Ωmega
Short System Description

E	5.94013.03

Short System Description
Ωmega

Edition: August 1996
E 5.94013.03

Within the scope of further development of our products, Baumüller Nürnberg GmbH reserves the right to change their technical data and handling.

Manufacturer and Supplier´s Adress:

Baumüller Nürnberg GmbH
Ostendstraße 80
90482 Nürnberg
Germany
Tel. ++49(0)911/5432-0
Fax ++49(0)911/5432-130

Country of Origin: Germany

Contents

1 GENERAL

Ωmega: Synthesizing Open-Loop Control Technology, Rapid-Configuration Closed-Loop Technology and Integrated Drive Technology

The current structural changes in automation technology from central systems to distributed units are leading to an ongoing migration of functions, which used to be the exclusive preserve of central controllers, to new distributed peripherals.

The ideal complete solution to all automation requirements would be a consistent modular hardware and software system that can be optimally tailored both from a technical and commercial point of view to the specific range of functions of the distributed unit. This results in the following advantages:

- ❏ A single supplier guarantees responsibility for the entire system.
- ❏ A single supplier guarantees the availability of spare parts.
- ❏ A single supplier guarantees staff training.
- ❏ Optimization to the minimum functionality that is needed.
- ❏ Optimization to as few, rapid transfer paths as possible.
- ❏ Optimum distribution of tasks to the central and distributed systems

- ❏ A single programming environment for all the system components results in
  - One-off, easy training
  - Easy program generation
  - Rapid commissioning
  - Rapid service and
  - Easy maintenance
- ❏ Data transfer to the distributed system unit is provided to the user due to:
  - Short response times by parallel transfer and
  - Simple programming
- ❏ Best cost/benefit ratio with the hardware and software.

The Ωmega configurable automation system is just such an open- and closed-loop control system. The system is of compact modular structure using high-quality peripheral and communication modules which provides functionality in addition to open-loop control, such as configurable closed loop control technology, integrated drive technology, visualization and communications. This makes Ωmega the perfect cutting-edge system for all automation technology applications both today and in the future.

PROPROG is the system's convenient programming environment on IBM-compatible PCs for the graphical and text-based programming operations that you are familiar with from the PLC world.

- Sequential function chart, SFC
- Function plan, FUP,
- Ladder diagram, LD and
- Instruction list, IL.

Programming according to IEC-1131-3 is in modular sections.

In addition, PROPROG integrates

- the C high-level language and
- Assembler

with an appropriate powerful debugger.

For commissioning and servicing, the system has a much wider range of status displays and debugging functions than are normally available:

- Unconditional breakpoints,
- Conditional breakpoints,
- Trigger conditions,
- Overwrite,
- Single step,
- Single cycle,
- Variable status (cycle-consistent),
- Address status with flow control,
- Online lists,
- Condition lists

Uncomplicated Closed-Loop Control

The supplied library provides a wide range of function elements that you can use for open- and closed-loop programming in all programming languages. The optimized program code for individual types of applications, e.g. assembly language for fast closed-loop control technology, was archived such that the user does not come into contact with it.

Hardware interrupts meet the demand for sampling at equidistant time intervals. This makes possible a symbiosis in the same system of processing operations that are different with respect to time. System interrupts trigger closed-loop functions and this results in immediate interruption of the cyclical open-loop control program. The constant, configurable sampling intervals that are needed for digital open-loop control technology are reflected at programming in the parameterization of a library element and the definition of the program section as an interrupt closed-loop control program. In addition to these programs that are called cyclically, it is just as easy to implement fast, event-driven programs.

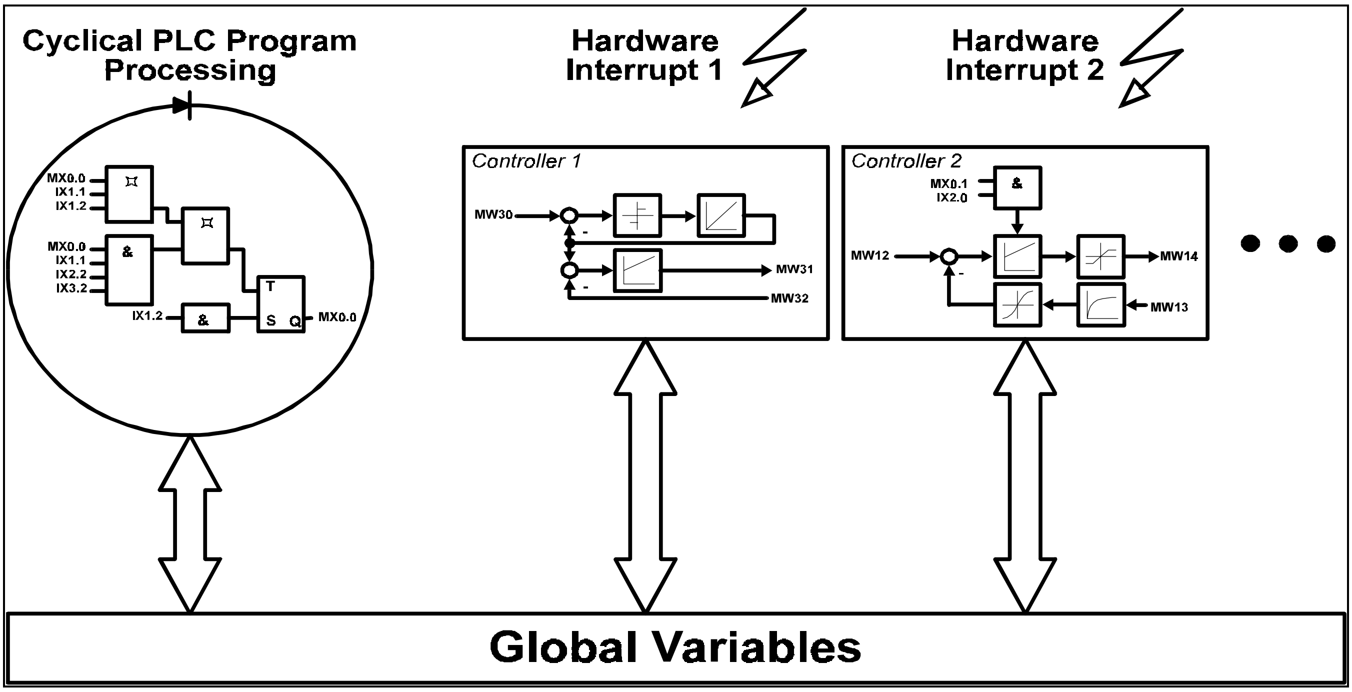


Figure 1.1: Cyclical and Controller Program Sections

Cyclical PLC programs, event-driven programs and controller functions are available in-parallel in the system on one or more CPUs and they can use common variables. This makes possible rapid data transfer (by means of internal RAM or dual-port RAM) between the cyclical open-loop control program and the closed-loop control programs.

To allow (PLC) program developers to do without special knowledge of assembly language and the system, the PROPROG development environment provides appropriate aids for configuring open-loop control functions and exploiting to the full all the necessary system components.

The link element library provides a large number of function blocks that you can freely and permanently define. The link elements are listed sorted in the following function groups:

- Closed-loop control,
- Bulk storage,
- Memory,
- Data conversion,
- Arithmetic,
- Logic,
- Counter,
- Timer,
- Comparator,
- Jumps,
- Modules,
- etc.

Users can define their own link elements or replace existing ones, which can then be used in all the programming languages implemented in PROPROG.

PI-REG_D PI-Regler, Doppelwort REGELUNG

Universeller PI-Regler, verwendbar als Drehzahlregler oder überlagerter Regler. Geeignet für dynamische Ablöseregelungen.

Flexible Integratorfunktionen:

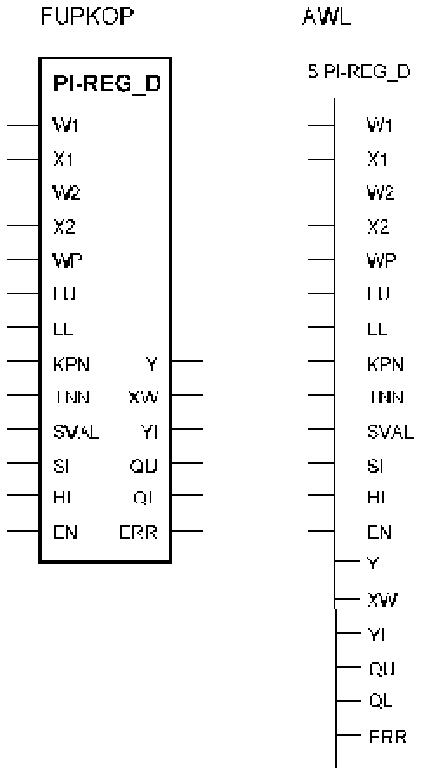
- Anfangswert setzen ⇒ SVAL in Integrator laden
- Momentanwert des Integrators festhalten ⇒ P-Regler
- Führung des Integrators durch SVAL
- Führung des Integrators durch Begrenzung des Reglers

Regler-Gesamtfunktionen:
Unabhängige Einstellung sowie Veränderung folgender Größen während des Betriebs:

- Regelverstärkung KPN
- Normierte Nachstellzeit TNN
- Reglerbegrenzungen LU und LL
- Vorsteuerwert WP, z.B. für Beschleunigungsbeaufschlagung

Zweiter Istwerteingang X2, z.B. für Statikaufschaltung

Anzeige bei Erreichen der eingestellten Grenzen.



Aufgerufene Firmwarebausteine: FB 100

Parameter

Eingänge:

Parameter-name	Beschreibung	Plaus.-Gruppe	Pflicht-Belegung	Inver-tierung	Anzeige	Dopplungs-art	Dopplungs-Nummer
W1	Sollwert 1	DOPPELW	ja	nein	ja	0	00
X1	Istwert 1	DOPPELW	ja	nein	ja	0	00
W2	Sollwert 2	DOPPELW	nein	nein	ja	0	00
X2	Istwert 2	DOPPELW	nein	nein	ja	0	00
WP	Vorsteuerwert	DOPPELW	nein	nein	ja	0	00
LU	oberer Grenzwert	DOPPELW	nein	nein	ja	0	00
LL	unterer Grenzwert	DOPPELW	nein	nein	ja	0	00
KPN	norm. prop. Verstärkungsfaktor	WORT	ja	nein	ja	0	00
TNN	norm. reziproge Nachstellzeit	WORT	ja	nein	ja	0	00
SVAL	Integrator-Setzwert	DOPPELW	nein	nein	ja	0	00
SI	Integrator setzen	BIT	nein	ja	ja	0	00
HI	Integrator halten	BIT	nein	ja	ja	0	00
EN	Reglerfreigabe	BIT	ja	ja	ja	0	00

Figure 1.2: Extract from the PI Controller Documentation

You can integrate link elements as often as you like in cyclical and interrupt program sections – as macros – as you choose. In this connection, the linking, "wiring" and parameterization of the link elements can be done in FUP, LD and IL. While the program is running, it is possible to load or exchange or compile and test link elements.

