file register
Operations with the PLC's integrated clock	Read	**DATERD**	Read clock time and date
	Set	**DATEWR**	Write time and date to PLC clock
	Add	**DATE+**	Add clock data
	Subtract	**DATE-**	Subtract clock data
	Conversion	**SECOND**	Convert hours / minutes / seconds time value to seconds
		HOUR	Convert seconds time value to hours / minutes / seconds
Peripheral device instructions	Output	**MSG**	Output of messages to peripheral devices
	Input	**PKEY**	Key input of data from peripheral devices
		KEY	Key input of numerical values
Program control instructions	Stand-by mode	**PSTOP**	Switch program into stand-by mode.
		POFF	Switch program into stand-by mode and reset outputs.
	Scan execution mode	**PSCAN**	Switch program into scan execution mode
	Low speed execution mode	**PLOW**	Switch program into low speed execution mode
Instructions for programs	Load program	**PLOADP**	Load program from a memory card
	Delete program	**PUNLOADP**	Delete a stand-by program from program memory
	Delete and load	**PSWAPP**	Delete a stand-by program from program memory and load program from a memory card
Data link instructions	Refresh	**ZCOM**	Network data refresh
	Routing	**RTREAD**	Read routing information
		RTWRITE	Write routing information
Instructions for use in a Multi-CPU system	Write data	**S.TO**	Write data to the CPU shared memory
	Read data	**FROM**	Read data from shared memory of another CPU
	Data refresh	**COM**	Refresh shared memory used by multi-CPU system
Instructions for system control	Watchdog timer	**WDT**	Reset watchdog timer
	Module information	**UNIRD**	Read module information
	Index register	**ZPUSH**	Batch save of index register contents
		ZPOP	Batch recovery of index register contents
	Store device address	**ADRSET**	Store device address for indirect designation (not available with GX IEC Developer)
	Execution scans	**DUTY**	Preset execution scans of a device
Application instructions	Counter	**UDCNT1**	1-Phase input count-up/-down counter
		UDCNT2	2-Phase input count-up/-down counter
	Timer	**TTMR**	Programmable timer
		STMR	Special function timer (low-speed timer)
		STMRH	Special function timer (high-speed timer)
	Instruction für rotary table	**ROTC**	Positioning of rotary table
	Ramp signal	**RAMP**	Change the content of a device gradually
	Pulse density measurement	**SPD**	Count pulses at an input for a specified time and store the result.
	Pulse output	**PLSY**	Pulse output with adjustable number of pulses
	Puls width modulation	**PWM**	Pulse output with adjustable cycle time and ON time
	Input matrix	**MTR**	Building of an input matrix for reading of information

Category		Instruction	Function
Instructions for serial communication modules	Read data	**BUFRCVS**	Reading of received data from an interface module
	Write data	**PRR**	Send data via the interface module by means of user frames
	User registered frames	**GETE**	Read user registered frames
		PUTE	Register / delete user frames
Instructions for PROFIBUS/DP interface modules	Read data	**BBLKRD**	Read data from the buffer memory of a PROFIBUS/DP interface module and store the data in the PLC CPU
	Write data	**BBLKWR**	Move data from the PLC CPU to the buffer memory of a PROFIBUS/DP interface module
Instructions for ETHERNET interface modules	Read data	**BUFRCV**	Read received data from fixed buffers
		BUFRCVS	
	Write data	**BUFSND**	Move data from the PLC CPU to the ETHERNET interface module
	Open connection	**OPEN**	Open connection
	Close connection	**CLOSE**	Close connection
	Error reset	**ERRCLR**	Clear error code and turn off "ERR." LED
	Read error code	**ERRRD**	Read error code from buffer memory
	Re-initialization	**UINI**	Re-initialization of an ETHERNET interface module
Instructions for CC-Link	Parameter setting	**RLPASET**	Parameter setting for a CC-Link network and start of the data link
	Read data	**RIRD**	Read from buffer memory of intelligent device station or from device memory of PLC CPU
		RICV	Read data from the buffer memory of an intelligent device station (with handshake).
		RIFR	Read data from another station entered in automatic updating buffer memory of CC-Link master.
	Write data	**RIWT**	Write to buffer memory of intelligent device station or to device memory of PLC CPU.
		RISEND	Write (with handshake) to the buffer memory of an intelligent device station.
		RITO	Write data from the PLC CPU to the automatic updating buffer memory of the CC-Link master. Afterwards the data will be send to the specified station.

6.1.1 Additional Instructions for Process CPUs

For the efficient programming of PID controls the following instructions are available for the Q12PHCPU and Q25PHCPU Process CPUs.

Category		Instruction	Function
I/O control instructions	Input	**IN**	Analog input processing (actual value)
	Output	**OUT1**	Output processing
		OUT2	
	Manual output	**MOUT**	Output processing in manual mode of the PID control
	PWM	**DUTY**	Output of a pulse width modulated signal (0 to 100 %)
	Compare	**BC**	Compare an input value with up to two set values and output of the result as bit data.
	Pulse retentive	**PSUM**	Integrate input value with limit detection and output the result.
Control operation instructions	PID control	**PID**	Basic PID control
		2PID	2-degree-of-freedom PID
		PIDP	Position type PID
	PI control	**SPI**	Sample PI
	I-PD control	**IPD**	I-PD control
	PI control	**BPI**	PI control
	2-position ON/OFF	**ONF2**	2-position ON/OFF
	3-position ON/OFF	**ONF3**	3-position ON/OFF
Signal processing	Rate operation	**R**	Limits the change rate of the output signal
	Limit alarms	**PHPL**	Checks an input value and issues alarms when upper or lower limits are reached
	Lead/lag	**LLAG**	The output of the LLAG instruction either follows the input delayed or leads the input.
	Integration	**I**	Performs integral operation of an input signal
	Differentiation	**D**	Performs derivative operation of an input signal
	Dead time	**DED**	Output of the input value with a delay of dead time.
	Output of maximum /intermediate/minimum value	**HS**	Outputs the maximum value from up to 16 input values.
		LS	Outputs the minimum value from up to 16 input values.
		MID	Outputs the intermediate value from up to 16 input values.
	Average value	**AVE**	Calculates the average value of up to 16 input values.
	Upper/lower limiter	**LIMT**	Limits an input value to an area defined by an upper and a lower limit.
	Change rate	**VLMT1**	Limits the varying speed of the output value
		VLMT2	
	Dead zone	**DBND**	An input value which is within the dead zone, is not output.
	Program setting device	**PGS**	Provides a control output according to a pattern
	Loop selector	**SEL**	In automatic mode either one of two input signals is output. In manual mode the manipulated value is output.
	Bump-less transfer	**BUMP**	Provides a bump-less transfer when switching from manual to automatic mode.
	Analog memory	**AMR**	Increases or decreases the output value at a fixed rate.