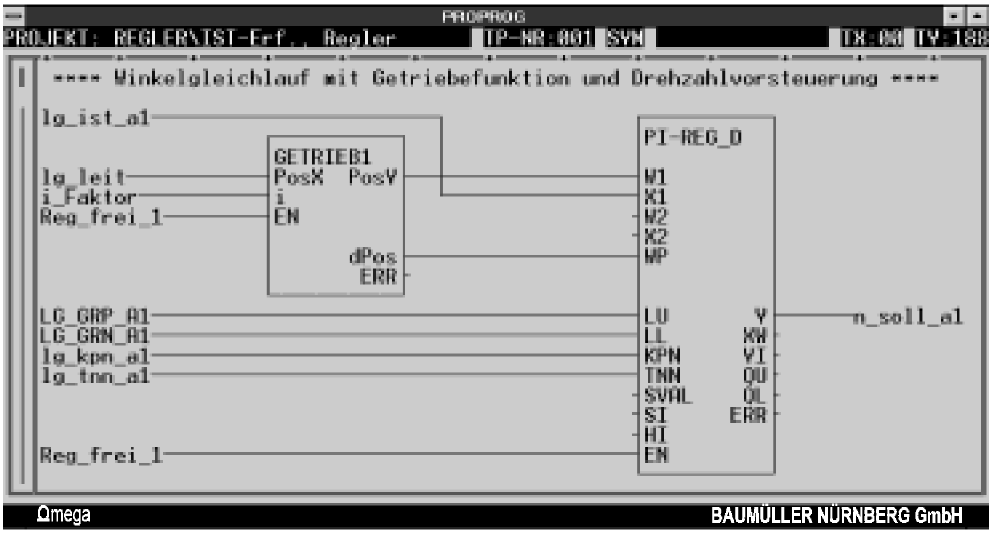


Figure 1.3: Sample program: Accurate synchronism with gear function and speed control

Other aids, e.g. ONLINE list or ONLINE display make it possible to monitor and dynamically display any operands and parameters of the system.

Digital closed-loop control technology: analog-optimized

There is an optimization kit for matching control loops in the classical analog way. It is possible to use potentiometers to set up to 32 controller program-internal parameters (k_p, t_n, etc.) in real time. In addition up to 16 oscilloscope connections are available for evaluating highly dynamic internal controller variables (specified and actual values, controller deviation, etc.). Input values are accepted directly, internal variables are updated at the time of sampling and output variables are routed directly to the output channels as they occur. When optimization has been carried out, the kit is removed.

The ONLINE functions of the PROPROG programming system represent a very efficient tool for monitoring and manipulating static bit, byte, word, doubleword and floating point variables.

System characteristic values such as the operating status and processor loading due to cyclical and interrupt programs are evaluated ONLINE.

Programs are stored in (buffered) RAM and you can change them ONLINE. After successfully testing the program, you can store it in a (non-volatile) area of memory of the deletable flash EPROM and run it there too. This makes it possible for you to quickly and cheaply match program changes to system changes or extensions.

The Ωmega open-loop control system's extremely modular structure allows users to physically and functionally influence their projects in an optimum way at all stages of development (hardware configuration, ..., commissioning and maintenance) as required.

The components that the open- and closed-loop control system can be composed of are assigned to several different functional levels:

- ❏ CPU Master Modules (BCCOM-BUS), these are the central processor units and the integrated industrial PC
- ❏ Digital/analog peripheral modules (B-BUS) such as I/O modules, counter modules, absolute value encoders, evaluation modules
- ❏ System bus modules with slave processors (SYS-BUS), these include multifunction modules, serial interface modules, Interbus-S modules, as well as the signal processor module for integrated drive technology of the highest quality.

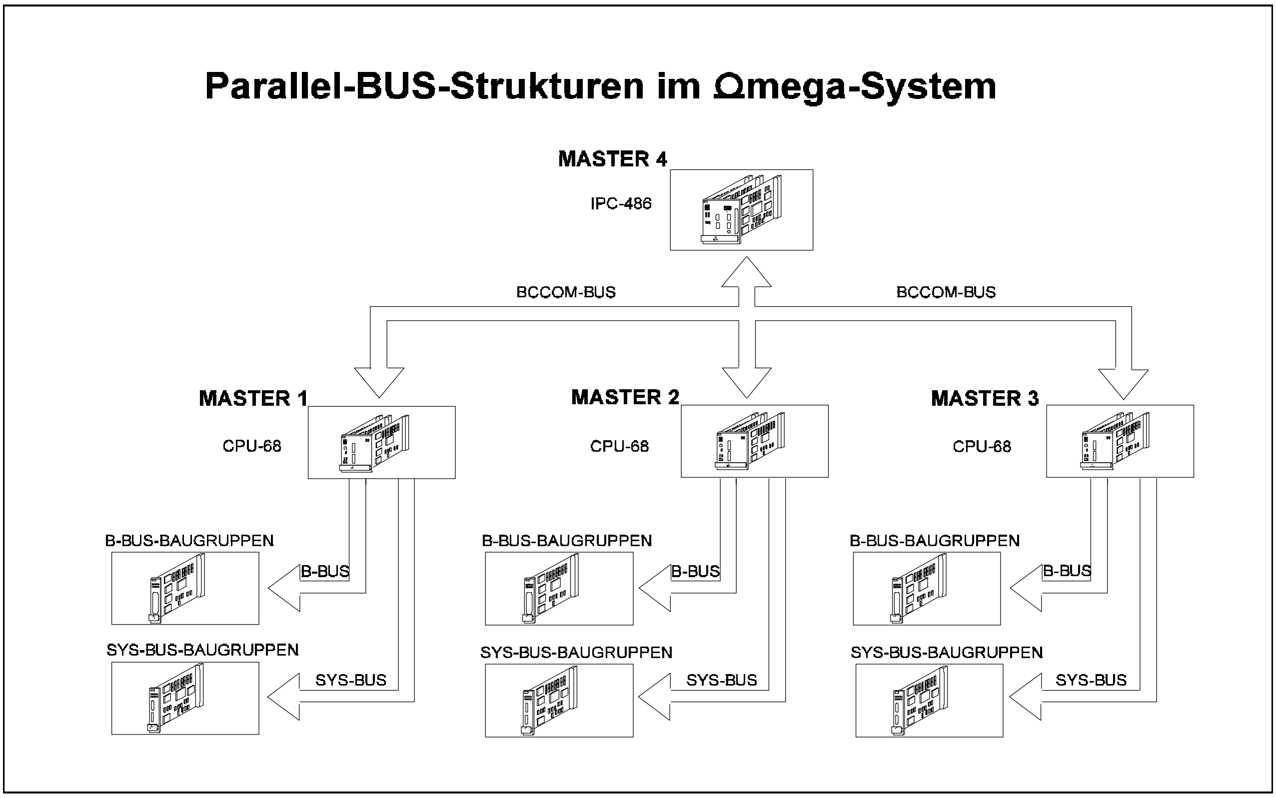


Figure 1.4: Parallel Bus Systems

This makes it possible to implement small distributed units with an I/O level (B-BUS modules only) and efficient closed-loop technology as well as powerful multiprocessor systems with 'intelligent' processor modules for communications, visualization and direct drive technology.

The parallel internal bus systems are optimized with regard to the requirements of closed-loop technology:

- Due to its sampling cycle, which you can program independently of the program run time, the B-BUS can be used to synchronize the control loops. In this connection, synchronization can be carried out to the inputs or to the outputs.

- The SYS-BUS parallel master-slave bus system makes possible high-performance systems by means of distributed intelligence with powerful processor modules that unload communications, and integrated drive and closed-loop technology functions from the master CPU.

- In a multiprocessor multimaster system with two or more CPU modules, it is possible to synchronize the central CPUs via the BCCOM-BUS such that specified value settings, for example, are calculated in one CPU and provided to the others by direct memory accesses to dual-port RAMs before the other CPUs' controller calculations are triggered. This ensures that all the CPUs in the controller program have values available that are time-consistent. In addition, the BCCOM-BUS makes it easier to structure redundant systems.

For every bus system, we have developed favourably priced, efficient, specific modules with optimized functionality.

To prevent disturbances in distributed systems, all the input/output interfaces are optically and galvanically isolated to the outside.

Due to the modular structure and ongoing development of additional modules, the system is open and can be easily adapted to future requirements at any time.

There are tailor-made, tested link elements for all the hardware components and modules that can be used in the system. These elements are all fully documented.

Table: Modules of the Ωmega Rack Line

		Brief Description
Digital I/O Modules	DIO-32A	32 digital outputs, optically isolated, 24-V industrial logic, 0.5 A, short-circuit-proof, B-BUS
	DIO-32E	32 digital inputs, optically isolated, 24-V industrial logic, B-BUS
Analog I/O Modules	AIO-16A	16 output channels, 12-bit, 0..10V, ±10V, 5 mA, optically isolated, B-BUS
	AIO-32E	32 single-ended/16 differential input channels, 12-bit, optically isolated, conversion time 10 µs, B-BUS
Counter modules	IZB-04	4 single-ended/differential counter channels, optically isolated, 4 MHz, direction detection, B-BUS
Multifunction module	MIO-16	16 input channels, optically isolated, 24-V industrial logic, interrupt-capable, edge sensitive, SYS-BUS
Serial port modules	SIO-02	2 RS 232/485 serial ports, optically isolated, 50 bps - 38.4 kbps, SYS-BUS
	SIO-06	6 RS 232/485 serial ports, optically isolated, 50 bps - 38.4 kbps, SYS-BUS
Comms. modules	INT-M-01	Interbus-S master controller, remote bus interface, SYS-BUS
	INT-S-01	Interbus-S slave controller, local bus interface, CPU adapter
	INT-S-02	Interbus-S slave controller, local bus interface, SYS-BUS
	CAN-01	1 CAN interface, Basic CAN Controller, ISO 11898, CPU adapter
	CAN-02	2 CAN interfaces, 2 Basic CAN Controllers, ISO 11898, CPU adapter
	CAN-03	2 CAN interfaces, 2 Basic CAN Controllers, bidirectional CAN Bridge, ISO 11898, CPU adapter
Integrated drive technology	DSP-C30	Digital signal processor, FPU, local bus interface, SYS-BUS
	APM-02	Axial peripheral module for 2 axes, local bus interface
	ANO-04	Analog output module, 4 output channels, local bus interface
Integr. industrial PC	IPC-486	IBM-compatible, 8 MB of RAM, SVGA, IDE controller, 2 ser./1 par., B-BUS, SYS-BUS, BCCOM-BUS
Central processor units	CPU-68-001	256 kB SRAM, 256 flash EPROM, 2 RS 232, B-BUS
	CPU-68-002	256 kB SRAM, 256 flash EPROM, 2 RS 232, optically isolated, B-BUS, SYS-BUS
	CPU-68-003	256 kB SRAM, 256 flash EPROM, 2 RS 232, optically isolated, RTC, FPU, B-BUS, SYS-BUS
	CPU-68-004	256 kB SRAM, 256 flash EPROM, 2 RS 232, optically isolated, B-BUS, SYS-BUS, BCCOM-BUS
	CPU-68-005	256 kB SRAM, 256 flash EPROM, 2 RS 232, optically isolated, RTC, FPU, B-BUS, SYS-BUS, BCCOM-BUS
	CPU-68-006	256 kB SRAM, 256 flash EPROM, 2 RS 232, optically isolated, RTC, FPU, SCSI bus, B-BUS, SYS-BUS, BCCOM
Power supply modules	PSB-01	Input voltage 230V AC, output 5V, 10A
	PSB-02	Input voltage 230V AC, output 5V, 10A and +12V, 2A and -12V, 0.5A
	PSB-03	Input voltage 24V DC, output 5V, 10A
	PSB-04	Input voltage 24V DC, output 5V, 10A and +12V, 2A and -12V, 0.5A
	PSB-05	Input voltage 230V AC, output 5V, 20A
	PSB-06	Input voltage 230V AC, output 5V, 30A

The Ωmega open-loop control system is a flexible multiprocessor system with a very wide range of functionality that covers all performance classes. It makes possible the implementation of a programmable logic controller in conjunction with configurable closed-loop control technology, integrated driver technology, an integrated industrial PC as well as very powerful communications based on open serial ports.

2 HARDWARE OVERVIEW

The configurable Ωmega open-loop control system is of modular structure which goes a long way towards meeting the requirements of modern and future-oriented automation technology. The system also provides additional integrated functionality, e.g. configurable closed-loop control technology, drive technology, visualization and communications.

Due to Ωmega's modular structure and the wide range of high-quality peripheral and communications modules, the system can be expanded appropriately to deal with any applications. The easy system handling and configuration save you time and money.

Ωmega has a convenient programming interface on IBM-compatible PCs for the following PLC languages:

- ❑ Sequential function chart, SFC,
- ❑ Function plan, FUP,
- ❑ Ladder diagram, LD and
- ❑ Instruction list, IL

that includes a wide range of status display and debugging functions. As an option, you can carry out programming on the interface in high level languages or assembly language with the corresponding debugging functions.

Ωmega systems are designed for use in harsh industrial environments and are particularly rugged and resistant to electromagnetic disturbances. Comprehensive diagnostic functions make commissioning and trouble shooting in the plant easier.

Ωmega is based on a Motorola processor and has three parallel BUS structures for different ranges of requirements

- ❑ I/O peripherals,
- ❑ intelligent peripherals and
- ❑ multiprocessor system.

In addition you can connect communications modules as CPU extensions.

CPUs with different levels of expansion and various interface modules make possible optimum adaptation to the open- or closed-loop task in each case.

The simplest Ωmega systems comprise a module rack with a backplane, a power supply module for internal supply, a central processor unit with program memory and several I/O modules.

To expand the central system, appropriate module racks, backplanes and peripheral modules are available.

Distributed structuring is made easier by distributed I/O modules.

High-performance serial bus systems connect physically separated stations:

- ❏ CAN bus as a low-cost network for object-oriented distributed process communication at up to 1 Mbps
- ❏ Interbus-S for master/slave process communication at up to 500 kbps

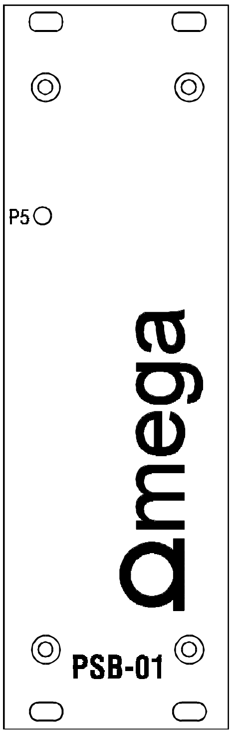
Advantages:

- ❏ A flexible, modular system in the medium- to high-performance class with closed-loop technology that can be configured more quickly and integrated drive technology
- ❏ Basic module racks measuring 42, 63 and 84 TE (units of depth)
- ❏ Central processing units with graduated performance and functionality
- ❏ Easy system expansion
- ❏ A wide range of subassemblies and modules are available some of which are interrupt-capable
- ❏ Configuration and programming carried out using an IBM-compatible PC
- ❏ Library of closed-loop control elements
- ❏ The wide range diagnostics functions considerably reduces the amount of time needed for commissioning
- ❏ Small dimensions
- ❏ Reliable and economic due to the high integration density

System Components:

- ❏ Module rack with backplane
- ❏ Power supply module
- ❏ Central processor units
- ❏ Integrated industrial PC
- ❏ Integrated drive technology
- ❏ Digital input and output modules
- ❏ Analog input and output modules
- ❏ Absolute value encoder evaluation module
- ❏ Counter module
- ❏ Multifunction module
- ❏ Interface modules
- ❏ Communications adaptor modules
- ❏ Distributed I/O modules
- ❏ Various converter modules
- ❏ Various diagnostics modules

Power Supply Module PSB-01



- Short-circuit-proof
- Overvoltage protection on the output
- LED for function indication

The PSB-01 power supply module was designed for use in the Ωmega system.

At an input voltage of 230 V AC, the PSB-01 supplies an output voltage of 5 V that can be loaded with a 10-A continuous current. The output voltage is short-circuit-proof for an unlimited period of time.

Due to its compact structure, measuring 3 units of height (HE) and 8 units of depth (TE), the module can be used in all Ωmega systems.

With standard systems the connection is made via the backplane thus obviating the need for separate wiring.

Power Supply Module PSB-02

At an input voltage of 230 V AC, the PSB-02 supplies three output voltages of +5V, +12V and -12V that can be loaded with continuous currents of 10 A, 2 A and 0.5 A respectively. The output voltage is short-circuit-proof for an unlimited period of time.

Power Supply Module PSB-03

At an input voltage of 24 V DC, the PSB-03 supplies an output voltage of 5 V that can be loaded wiith a 10-A continuous current. The output voltage is short-circuit-proof for an unlimited period of time.

Power Supply Module PSB-04

At an input voltage of 24 V DC, the PSB-04 supplies three output voltages of +5V, +12V and -12V that can be loaded with continuous currents of 10 A, 2 A and 0.5 A respectively. The output voltage is short-circuit-proof for an unlimited period of time.

Power Supply Module PSB-05

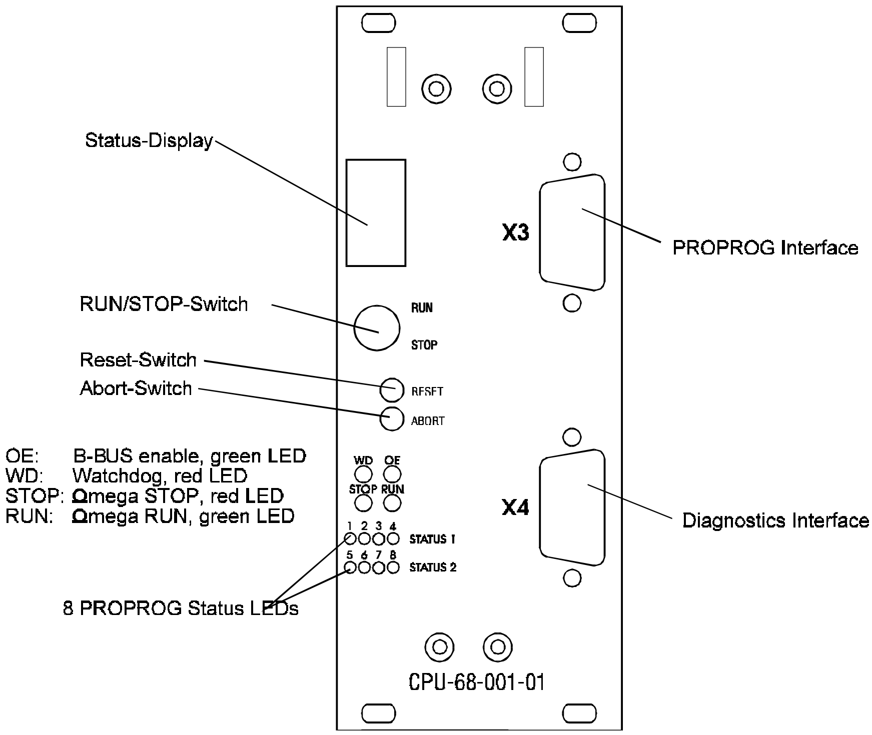
At an input voltage of 230 V AC, the PSB-05 supplies an output voltage of +5 V that can be loaded with a 20-A continuous current. The output voltage is short-circuit-proof for an unlimited period of time. The depth of this module is 12 TE.

Power Supply Module PSB-06

At an input voltage of 230 V AC, the PSB-05 supplies an output voltage of +5 V that can be loaded with a 30-A continuous current. The output voltage is short-circuit-proof for an unlimited period of time. The depth of this module is 12 TE.

Central Processing Unit CPU-68-001-01/...-02

- Motorola 68HC000 CPU, 16-Mhz clock speed
- 2 RS232 serial interfaces
- Status hexadecimal display 0...F
- 256 kB of S-RAM (static program and data memory)
- 256 kB of flash EPROM (non-volatile memory)
- B-BUS interface



The CPU-68-001-0x is a central processing unit based on the Motorola MC68HC000 processor with a clock frequency of 16 MHz.

The module is a single-height Eurocard constructed from two PCBs linked by a board-to-board plug-in connector. On the back, the connections of the parallel bus system, B-BUS are routed out. The front of the module contains the status display and the operator controls RUN/STOP switch, RESET switch, ABORT switch, as well as the PROPROG (download) interface and a diagnostics (or terminal) interface.