Category		Instruction	Function
Compensation and conversion operation instructions	Polygon	**FG**	The output value depends on the input value and an user defined polygon pattern.
	Inverted polygon	**IFG**	
	Filter	**FLT**	Samples an input value in specified intervals and calculates the average value.
	Retentive	**SUM**	Integrates the input value and outputs the result
	Temperature/pressure compensation	**TPC**	Performs temperature or pressure compensation for an input value. The result is output.
	Engineering value conversion	**ENG**	Converts an input value with the unit % to an output value with a physical unit.
	Engineering value reverse conversion	**IENG**	Converts an input value with a physical unit to an output value with the unit % .
Arithmetic operation instructions	Addition	**ADD**	Arithmetic operations with additional coefficients
	Subtraction	**SUB**	
	Multiplication	**MUL**	
	Division	**DIV**	
	Extraction	**SQR**	Calculates the square root of the input value
	Absolute value	**ABS**	The absolute value of the input value is output.
Comparison operation instructions	Compare for „greater than“	**> (GT)**	Comparison of to input values with hysteresis
	Compare for „less than“	**< (LT)**	
	Compare for „equals“	**= (EQ)**	
	Compare for „greater than or equal to“	**>= (GE)**	
	Compare for „less than or equal to“	**<= (LE)**	
Initial setting of PID constants	Autotuning	**AT1**	Automatic determination of the PID constants for a PID control with the instructions PID or 2PID

NOTE

For more information about PID control instructions please refer to the programming manual for the QnPHCPUs, Art. no. 149256.

6.2 Instructions for Moving Data

The PLC uses data registers for storing measurements, output values, intermediate results of operations and table values. The controller's math instructions can read their operands directly from the data registers and can also write their results back to the registers if you want. However, these instructions are also supported by additional "move" instructions, with which you can copy data from one register to another and write constant values to data registers.

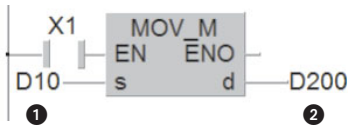
6.2.1 Moving individual values with the MOV instruction

The MOV instruction "moves" data from the specified source to the specified destination.

NOTE

Note that despite its name this is actually a copy process – it does not delete the data from the source location.

Ladder Diagram



MELSEC Instruction List

```
LD    X1
MOV   D10   ❶
      D200  ❷
```

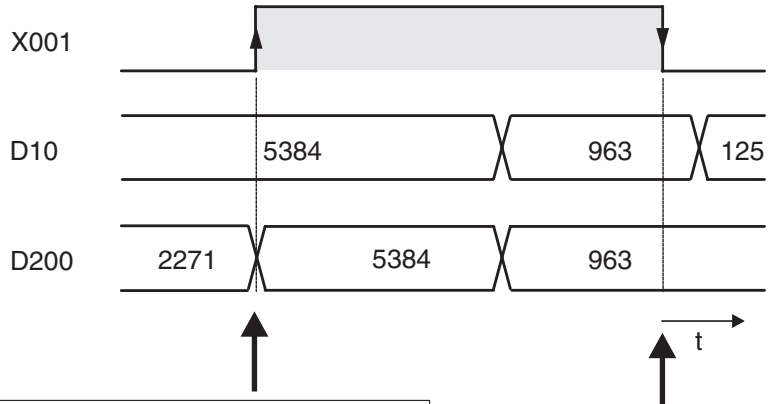
IEC Instruction List

```
LD    X1
MOV_M D10, D200
      ❶    ❷
```

❶ Data source (this can also be a constant). The "s" in Ladder Diagram instructions means *source*.

❷ Data destination (In Ladder Diagram instructions, "d" means *destination.*

In the example the value in data register D10 will be copied to register D200 when input X1 is on. This results in the following signal sequence:



The contents of the data source will be copied to the data destination as long as the input condition evaluates true. The copy operation does not change the contents of the data source.

When the input condition is no longer true the instruction will no longer change the contents of the data destination.

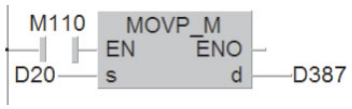
Pulse-triggered execution of the MOV instruction

In some applications it is better if the value is written to the destination in one program cycle only. For example, you will want to do this if other instructions in the program also write to the same destination or if the move operation must be performed at a defined time.

If you add a "P" to the MOV instruction (MOVP) it will only be executed **once**, on the rising edge of the signal pulse generated by the input condition.

In the example below the contents of D20 are written to data register D387 when the state of M110 changes from “0" to ”1".

Ladder Diagram



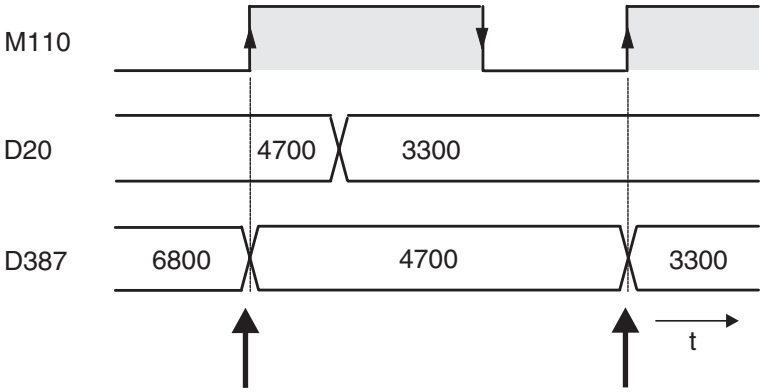
MELSEC Instruction List

```
LD      M110
MOVP    D20
        D387
```

IEC Instruction List

```
LD      M110
MOVP_M  D20, D387
        ❶  ❷
```

After this single operation has been performed copying to register D387 stops, even if the M110 remains set. The signal sequence illustrates this:

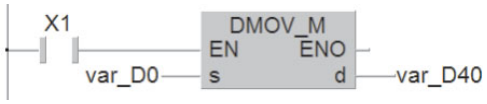


The contents of the data source are only copied to the destination on the rising pulse of the input condition.