CPU-68-00y-01 have a memory module with 256 kB of SRAM and a 256-kB flash EPROM. Central processing units CPU-68-00y-02 are fitted with a memory module with 512 kB S-RAM and a 512-kB flasch EPROM.

In addition, it is possible to fit an adaptor for communications tasks (e.g. CAN-01/...-03, INT-S-01, ...) on the CPU's network expansion interface.

Central Processing Unit CPU-68-002-01/...-02

The CPU-68-002-0x central processing unit contains an additional system bus interface.

Central Processing Unit CPU-68-003-01/...-02

The CPU-68-003-0x central processing unit is fitted with a real-time clock (RTC) and a floating point processor (FPU).

Central Processing Unit CPU-68-004-01/...-02

The CPU-68-004-0x central processing unit is the same as the CPU-68-002-0x with the addition of a BCCOM-BUS interface.

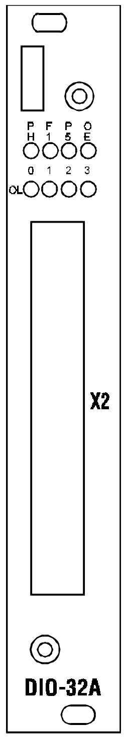
Central Processing Unit CPU-68-005-01/...-02

This is the same as the CPU-68-004-0x with the addition of a (RTC) real-time clock (RTC) and a floating point processor (FPU).

Central Processing Unit CPU-68-006-01/...-02

This is the same as the CPU-68-005-0x with the addition of a SCSI hard disk interface.

Digital Output Module DIO-32A



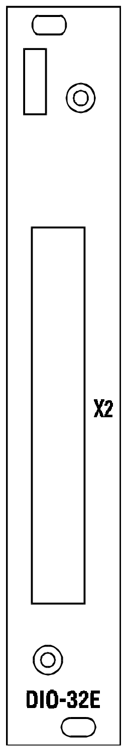
- 32 output channels, optically isolated, short-circuit-proof
- Display of operating status conditions by means of LEDs
- User-friendly cabling concept

The DIO-32A is an output module with 32 digital, 24-V industrial logic outputs that is designed for process automation and machine control in the Ωmega system.

The module is of single Eurocard format with a B-BUS connection on the back and a 50-pin ribbon cable plug-in connector on the front for connecting peripherals.

The process signals are connected to the screw terminals of conversion module UMS DIO-32. The conversion module is linked to output module DIO-32A via a plug-in ribbon cable; outside the module rack, it is clipped on to mounting rails in the switching cabinet. The module is externally supplied with 24 V DC.

The outputs are separated from the system by means of optocouplers. All the outputs are short-circuit-proof and can be loaded with up to 0.45 A. A red LED indicates when an output is being overloaded or short-circuited. This display is always shown as a group message for eight outputs. If the supply voltage drops below 18 V, LED PH goes out. With a defective fuse, LED F1 goes out. The system power supply is shown by LED P5 and LED OE indicates module enable.



Digital Input Module DIO-32E

- 32 output channels, optically isolated
- User-friendly cabling concept

The DIO-32E is an input module with 32 digital, 24-V industrial logic inputs that is designed for process automation and machine control in the Ωmega system.

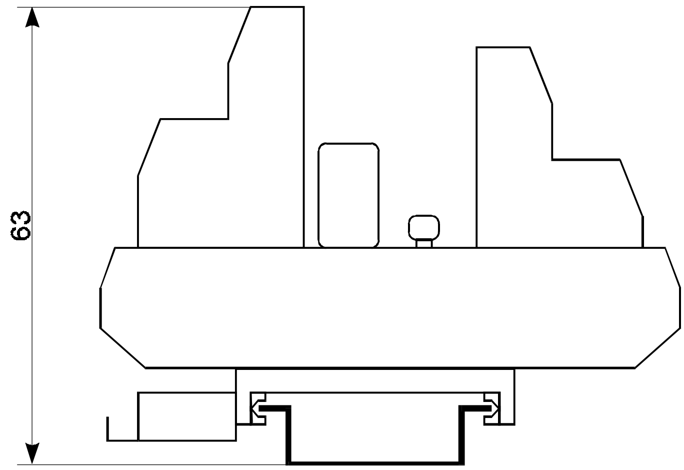
The module is of single Eurocard format with a B-BUS connection on the back and a 50-pin ribbon cable plug-in connector on the front for connecting peripherals.

The process signals are connected to the screw terminals of conversion module UMS DIO-32. The conversion module is linked to output module DIO-32E via a plug-in ribbon cable; outside the module rack, it is clipped on to mounting rails in the switching cabinet. The module is externally supplied with 24 V DC.

All the input signals are designed for 24-V operation and are separated from the system by means of optocouplers

Conversion Module UMS DIO-32

- Display of signal statuses using LEDs
- User-friendly cabling concept



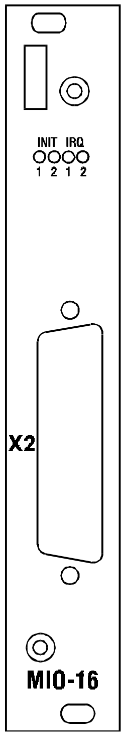
The UMS DIO-32 is a conversion module that was designed for digital input module DIO-32E and digital output module DIO-32A.

You mount the conversion module outside the module rack in the switching cabinet. Using a universal pedestal, you can clip the conversion modules on to commercially available DIN/EN-standard mounting rails. The connection to the DIO-32A or DIO-32E is by means of a plug-in 50-pin ribbon cable. The process signals are connected to the UMS DIO-32 conversion module's screw terminals.

When running in conjunction with a DIO-32A, connect the UMS DIO-32 to 24V/DC.

If you connect the UMS DIO-32 to a DIO-32E, connection of a reference potential (ground) is necessary.

The system separately displays the status conditions of all 32 input/output signals by means of LEDs. When operating with a DIO-32A, an LED also shows the supply voltage.



Multifunction Input Module MIO-16

- 16 digital input channels, optically isolated, interrupt-capable, edge-sensitive
- 4 input channels can be configured as timer inputs
- User-friendly cabling concept

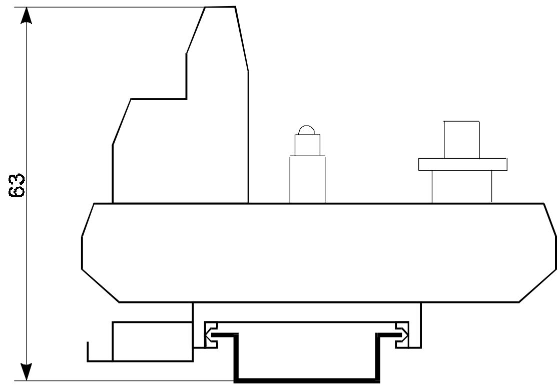
The MIO-16 is an input module with 16 24-V industrial logic digital inputs that are interrupt-capable and edge-sensitive. The module is designed for process automation and machine control in the Ωmega system. In addition, you can program eight hardware timers four of which can be run in cyclical mode as well as in counter and pulse width mode.

The module is of single Eurocard format with a SYSTEM-BUS connection on the back and a 37-pin sub-D female connector on the front for connecting peripherals.

The process signals are connected to the screw terminals of conversion module UMS-MIO-16. The conversion module is linked to multifunction module MIO-16 via a plug-in ribbon cable or a round cable; outside the module rack, it is clipped on to mounting rails in the switching cabinet.

All the input signals are designed for 24-V operation and are separated from the system by means of optocouplers.

Conversion Module UMS MIO-16



- LED display of all the input signals
- User-friendly cabling concept

The UMS MIO-16 conversion module connects the interrupt-capable edge-sensitive, digital input module MIO-16 to the peripherals.

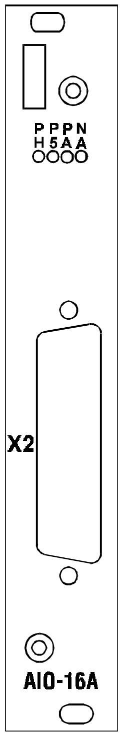
You mount the conversion module outside the module rack in the switching cabinet. Using a universal pedestal, you can clip the conversion modules on to commercially available DIN/EN-standard mounting rails. The connection to the MIO-16 is by means of a plug-in 37-pin ribbon cable; with relatively long distances, use a round cable. The process signals are connected to the UMS MIO-16 conversion module's screw terminals

Sixteen LEDs indicate the status conditions of all 16 input signals separately for each input.

Analog Output Module AIO-16A

- 16 output channels
- All outputs optically isolated
- 12-bit resolution
- Conversion time of 4 µs
- Output voltage ranges:

  0 – 10 V
  +/- 10 V
  +/- 5 V



The AIO-16A is an analog output module with 16 output channels. The module is designed for process automation and machine control in the Ωmega system.

The module is of single Eurocard format with a B-BUS connection on the back and a 25-pin sub-D male connector on the front for connecting peripherals.

The process signals are connected to the screw terminals of conversion module UMS AIO-16A. The conversion module is linked to output module AIO-16A via a plug-in ribbon cable or a round cable; outside the module rack, it is clipped on to mounting rails in the switching cabinet. The module is supplied externally with 24 V DC.

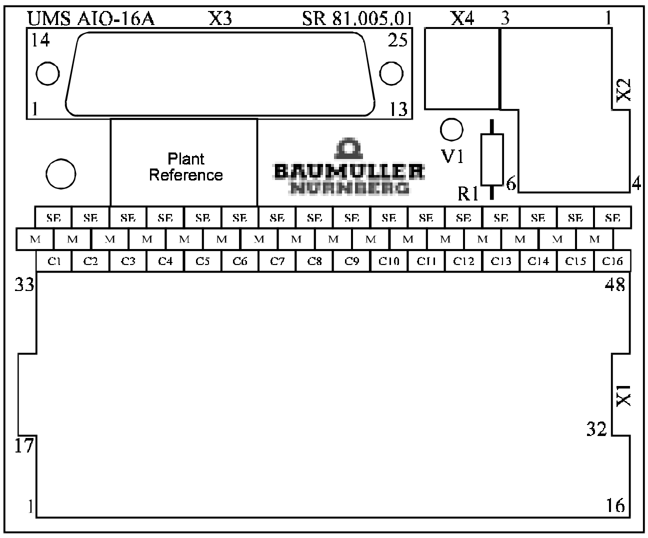
You have the option of configuring the output voltages for unipolar operation (0–10V), or for bipolar operation (+/-10V, +/-5V). The outputs are separated from the system by optocouplers.