Moving 32-bit data

To move 32-bit data just prefix a D to the MOV instruction (DMOV):

Ladder Diagram



MELSEC Instruction List

```
LD      X1
DMOV    D0      D40
```

IEC Instruction List

```
LD      X1
DMOV_M  var_D0,  varD40
```

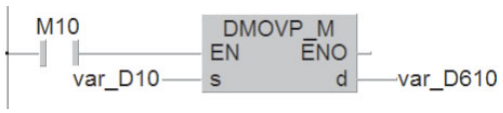
When input X1 is on the contents of the data registers D0 and D1 is written to data registers D40 and D41. (The contents of D0 is copied to D40 and the contents of D1 is copied to D41).

NOTE

With GX IEC Developer it is not possible in Ladder Diagram and the IEC instruction list to define 32-bit devices as input and output variables directly. These devices must be declared as Global Variables (see chapter 4.6.2). In this example the identifiers var_D0 and var_D40 were chosen to point to this fact.

As you might expect, there is also a pulse-triggered version of the 32-bit DMOV instruction:

Ladder Diagram



MELSEC Instruction List

```
LD      M10
DMOVP   D10     D610
```

IEC Instruction List

```
LD      X1
DMOVP_M var_D10, var_D610
```

When relay M10 is set the contents of registers D10 and D11 are written to registers D610 and D611.

NOTE

In Ladder Diagram and the IEC instruction list 32-bit devices have to be declared as Global Variables (see chapter 4.6.2). It is not possible to enter these devices directly.

6.2.2 Moving groups of bit devices

The previous section showed how you can use the MOV instruction to write constants or the contents of data registers to other data registers. Consecutive sequences of relays and other bit devices can also be used to store numerical values, and you can copy them as groups with applied instructions. To do this you prefixing a “K” factor to the address of the first bit device, specifying the number of devices you want to copy with the operation.

Bit devices are counted in groups of 4, so the K factor specifies the number of these groups of 4. K1 = 4 devices, K2 = 8 devices, K3 = 12 devices and so on.

For example, K2M0 specifies the 8 relays from M0 through M7. The supported range is K1 (4 devices) to K8 (32 devices).

Examples for addressing groups of bit devices:

– K1X0: 4 inputs, start at X0 (X0 to X3)

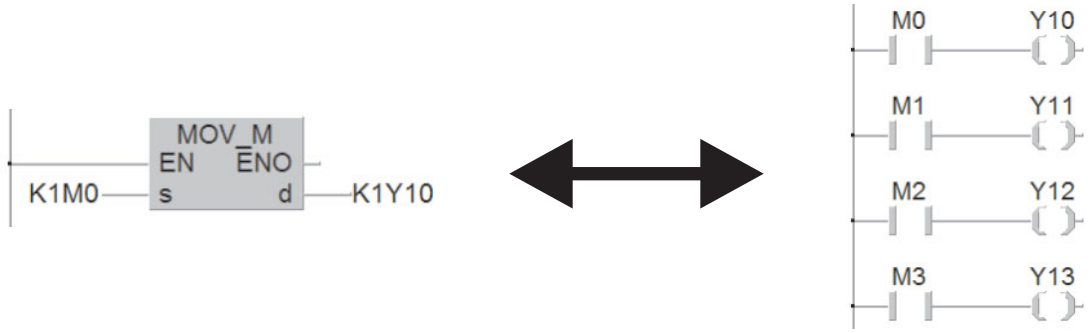
– K2X4: 8 inputs, start at X4 (X4 to X1B, hexadecimal notation)

– K4M16: 16 relays, start at M16 (M16 to M31)

– K3Y0: 12 outputs, start at Y0 (Y0 to Y1B, hexadecimal notation)

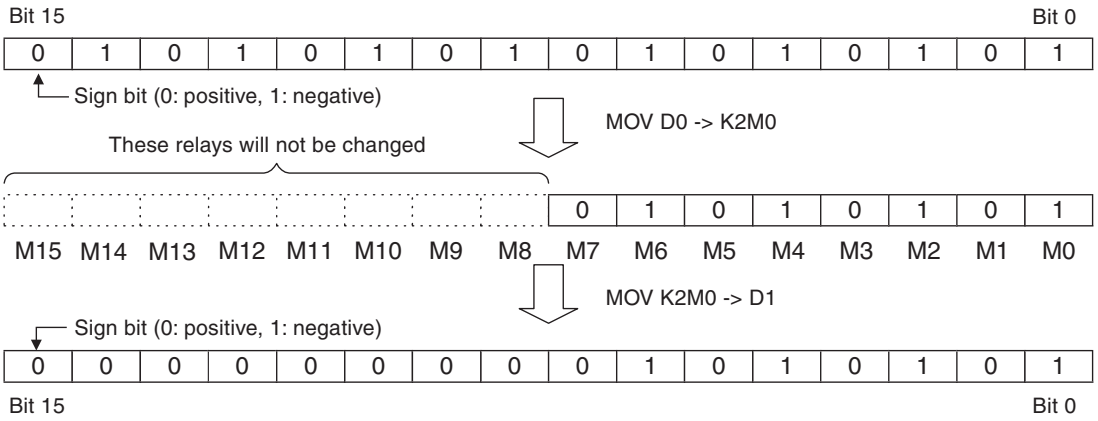
– K8M0: 32 relays, start at M0 (M0 to M31)

Addressing multiple bit devices with a single instruction makes programming quicker and produces more compact programs. The following two examples both transfer the signal states of relays M0 – M3 to outputs Y10 – Y13:



If the destination range is smaller than the source range the excess bits are simply ignored (see the following illustration, top example). If the destination is larger than the source “0” is

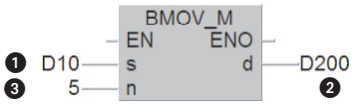
written to the excess devices. Note that when this happens the result is always positive because bit 15 is interpreted as the sign bit (lower example in the following illustration).



6.2.3 Moving blocks of data with the BMOV instruction

The MOV instruction described in section 6.2.1 can only write single 16 or 32 bit values to a destination. If you want, you can program multiple sequences of MOV instructions to move contiguous blocks of data. However, it is more efficient to use the BMOV (Block MOVe) instruction, which is provided specifically for this purpose.

Ladder Diagram



MELSEC Instruction List

```
BMOV   D10    ❶
       D200   ❷
       K5     ❸
```

IEC Instruction List

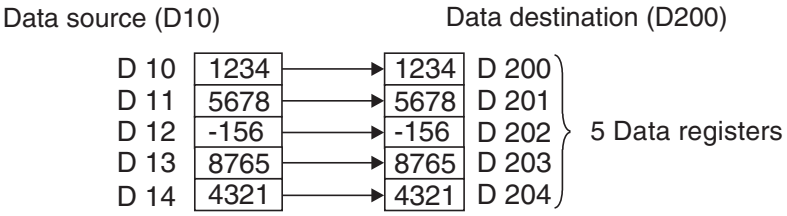
```
BMOV_M   D10, 5, D200
         ❶  ❸   ❷
```

❶ Data source (16-bit device, first device in source range)

❷ Data destination (16-bit device, first device in destination range)

❸ Number of elements to be moved

The example above works as follows:



BMOV also has a pulse-triggered version, BMOVP (see section 6.2.1 for details on pulse-triggered execution).

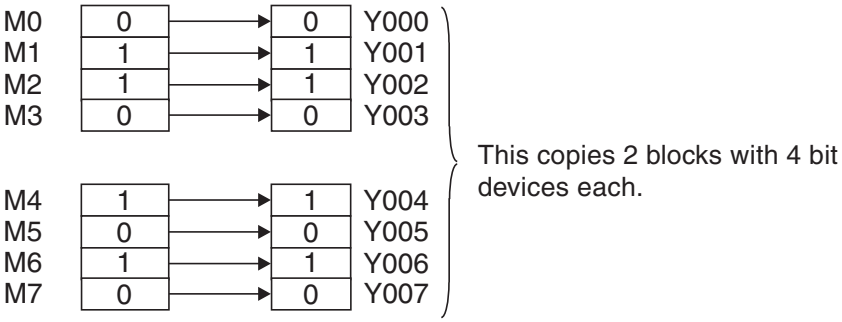
Blocks of bit devices: When you move blocks of bit devices with BMOV the K factors of the data source and the data destination must always be identical.

Example

– Data source: K1M0

– Data destination: K1Y0

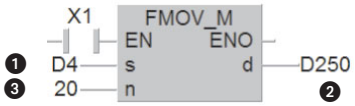
– Number of elements to be moved: 2



6.2.4 Copying source devices to multiple destinations (FMOV)

The FMOV (Fill MOVe) instruction copies the contents of a word device or a constant to multiple consecutive word devices. It is generally used to delete data tables and to set data registered to a predefined starting value.

Ladder Diagram



MELSEC Instruction List

```
FMOV   D4     ❶
       D250   ❷
       K20    ❸
```

IEC Instruction List

```
FMOV_M   D4, 20, D250
         ❶  ❸   ❷
```

❶ Data to be written to the target devices (constants can also be used here)

❷ Data destination (first device of the destination range)

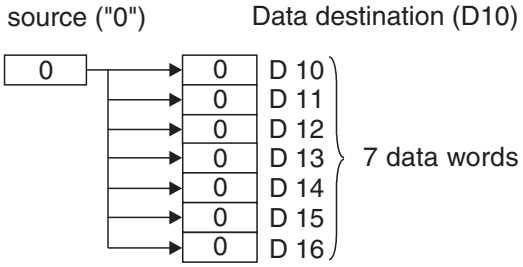
❸ Number of elements to be written in the destination range

The following example writes the value “0” to 7 elements:

– Data source: K0 (constant)

– Data destination: D10

– Number of elements to be written in the destination range: 7



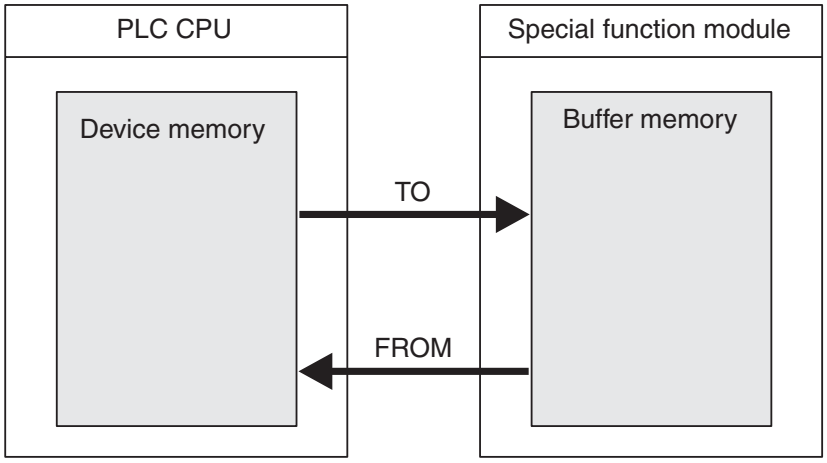
Here too, FMOV has a pulse-triggered version, FMOVP (see section 6.2.1 for details on pulse-triggered execution).

6.2.5 Exchanging data with special function modules

You can supplement the controller's functions by adding so-called "special function modules" – for example for reading analog signals for currents and voltages, for controlling temperatures and for communicating with external equipment.

Each special function module has a memory range assigned as a buffer for temporary storage of data, such as analog measurement values or received data. The PLC CPU can access this buffer and both read the stored values from it and write new values to it, which the module can then process (settings for the module's functions, data for transmission etc).

In addition to the buffer memory special function modules are equipped with digital inputs and outputs. These I/O signals are used e. g. for exchanging status signals between the PLC CPU and the special function module. The digital I/Os of the special function modules do not require special instructions; these inputs and outputs are handled in exactly the same way as those in digital I/O modules. Communication between the PLC CPU and the buffer memory of special function modules however is performed with two special applied instructions: the FROM and TO instructions.



The buffer memory can have up to 32,767 individual addressable memory cells, each of which can store 16 bits of data. The functions of the buffer memory cells depends on the individual special function module – see the module's documentation for details.

Buffer memory address 0
Buffer memory address 1
Buffer memory address 2
:
:
Buffer memory address n-1
Buffer memory address n

The following information is required when you use the FROM and TO instructions:

– The special function module to be read from or written to
– The address of the first buffer memory cell to be read from or written to
– The number of buffer memory cells to be read from or written to
– The location in the PLC CPU where the data from the module is to be stored or containing the data to be written to the module

Special function module address

Since you can attach multiple special function modules to a base unit each module needs to have a unique identifier so that you can address it to transfer data to and from it. This identifier results from the slot where the module is installed respectively from the I/O numbers occupied by the digital inputs and outputs of the special function module (see chapter 3.2.2).

Crucial is the head address of the special function module. For example, if a special function module occupies the range from X/Y010 to Y/X01F the head address is X/Y010. For a FROM or TO instruction however the least significant digit is omitted and the head address in this case reads as "1". When the special function module occupies the range from X/Y040 to Y/X04F, the head address will be "4".