Conversion Module UMS AIO-16A

- Can be clipped on to all common DIN/EN-standard rails
- User-friendly cabling concept

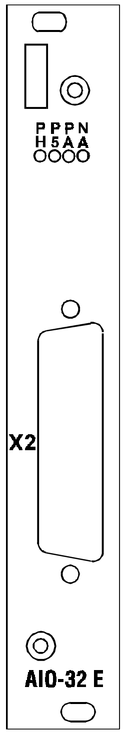
The UMS AIO-16A is a conversion module designed for the AIO-16A analog output module.

You mount the conversion module outside the module rack in the switching cabinet. Using a universal pedestal, you can clip the conversion modules on to commercially available DIN/EN-standard mounting rails. The connection to the AIO-16A is by means of a plug-in 25-pin ribbon cable or a round cable. The analog signals are connected to the UMS AIO-16A conversion module's screw terminals.



The UMS AIO-16A is supplied with 24 V DC. An LED indicates that voltage is being supplied.

Analog Input Module AIO-32E



- 32 single-ended or 16 differential input channels
- 12-bit resolution
- Conversion time of 10 µsec.
- All inputs are optically isolated
- Input voltage ranges:

  0 – 10 V
  ± 10 V
  ± 5 V

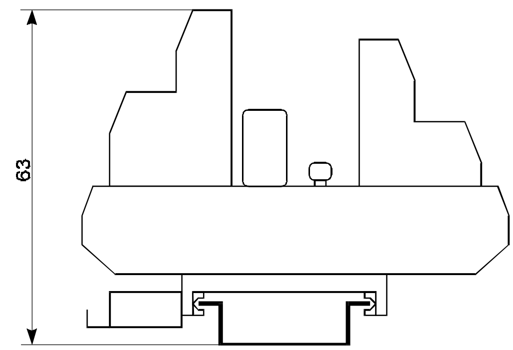
The AIO-32E is an analog input module with 32 input channels in single-ended operation or 16 input channels in differential mode. The module is designed for process automation and machine control in the Ωmega system.

The module is of single Eurocard format with a B-BUS connection on the back and a 37-pin sub-D male connector on the front for connecting peripherals.

The process signals are connected to the screw terminals of conversion module UMS AIO-32ES (single-ended operation) or UMS AIO-16ED (differential operation). The conversion module is linked to input module AIO-32E via a plug-in ribbon cable or a round cable; outside the module rack, it is clipped on to mounting rails in the switching cabinet. The module is supplied externally with 24 V DC.

You have the option of configuring all the input signals for unipolar operation between 0 and 10 V, or bipolar operation between -10 and + 10 V, or -5 and + 5 V. In addition, it is also possible to set the input gain (1x, 10x, 100x) for all channels. The intput signals are separated from the system by optocouplers.

Conversion Module UMS AIO-32ES/...-16ED



- Can be clipped on to all common DIN/EN-standard rails
- User-friendly cabling concept

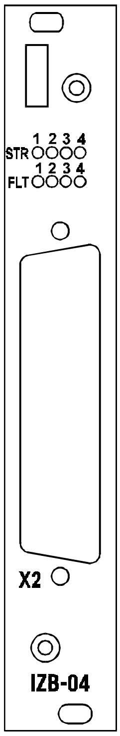
The UMS AIO-32ES/ ...-16ED is a conversion module designed for the AIO-32E analog input module.

In single-ended operation, use the UMS AIO-32ES, in differential operation, the UMS AIO-16ED is to be used.

You mount the conversion module outside the module rack in the switching cabinet. Using a universal pedestal, you can clip the conversion modules on to commercially available DIN/EN-standard mounting rails. The connection to the AIO-32E is by means of a plug-in 37-pin ribbon cable or a round cable. The analog signals are connected to the UMS AIO-32ES/ ...-16ED conversion module's screw terminals.

The UMS AIO-32ES/ ...-16ED is supplied with 24V DC. An LED indicates that voltage is being supplied.

Counter Module IZB-04



- 4 single-ended or differential counter channels
- 4-MHz input frequency
- Direction detection (90° phase offset)
- Tracer and reference cam inputs
- All signals optically isolated
- LED display of encoder disturbance
- LED display of counter latch signal

The IZB-04 is a counter module for a maximum of four incremental encoders that is designed for positioning and synchronization tasks in the Ωmega system.

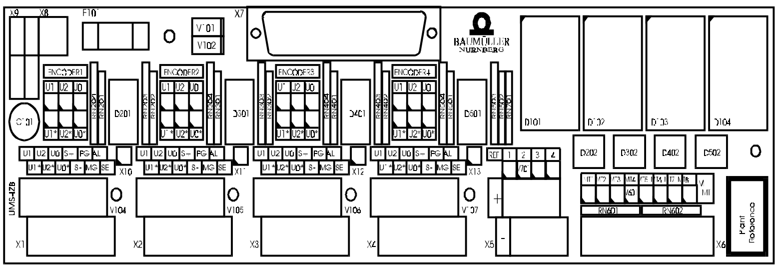
The module is of single Eurocard format with a B-BUS connection on the back and a 50-pin sub-D male connector on the front for connecting peripherals.

The encoder signals are connected to the screw terminals of conversion module UMS-IZB 04. The conversion module is linked to counter module IZB-04 via a plug-in ribbon cable or a round cable; outside the module rack, it is clipped on to mounting rails in the switching cabinet. The module is supplied externally with 24 V DC.

All the encoder signals are designed for 5 V, with the tracer and the reference cams being designed for 24 V and isolated from the system by optocouplers. LEDs indicate separately for each channel disturbances to the encoder or the encoder supply. There is also a display of the latch signals of the four counters.

Conversion Module UMS IZB-04

- Connections for four incremental encoders (rotary encoders)
- Provides four potential-isolated encoder supplies
- LED display of all encoder signals
- Monitoring of the encoder supply
- Monitoring of encoders with alarm line
- User-friendly cabling concept



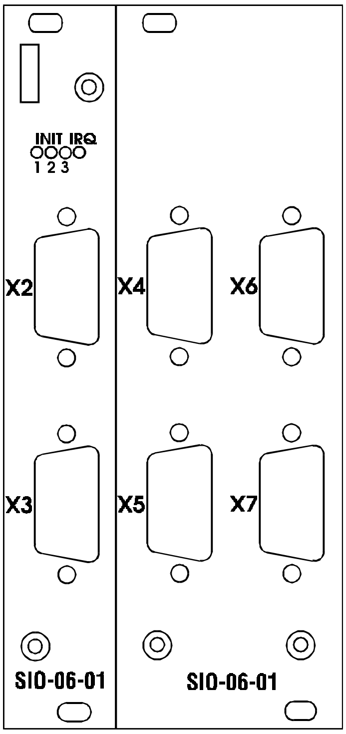
The UMS IZB-04 conversion module connects counter module IZB-04 to peripheral units like incremental encoders, tracers and reference cams.

You mount the conversion module outside the module rack in the switching cabinet. Using a universal pedestal, you can clip the conversion modules on to commercially available DIN/EN-standard mounting rails. The connection to the IZB-04 is by means of a plug-in 50-pin ribbon cable; with relatively long distances, use a round cable. The encoder signals are connected to the UMS IZB-04 conversion module's screw terminals

The UMS IZB-04 is supplied with 24 V DC. An LED indicates that voltage is being supplied. From this voltage, the UMS IZB-04 generates four 5-V DC operating voltages, which are galvanically isolated from one another, for the incremental encoders.

In the same way, six LEDs are provided to indicate separately for each encoder the status conditions of their individual signals (zero track, track 1, track 2 normal and inverted) U0, U0*, U1, U1*, U2 and U2*. Optical checking of the tracer and reference cam positions is also carried out by means of LEDs.

Interface Module SIO-06 - 01/02/03/04



- 6 RS232 or RS485 serial ports, optically isolated
  50 bps – 38.4 kbps

- Interrupt-capable module
  with settable interrupt
  levels (Level1 – Level7)

- User-friendly cabling concept

Interface module SIO-06-01 was designed for linking input/output units to the Ωmega system. The module has six RS232 ports that are independent of one another. Using the appropriate SIO software under PROPROG, you can configure each interface's baud rate, parity, number of stop bits per character, bits per character and CTS/RTS response.

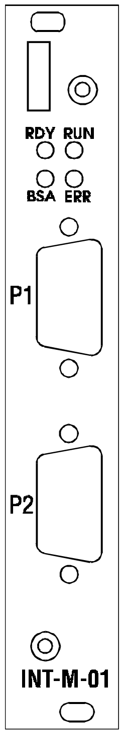
Interface module SIO-06-02 was designed for linking BMS or display and operating units. The module has six RS485 ports that are separate from one another.

Interface module SIO-06-03 has two RS232 and four RS485 ports; module SIO-06-04 has four RS232 and two RS485 ports each of which are separate from one another.

You can set the module's interrupt level using a dip switch. Seven levels are available with Level 7 having the highest priority and Level 1 the lowest.

The module is of single Eurocard format with a SYSTEM-BUS connection on the back and 9-pin sub-D female connectors on the front for connecting peripherals.

Interbus-S Master Controller INT-M-01



- 64 remote bus terminals, 4096 I/Os
- Integrated software for bus diagnostics
- Transfer rate of 500 kbps
- LED display of operating status conditions

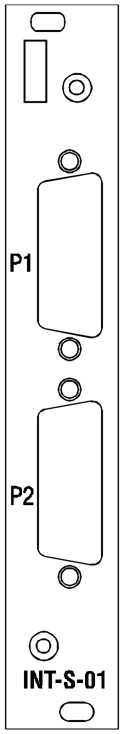
The INT-M-01 is a two-wire technology Interbus-S Master Controller. The module has an additional RS-232 port for diagnostic functions.

The module is of single Eurocard format with a SYSTEM-BUS connection on the back and an Interbus connection on the front.

Process signals are connected by means of SUB-D male connectors on the INT-M-01's front panel.

The RS-232 serial port and the Interbus-S are separated from the system by means of optocouplers.