Starting address in the buffer memory

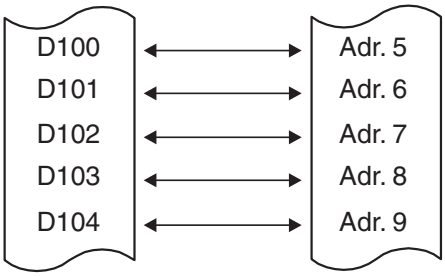
Every single one of the 32,767 buffer addresses can be addressed directly in decimal notation in the range from 0 – 32,767. When you access 32-bit data you need to know that the memory cell with the lower address stores the less significant 16 bits and the cell with the higher address stores the more significant bits.

This means that the starting address for 32-bit data is always the address containing the less significant 16 bits of the double word.

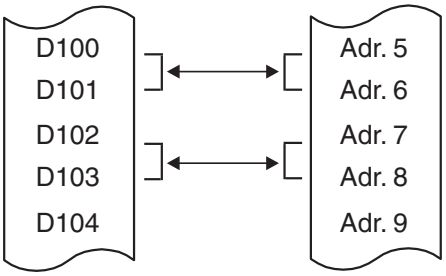
Number of data units to be transferred

The quantity of data is defined by the number of data units to be transferred. When you execute a FROM or TO instruction as a 16-bit instruction this parameter is the number of words to be transferred. In the case of the 32-bit versions DFROM and DTO the parameter specifies the number of double words to be transferred.

16-bit instruction
Units of data: 5



32-bit instruction
Units of data: 2



Data destination or source in the PLC CPU

In most cases you will read data from registers and write it to a special function module, or copy data from the module's buffer to data registers in the PLC CPU. However, you can also use outputs, relays and the current values of timers and counters as data sources and destinations.

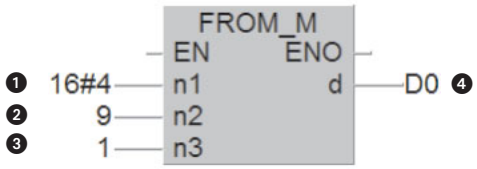
Pulse-triggered execution of the instructions

If you add a P suffix to the instructions the data transfer is initiated by pulse trigger (for details see the description of the MOV instruction in section 6.2.1).

How to use the FROM instruction

The FROM instruction is used to transfer data from the buffer of a special function module to the PLC CPU. Note that this is a copy operation – the contents of the data in the module buffer are not changed.

Ladder Diagram



MELSEC Instruction List

```
FROM    H4  ❶
        K9  ❷
        D0  ❹
        K1  ❸
```

IEC Instruction List

```
FROM_M   16#4, 9 , 1 , D0
         ❶  ❷  ❸  ❹
```

❶ Head address of the special function module on base unit. The address can be entered as decimal or hexadecimal constant (16#).

❷ Starting address in buffer. You can use a constant or a data register containing the value.

❸ Number of data units to be transferred

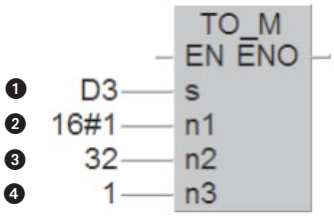
❹ Data destination in the PLC CPU

The example above uses FROM to transfer data from a special function module with the head address X/Y040. The instruction reads the contents of buffer address 9 and writes it to data register D0.

How to use the TO instruction

The TO instruction transfers data from the PLC CPU to the buffer of a special function module. Note that this is a copy operation, it does not change the data in the source location.

Ladder Diagram



MELSEC Instruction List

```
TO      H1  ❷
        K32 ❸
        D3  ❶
        K1  ❹
```

IEC Instruction List

```
FROM_M   D3, 16#1, 32, 1
         ❶  ❷  ❸  ❹
```

❶ Data source in the PLC CPU

❷ Head address of the special function module on base unit. The address can be entered as decimal or hexadecimal constant.

❸ Starting address in buffer

❹ Number of data units to be transferred

In the example above the contents of data register D3 is copied to the buffer address 32 of the special function module with the head address 1 (X/Y010).

Direct access of the buffer memory

The buffer memory of a special function module can also accessed directly, e. g. with a MOV instruction.

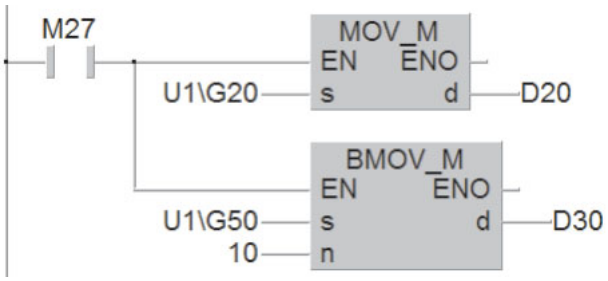
The special function module addressed in this way can be mounted on a base unit or an extension base unit but not in remote I/O stations.

Format of the device address:

For example the device address U3\G11designates the buffer memory address 11 in the special function module with the head address 3 (X/Y30 to X/Y3F).

When relay M27 is set in the following example, the contents of the buffer memory address 20 of the special function module with the head address 1 is copied to data register D20. Afterwards the contents of the buffer memory addresses 50 to 59 is copied into the data registers D30 to D39.

Ladder Diagram



MELSEC Instruction List

```
LD      M27
MOV     U1\G20
        D20
MOV     U1\G50
        D30
        K10
```

IEC Instruction List

```
LD      M27
MOV_M   U1\G20,  D20
BMOV_M  U1\G50,  10, D30
```

Automatic data transfer between PLC CPU and special function modules

Several add-in tools for the GX IEC Developer to set the initial data and condition data for special function modules are available. These tools simplify the configuration of special function modules and facilitate the automatic data transfer between the PLC CPU and special function modules. These optional software are commonly called GX Configurator. An extension to this name denotes the affinity to certain special function modules.

In the software ***GX Configurator-AD*** for example all settings for analog input modules can be made. To do this it is not necessary for the user to know the structure of the buffer memory. The special function module parameters are downloaded in the PLC once together with the sequence program. There is no need to transfer these settings in the sequence program. Thus the size of the program is reduced and error sources are eliminated.

In GX Configurator-AD can also be specified where in the PLC CPU for instance measured values should be stored. Afterwards this data transfer is processed automatically. FROM-/TO instructions or the above described direct access to the buffer memory are not required.

6.3 Compare Instructions

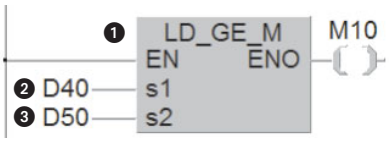
Checking the status of bit devices like inputs and relays can be achieved with basic logic instructions because these devices can only have two states, “0” and “1”. However, you will also often want to check the contents of word devices before doing something – for example switching on a cooling fan when a specified setpoint temperature is exceeded.

To achieve this an output instruction or an logical operation can be made conditional on the result of a compare instruction. In addition to the compare instructions described here the PLC CPUs of the MELSEC System Q can compare also floating decimal point values, binary block data and character strings.

For comparisons MELSEC and IEC instructions are available.

Comparison at the beginning of a logic operation

Ladder Diagram



MELSEC Instruction List

```
❶ LD>=      D40 ❷
            D50 ❸
  OUT       M10
```

IEC Instruction List

```
  LD        TRUE
❶ LD_GE_M   D40, D50
              ❷   ❸
  ST        M10
```

This instruction is equivalent to the "wiring" of the EN input in the Ladder Diagram. "TRUE" means that the input condition is always fulfiled.

❶ Compare condition

❷ First value to be compared

❸ Second value to be compared

If the condition evaluates true the signal state after the comparison is set to “1”. A signal state of “0” shows that the comparison evaluated as false. In the above example the relay M10 is set when the contents of data register D40 is greater than or equal to the contents of data register D50.

The following comparisons are possible:

– Compare for "equals": **=** (value 1 = value 2)
IEC instruction: EQ (*Equal*)

The output of the instruction is only set to "1" if the values of both devices are identical.

– Compare for "greater than": **>** (value 1 > value 2)
IEC instruction: GT (*Greater Than*)

The output of the instruction is only set to "1" if the first value is greater than the second value.

– Compare for "less than": **<** (value 1 < value 2)
IEC instruction: LE (*Less Than*)

The output of the instruction is only set to "1" if the first value is smaller than the second value.

– Compare for "not equal": **<>** (value 1 <> value 2)
IEC instruction: NE (*Not Equal*)

The output of the instruction is only set to "1" if the two values are not equal.

– Compare for "less than or equal to": **<=** (value 1 ≤ value 2)
IEC instruction: LE (*Less Equal*)

The output of the instruction is only set to "1" if the first value is less than or equal to the second value.

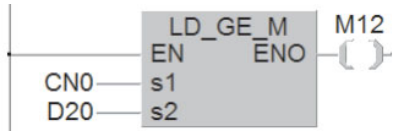
– Compare for "greater than or equal to": **>=** (value 1 ≥ value 2)
IEC instruction: GE (*Greater Equal*)

The output of the instruction is only set to "1" if the first value is greater than or equal to the second value.

To compare 32-bit data prefix a D (for double word) to the compare condition. (For example LD**D**_EQ-M or LD**D**_GE_M)

More examples:

Ladder Diagram



MELSEC Instruction List

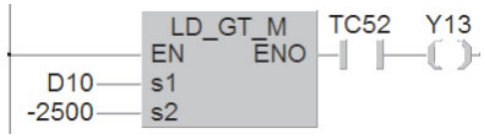
LD>=	C0
	D50
OUT	M12

IEC Instruction List

LD	TRUE
LD_GE_M	CN0, D20
ST	M12

Relay M12 is set to "1" when the value of counter C0 is equal to or greater than the contents of D20.

Ladder Diagram



MELSEC Instruction List

LD>	D10
	K-2500
AND	T52
OUT	Y13

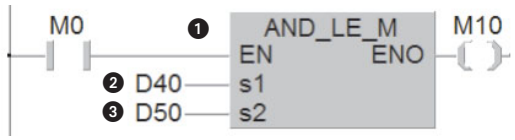
IEC Instruction List

LD	TRUE
LD_GT_M	D10, -2500
AND	TC52
ST	Y13

Output Y13 is switched on when the contents of D10 is greater than -2,500 and timer T52 has finished running.

Compare as a logical AND operation

Ladder Diagram



MELSEC Instruction List

	LD	M0
❶	AND<=	D40 ❷
		D50 ❸
	OUT	M10

IEC Instruction List

	LD	M0
❶	AND_GE_M	D40, D50
		❷ ❸
	ST	M10

❶ Compare condition

❷ First comparison value

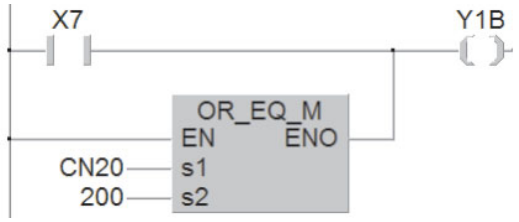
❸ Second comparison value

An AND comparison can be used just like a normal AND instruction (see chapter 4).

The comparison options are the same as those described above for a comparison at the beginning of an operation. In the example shown above the relay M10 is set when the relay M0 is "1" **and** the contents of data register D40 is equal to or smaller than the contents of data register D50.

Compare as a logical OR operation

Ladder Diagram



MELSEC Instruction List

	LD	X7
❶	OR=	C20 ❷
		K200 ❸
	OUT	Y1B

IEC Instruction List

	LD	X7
❶	OR_EQ_M	CN20, 200
		❷ ❸
	ST	Y1B

❶ Compare condition

❷ First comparison value

❸ Second comparison value

An OR comparison can be used just like a normal OR instruction (see chapter 4). In this example the output Y1B is set when the input X7 is on **or** the Counter C20 has reached the actual value "200".

6.4 Math Instructions

All the controllers of the MELSEC System Q can perform all four basic arithmetical operations and can add, subtract, multiply and divide. MELSEC instructions are available for math operations with binary values, binary block data, BCD values and character strings.

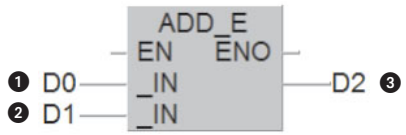
When you edit your program with GX IEC Developer in Ladder Diagram or IEC instruction list additional IEC instructions can be used. In this chapter only these IEC instructions are described. MELSEC instructions are covered in detail in the Programming Manual for the A/Q series and the MELSEC System Q, art. no. 87431.

The IEC instructions for addition, subtraction, multiplication und division can be applied for the data types INT (16-bit integer), DINT (32-bit integer) or REAL (floating decimal point values). Please note that DINT and REAL devices can not assigned directly to these instructions and must be defined as variables before (see chapter 4.6.2).

6.4.1 Addition

The ADD instruction calculates the sum of two values and writes the result to another device.

Ladder Diagram



IEC Instruction List

```
LD    D0  ❶
ADD   D1  ❷
ST    D2  ❸
```

❶ First source device or constant

❷ Second source device or constant

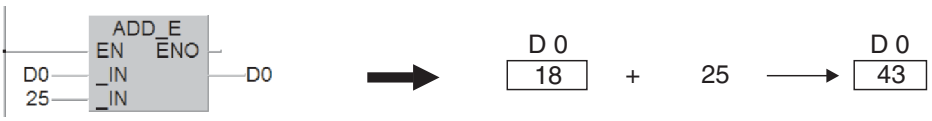
❸ Device in which the result of the addition is stored

The example above adds the contents of D0 and D1 and writes the result to D2.