Interbus-S Adaptor INT-S-01/...-03



- 16-MHz 80C32 CPU
- 8 or 9 IN words and
  8 or 9 OUT words
  as rapid cyclical data
  via the Interbus (INT-S-01)
- Can be expanded by 9 IN words and 9 OUT words (INT-S-03)
- Transfer of non-time-critical data using PCP

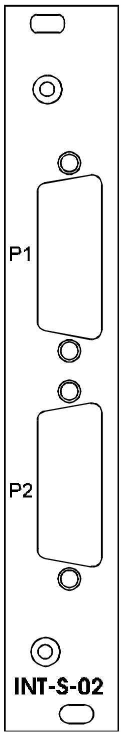
The INT-S-01/...-03 is an Interbus-S slave for connection to the Interbus-S local bus.

The module is is of single Eurocard format and is designed as a daughterboard for the CPU-68.

Interbus signals are connected by means of SUB-D male connectors on the INT-S-01's front panel.

The Interbus-S interface is separated from the system by means of an optocoupler.

Interbus-S Adaptor INT-S-02/...-04



- 16-MHz 80C32 CPU
- 8 or 9 IN words and
  8 or 9 OUT words
  as rapid cyclical data
  via the Interbus (INT-S-02)
- Can be expanded by 9 IN words and 9 OUT words
  (INT-S-04)
- Transfer of non-time-critical data using PCP
- Interrupt-capable module with settable interrupt levels (Level1 – Level7).

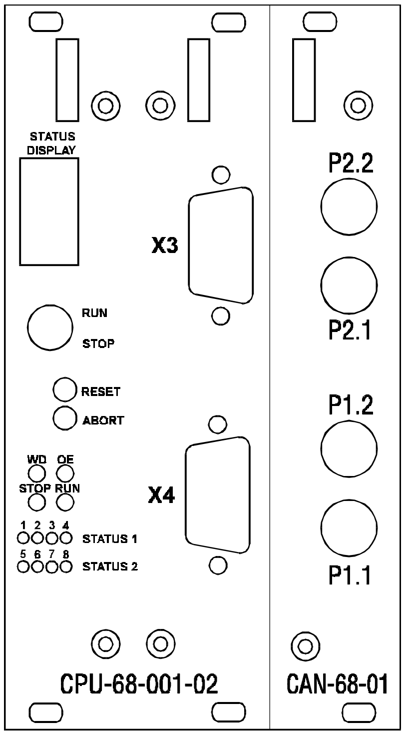
The INT-S-02/...-04 is an Interbus-S slave for connection to the Interbus-S local bus.

The module is is of single Eurocard format with a SYSTEM-BUS connection on the back.

Interbus signals are connected by means of SUB-D male connectors on the INT-S-02's front panel.

The Interbus-S interface is separated from the system by means of an optocoupler.

CAN Adaptor CAN-68-01/02/03



- BASIC-CAN Controller 82C200
- Galvanic isolation from the CAN bus via DC/DC converters and fast optocouplers
- Data transfer rate of up to 1Mbps
- Two independent CAN nodes with CAN-68-02 and CAN-68-03

The CAN-68-01/02/03 adaptors are intelligent interface cards that are designed as daughterboards for the CPU-68.

They are used for communicating with other Ωmega computer systems, PCs as well as for structuring distributed CAN networks with CAN I/O modules.

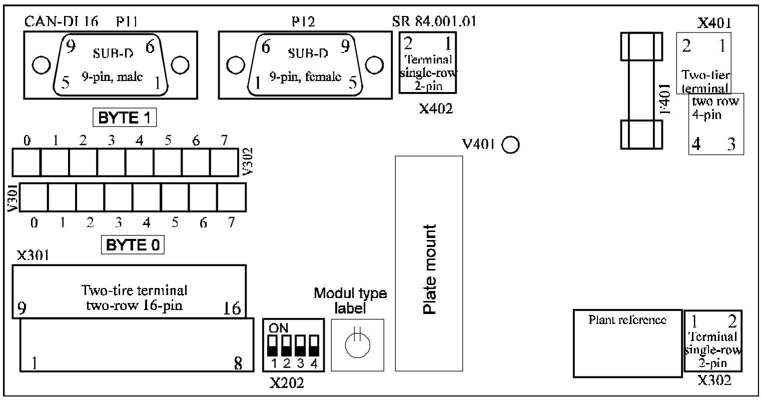
In this context, a differentiation is made between:

- **CAN-68-01:** One CAN interface with galvanic isolation from the bus
- **CAN-68-02:** Two CAN interfaces separate from one another with galvanic isolation from the bus
- **CAN-68-03:** Two CAN interfaces with galvanic isolation from the bus and a bidirectional CAN bridge between interfaces 1 and 2

Each CAN node has a 16-MHz 80C32 microcontroller and is linked to the CPU-68 via dual-ported RAM.

The integrated serial port can be used for diagnostics and commissioning.

Distributed Input Module CAN-DI-16



- 16 digital 24-V process inputs
- LED display of input status conditions
- Galvanic isolation from the CAN bus via DC/DC converters and optocouplers
- Industry-standard I/O plug connectors

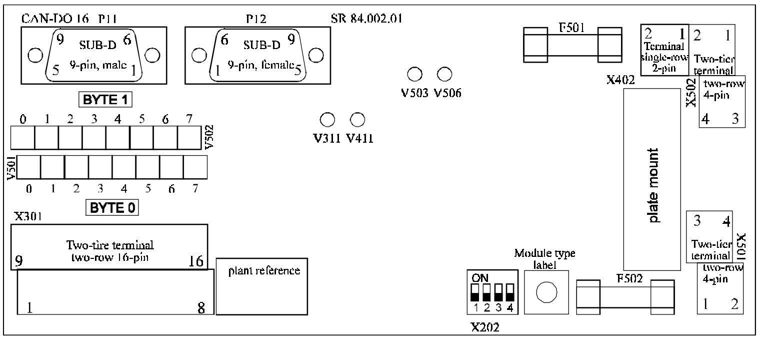
The CAN-DI-16 distributed CAN I/O module is an input module with 16 digital, 24-V industrial logic inputs.
Using a universal pedestal, you can clip the module on to commercially available DIN/EN-standard mounting rails.
Process signals and the auxiliary voltage for the I/O module are connected by means of screw terminals directly on the module.

The status conditions of the signals of the inputs are shown by green LEDs that are optically assigned to the associated terminals. Another green LED indicates the status of the auxiliary voltage.

The modules have a labelling field as well as a plate mount for individual labelling of the inputs.
You set the module address (0-15) using dip switches.

The distributed I/O modules work with a P82C150 serial linked I/O (SLIO) module, with bus interfacing being carried out by an ISO/DIN 11898-standard transceiver module with galvanic isolation by means of optocouplers and DC/DC converters.
The bus link is made with a 9-pin Sub-D connector with pin assignments according to CiA recommendations.

Distributed Output Module CAN-DO-16



- 16 digital outputs 24V/0.5A, short-circuit-proof
- Display of overload/short-circuit
- LED display of output status conditions
- Galvanic isolation from the CAN bus via DC/DC converters and optocouplers
- Industry-standard I/O plug-in connectors

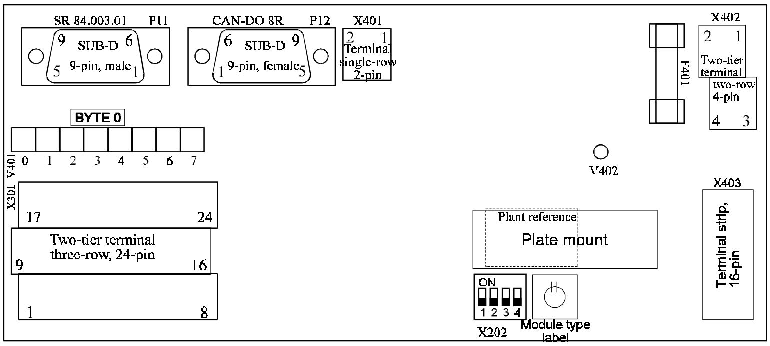
The CAN-DO-16 distributed CAN I/O module is an output module with 16 digital, 24-V industrial logic outputs.
Using a universal pedestal, you can clip the module on to commercially available DIN/EN-standard mounting rails.
Process signals and the auxiliary voltage for the I/O module are connected by means of screw terminals directly on the module.

The status conditions of the signals of the outputs are shown by green LEDs that are optically assigned to the associated terminals.

The modules have a labelling field as well as a plate mount for individual labelling of the outputs.
You set the module address (0-15) using dip switches.

The distributed I/O modules work with a P82C150 serial linked I/O (SLIO) module, with bus interfacing being carried out by an ISO/DIN 11898-standard transceiver module with galvanic isolation by means of optocouplers and DC/DC converters.
The bus link is made with a 9-pin Sub-D connector with pin assignments according to CiA recommendations.

Distributed Relay Module CAN-DO-8R



- 8 relay outputs
- Can be expanded by 8 relay outputs or 8 digital outputs
- Monostable changeover contacts
- Switching voltages of up to 250 V(AC)/ 125 V(DC)
- LED display of switching status conditions
- Galvanic isolation from the CAN bus via DC/DC converters and optocouplers
- Industry-standard I/O plug connectors

The CAN-DO-8R distributed CAN I/O module is an output module with eight monostable relay changeover contacts that can be loaded with a maximum of 6 A in continuous operation.
You can add eight relay changeover contacts using expansion module CAN-EO 8R (Order Number: 215 835); eight digital outputs can be added using expansion module CAN-EO 8 (Order Number: 215 834).

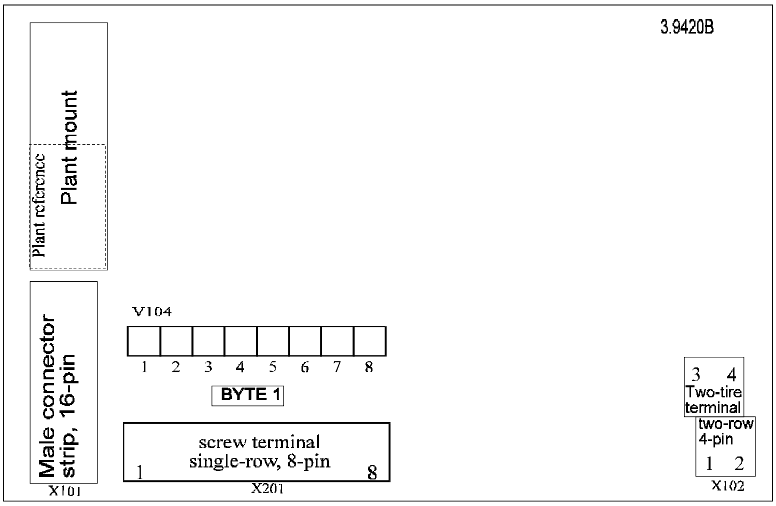
Using a universal pedestal, you can clip the module on to commercially available DIN/EN-standard mounting rails.
Process signals and the auxiliary voltage for the I/O module are connected by means of screw terminals directly on the module.