Examples

Add 1,000 to the contents of data register D100:

If you want you can also write the result to one of the source devices. However, if you do this remember that the result will then change in every program cycle if the ADD instruction is executed cyclically!

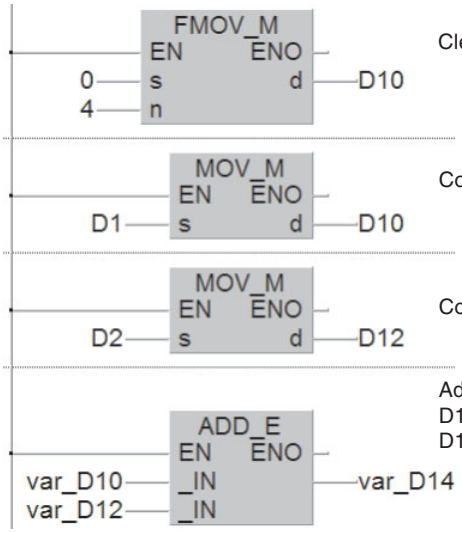


The signs of the values are taken into account by the ADD instruction (e. g. 10 + (-5) = 5).

The data types of the input and output variable of the ADD instruction must be identical. This can cause problems when the result of the addition exceeds the value range of the variables. For example, when you add the two integer values (16 bit) "32700" and "100" the result stored is not "32800" as expected but "-32736" since the maximum value of a 16 bit variable is "32767". An overflow is interpreted as negative value and will lead to a wrong result.

One possible solution is to copy the values to be added into 32-bit variables before the addition. The addition is then performed with these 32-bit variables.

Ladder Diagram



Clear D10 to D13

Copy contents of D1 to D10

Copy contents of D2 to D12

Add contents of D11/D10 and D13/D12, store result to D15/D14

IEC Instruction List

```
LD      TRUE
FMOV_M  0, 4, D10

LD      TRUE
MOV_M   D1, D10

LD      TRUE
MOV_M   D2, D12

LD      var_D10
ADD     var_D12
ST      var_D14
```

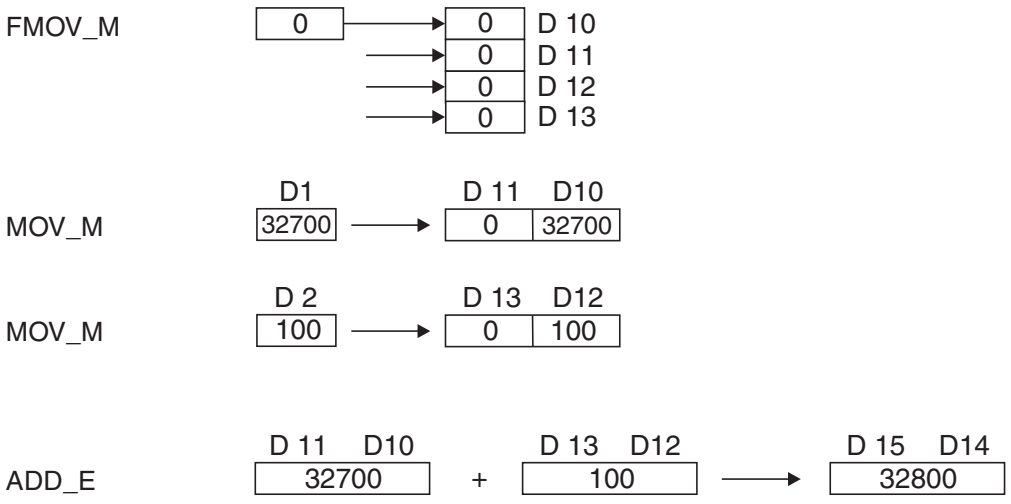
Since it is not possible to define 32-bit devices directly as input and output variables of an ADD instruction a declaration as Global Variables is necessary:

Global Variable List

	Class	Identifier	MIT-Addr.	IEC-Addr.	Type
0	VAR_GLOBAL	var_D10	D10	%MD0.10	DINT
1	VAR_GLOBAL	var_D12	D12	%MD0.12	DINT
2	VAR_GLOBAL	var_D14	D14	%MD0.14	DINT

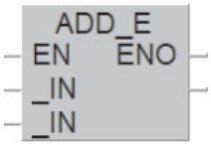
The names for the variables (*Identifiers*) can be chosen freely. For a better understanding the device addresses are used in this example.

With the values given above the contents of the data registers are changed during execution of the instruction as follows:

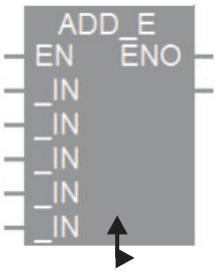


Double word register D14 contains the correct result of the addition.

An ADD instruction is not restricted to two input variables. Up to 28 input variables can be defined. In the Ladder Diagram this can be done as follows:



Select the ADD-E instruction from the ***Function Block Selection*** window (see chapter 4.7.7) and place the instruction into the body of the POU.

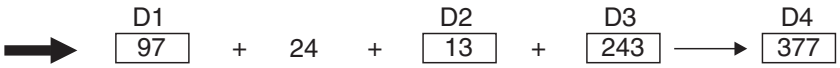


Click on the instruction. This changes the colour of the box. Move the cursor downward until he converts into a double arrow.

Then click the left mouse button and “Click – Drag” downward until the desired number of input variables is displayed. Release the left mouse button at this point

When programming in IEC instruction list just enter the ADD instruction several times. For example:

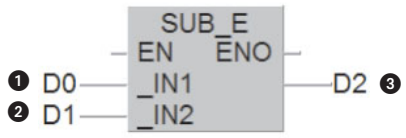
```
LD      D1
ADD     24
ADD     D2
ADD     D3
ST      D4
```



6.4.2 Subtraction

The SUB instruction calculates the difference between two numerical values (contents of 16-bit or 32-bit devices or constants). The result of the subtraction is written to a third device.

Ladder Diagram



IEC Instruction List

```
LD    D0  ❶
SUB   D1  ❷
ST    D2  ❸
```

❶ Minuend (the subtrahend is subtracted from this value)

❷ Subtrahend (this value is subtracted from the minuend)

❸ Difference (result of the subtraction)

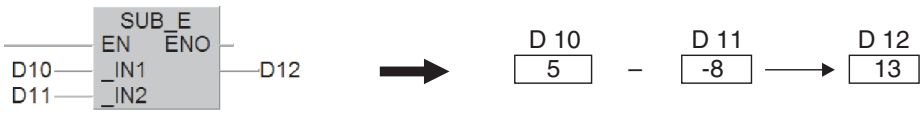
The data types of the input and output variable of the SUB instruction must be identical.

In the example above the contents of D1 is subtracted from the contents of D0 and the difference is written to D2.

Examples

Subtract 100 from the contents of data register D100 and write the result to D101 when relay M37 is set :

The signs of the values are taken into account by the SUB instruction:

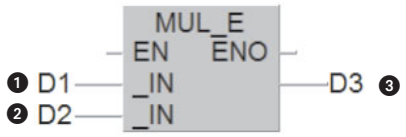


If you want you can also write the result to one of the source devices. However, if you do this remember that the result will then change in every program cycle if the SUB instruction is executed cyclically!

6.4.3 Multiplication

The "MUL" instruction multiplies two 16-bit or 32-bit values and writes the result to a third device.

Ladder Diagram



IEC Instruction List

```
LD    D1  ❶
MUL   D2  ❷
ST    D3  ❸
```

❶ Multiplicand

❷ Multiplier

❸ Product (result of the multiplication, multiplicand x multiplier = product)

The example above multiplies the contents of D1 and D2 and writes the result to D3.

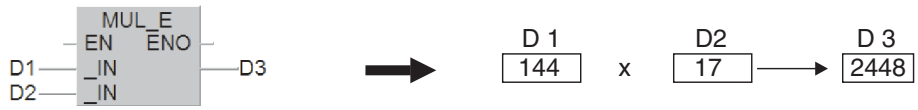
NOTE

The data types of the input and output variable of the MUL instruction must be identical. When the result of the multiplication exceeds the value range for 16-bit or 32-bit variables the most significant bits get lost and the product is not correct. When 16-bit values are to be multiplied they can be copied into 32-bit variables as described for the ADD instruction (see chapter 6.4.1). The MUL instruction is then performed with these 32-bit variables and the result will be correct.

For a MUL instruction up to 28 input variables can be defined. The setting is similar to the ADD instruction (see chapter 6.4.1).

Examples

Multiply the contents of D1 and D2 and store the product in D3:

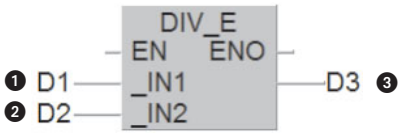


The signs of the values are taken into account by the MUL instruction. In this example the value in D10 is multiplied by the constant value -5:

6.4.4 Division

The DIV instruction divides one number by another

Ladder Diagram



IEC Instruction List

```
LD   D1 ❶
DIV  D2 ❷
ST   D3 ❸
```

❶ Dividend

❷ Divisor

❸ Quotient (result of the division, dividend ÷ divisor = quotient)

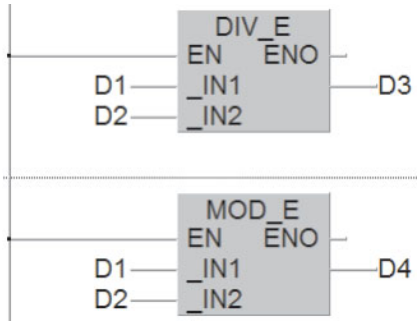
In the above example the contents of data register D1 is divided by the contents of D2. The result is stored in D3.

NOTE

The divisor should never be 0. Division by 0 is not possible and will generate an error which stops the PLC CPU. (This case is possible when, as shown in the above example, the division is performed with the contents of data registers and these registers have been cleared with a RESET operation. To avoid a stop of the PLC the data register containing the divísor should be set to an defined value by the PLC program **before** execution of the DIV instruction.)

The data types of the input and output variable of the DIV instruction must be identical. When integer values (INT or DINT) are devided, the quotient will be also an integer value. To determine the remainder of the calculation the MOD instruction can be used.

Ladder Diagram

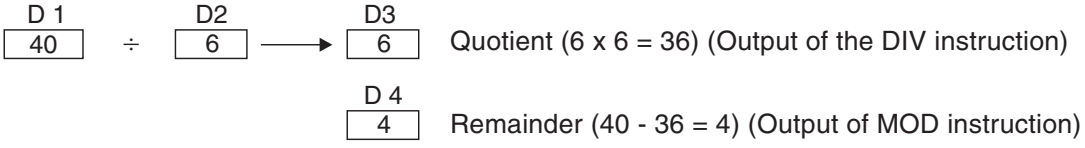


IEC Instruction List

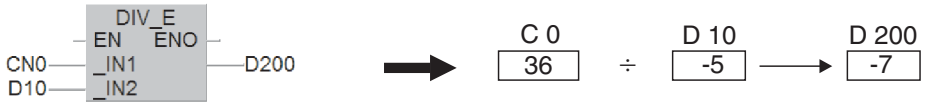
```
LD   D1
DIV  D2
ST   D3

LD   D1
MOD  D2
ST   D4
```

The input variables of the MOD instruction are the same as for the DIV instruction. In the above example the contents of data register D1 is divided by the contents of D2. The quotient is stored in D3 and the remainder in D4:



The signs of the values are taken into account by the DIV instruction. In this example the counter value of C0 is divided by the value in D10:

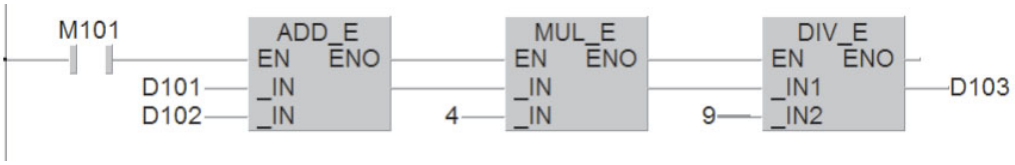


6.4.5 Combining math instructions

In real life one calculation is seldom all you want to perform. To solve more complex calculations math instructions can be combined easily.

The following example shows how you could calculate the sum of the values in data registers D101 and D102, then multiply the result by the factor 4 and finally divide the product by 9. The result of this calculation is stored in data register D103.

Ladder Diagram



Index

G

H

I

L

M

N

O

P

Q

R

S

T

V

W

MITSUBISHI ELECTRIC INDUSTRIAL AUTOMATION

Gothaer Straße 8
D-40880 Ratingen

Telefon: 02102 486-0
Hotline: 01805 000-7650

Fax: 02102 486-7170
megfa-mail@meg.mee.com

www.mitsubishi-automation.de
www.mitsubishi-automation.com