The status conditions of the signals of the outputs are shown by green LEDs that are optically assigned to the associated terminals.

The modules have a labelling field as well as a plate mount for individual labelling of the outputs.
You set the module address (0-15) using dip switches.

The distributed I/O modules work with a P82C150 serial linked I/O (SLIO) module, with bus interfacing being carried out by an ISO/DIN 11898-standard transceiver module with galvanic isolation by means of optocouplers and DC/DC converters.
The bus link is made with a 9-pin Sub-D connector with pin assignments according to CiA recommendations.

Distributed Expansion Module CAN-EO 8



- 8 digital outputs 24V/0.5A, short-circuit-proof
- Display of overload/ short-circuit
- LED display of output status conditions
- Industry-standard I/O plug connectors

The CAN-EO-8 distributed expansion module is an output module with eight digital 24-V industrial logic outputs.
The module is connected by means of a 16-pin ribbon cable to the CAN-DO 8R (Order Number: 214 899) CAN I/O module. This makes it possible to control eight relays and eight digital outputs via a distributed CAN node.

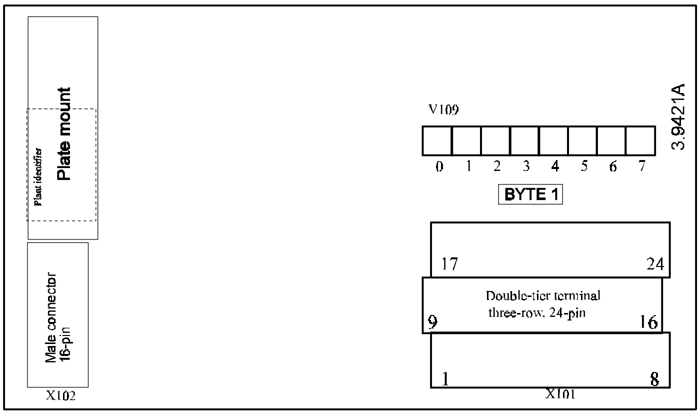
Using a universal pedestal, you can clip the module on to commercially available DIN/EN-standard mounting rails.
Process signals are connected by means of screw terminals directly on the module.

The status conditions of the signals of the outputs are shown by green LEDs that are optically assigned to the associated terminals.

The modules have a labelling field as well as a plate mount for individual labelling of the outputs.
You do not need to set the module address, since all eight outputs are assigned to the same identifier as the relay base module, CAN-DO 8R.
As a result, the system can treat the CAN-DO-8R module in conjunction with expansion module CAN-EO-8 as one distributed CAN node with 16 outputs.

Distributed Expansion Module CAN-EO 8R

- 8 relay outputs
- Monostable changeover contacts
- Switching voltages of up to 250 V (AC)/125 V (DC)
- LED display of switching status conditions
- Industry-standard I/O plug connectors



The CAN-EO-8R distributed expansion module is an output module with eight monostable relay changeover contacts that can be loaded with a maximum of 6 A in continuous operation.
The module is connected by means of a 16-pin ribbon cable to the CAN-DO 8R (Order Number: 214 899) CAN I/O module. This makes it possible to control 16 relays via one distributed CAN node.

Using a universal pedestal, you can clip the module on to commercially available DIN/EN-standard mounting rails.
Process signals are connected by means of screw terminals directly on the module.

The switching status conditions of the relays are shown by green LEDs that are optically assigned to the associated terminals.

The modules have a labelling field as well as a plate mount for individual labelling of the outputs.
You do not need to set the module address, since all eight relays are assigned to the same identifier as the base module, CAN-DO 8R
As a result, the system can treat the CAN-DO-8R module in conjunction with expansion module CAN-EO-8 as one distributed CAN node with 16 outputs.

3 SOFTWARE OVERVIEW

On the basis of the system description with the system overview of PROPROG-ProConOS and the PROPROG menu structure, we will show you how a modular project is structured.

3.1 System Description

Cyclical PLC programs and controller functions exist in parallel in the system and can use common variables. Controller functions are triggered by interrupts such that the PLC program is interrupted:

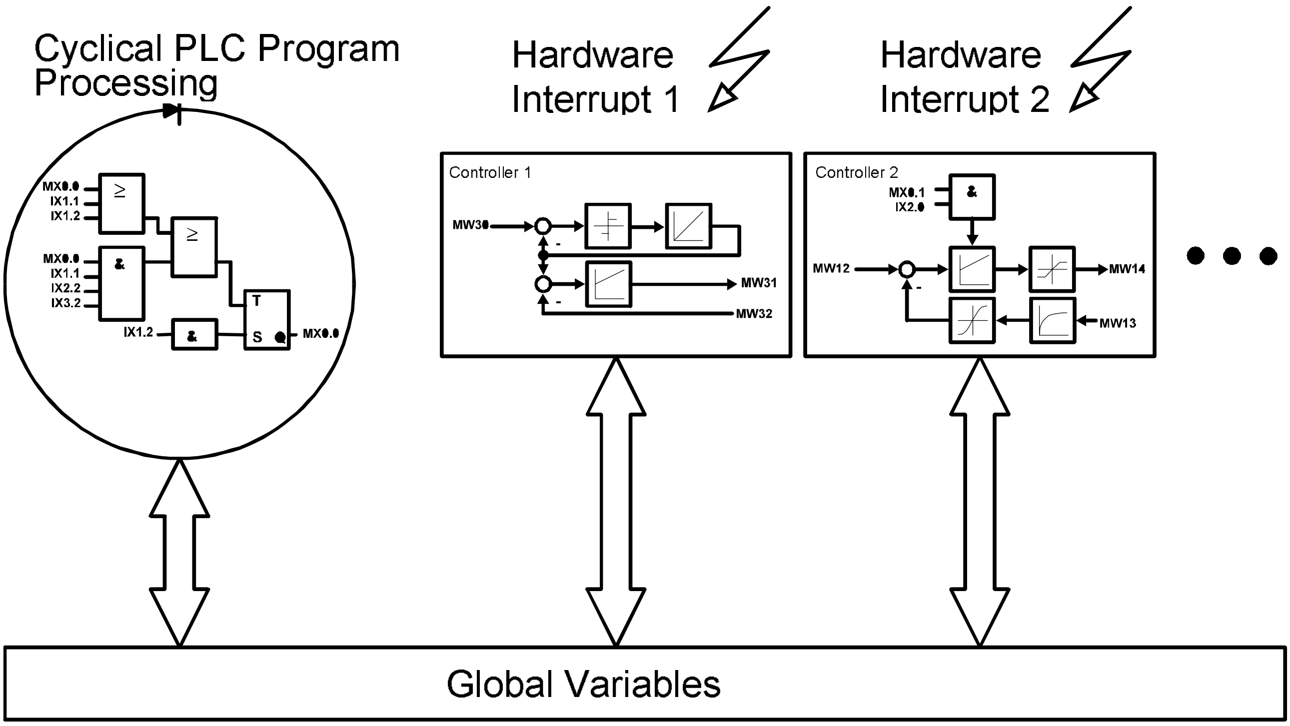


Figure 3.1: Cyclical and Controller Program Sections

The Ωmega system's command set corresponds to instruction list (IL) which is the common language level of the programming system and which all the other programming languages (FUP, LD, SFC) are converted to using compilers.

The programs are translated from the internal INL code format of the programming system into an intermediate Ωmega system code. During code generation, the system carries out various plausibility checks for programming errors. The executable code is generated on the run-time system.

3.1.1 System Overview PROPROG - ProConOS

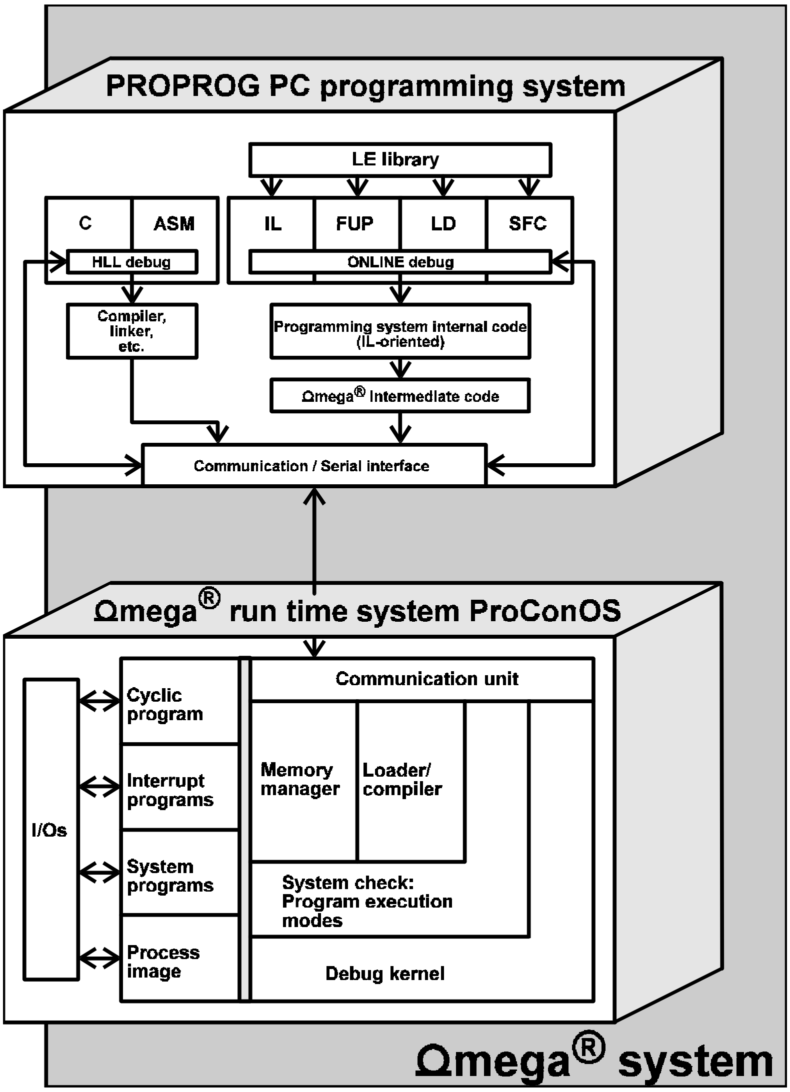


Figure 3.2: System Overview of PROPROG-ProConOS

3.2 PROPROG Menu Structure

An overview of the menu structure of the PROPROG PC programming system is shown below:

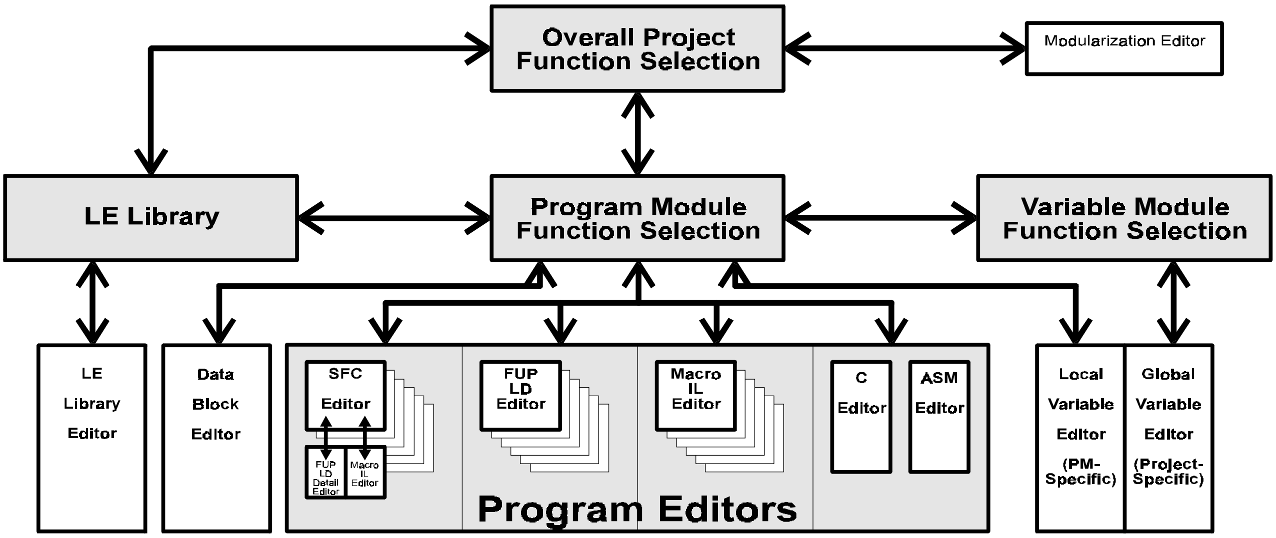


Figure 3.3: Menu Structure of PROPROG

3.3 Project Structure

An Ωmega® project consists of several program modules which can be linked together to an overall program in the same way as with high-level language object files. One program module can contain one or more subroutines (SPs, ISPs, IRPs). Each executable command must be part of a subroutine. You can define subroutines with and without parameter transfer

Cyclical program sections (compare with PLC):

Subroutine SP 0 is the cyclical main program to which the system branches at the start of the cycle. Processing of the cyclically executed program sections of a project is determined by the call structure of the subroutines with nested calling of the subroutines being possible.
It is irrelevant which program module contains the subroutine and in which programming language (SFC, FUP, LD, IL, C or ASM) you wrote the subroutine.

Acyclical program sections:

Interrupt controllers and interrupt service programs (IRPs and ISPs) are processed immediately on occurrence of the event stipulated for them. The system pauses execution of cyclical program sections while interrupt processing is being carried out.
In general, interrupt programs are configured in programming languages FUP and IL.

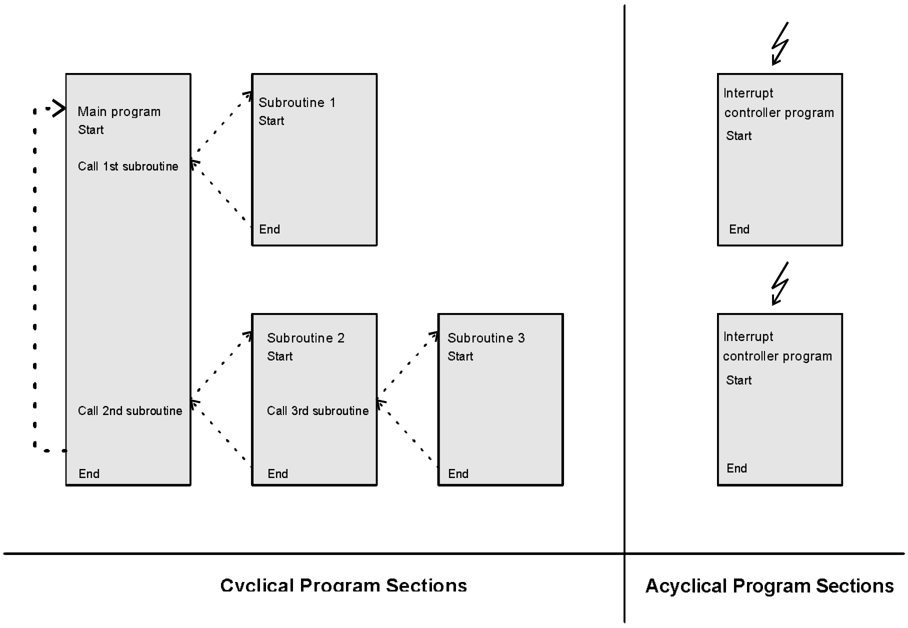


Figure 3.4: Project Structure

3.4 Fundamentals of the Modular Concept

Together with the subplan concept, the modular concept allows you to structure complex regulation and control tasks. PROPROG makes it possible to divide a program into various program modules (PMs) and variable modules (VMs). The program modules allow a further division into subplans.

This enables the implementation of projects which, due to their size, cannot be implemented by simple linear programming.

Dividing the overall project into modules and subplans results in high levels of flexibility with regard to creating, changing and testing programs. Big projects can be structured more clearly and thus handled more easily. Modules from existing projects can be integrated easily. If you make changes to the program, only one program module is involved. This allows faster program alterations.

3.4.1 Sections of a Project

The overall project consists of subprojects, i.e. the program modules. These program modules in their turn consist of subplans comprising several screen pages.

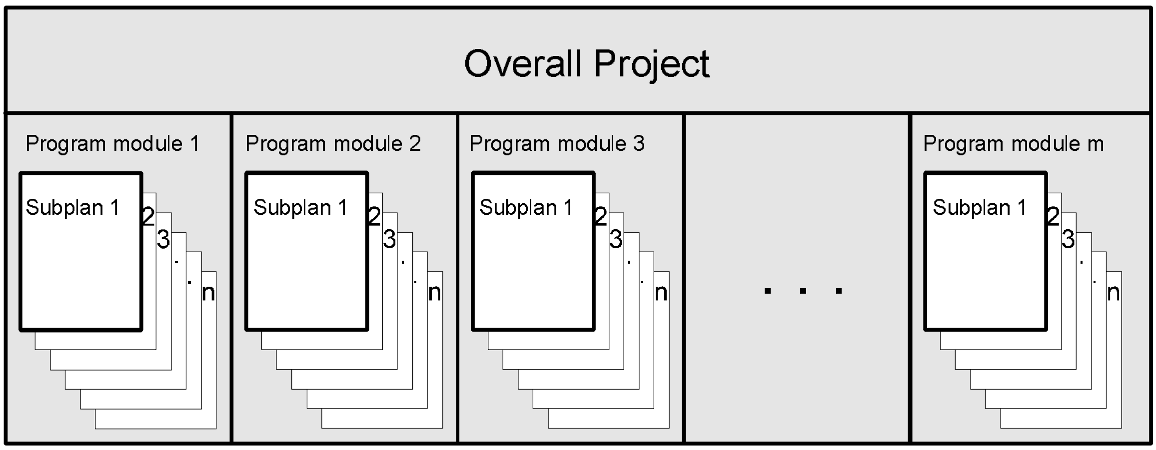


Figure 3.5: Project Sections

In PROPROG, there are different work levels that correspond to the sections of a project:

- **Overall project level**
  with module management

- **Module editing level**
  with editors for programs and variables

On calling PROPROG, the overall project level is automatically activated. At this level, you can call the functions which refer to the overall project e.g. project management and printing (the overall project). The modular concept editor can be called via menu item Project Management. The editor manages program modules and variable modules, i.e. it defines them and assigns them to one another (see programmer's manual Chapter 5.3, The Modular Concept Editor). Several program modules can be assigned to each variable module and by the same token you can assign several variable modules to each program module.

Individual program and variable modules are edited at the module editing level. You can switch between the program and variable modules as desired.

3.4.2 Program Modules (PMs) and Variable Modules (VMs)

Program modules and variable modules provide a means of structuring an overall project. Structuring your projects in this way into clearly laid-out program sections which belong together from a functional or logical point of view makes it easier to carry out configuration and program creation, program testing and commissioning.

Program Modules:

A program module is structured like a completely independent project without a modular concept and can be divided into several subplans.

You use one of the program editors below to edit a program module (PM):

- ❑ SFC .... sequential function chart
- ❑ FUPLD .... function plan ladder diagram
- ❑ Macro-IL.......... instruction list
- ❑ C..................... high-level language
- ❑ S..................... assembly language

Each program module which was created in SFC, FUPCOP or Macro-IL has a variable editor in which all the variables are listed that the program uses. Variables which are (globally) defined in an assigned variable module are marked with an "E" (for external).

Variable Modules:

A variable module (for SFC, FUPLD, Macro-IL) consists of a structured variable management.

Variables which are used in different program modules can and should be listed and managed centrally in the variable editors of the variable modules. Alterations in the variable modules can be transferred to the assigned program modules via an update function.

Assignment of program and variable modules

The diagram below shows an example of the assignment of several program modules to a variable module:

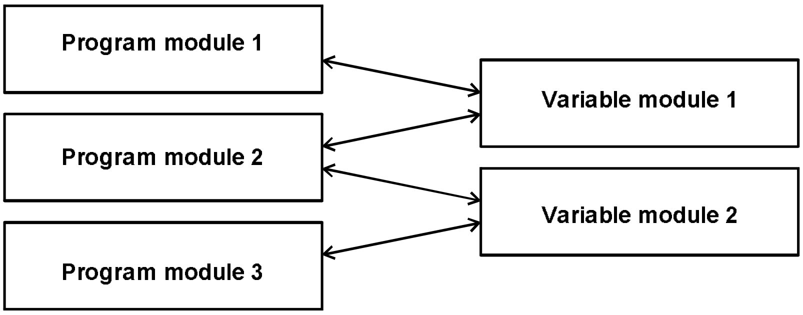


Figure 3.6: Program and Variable Module Assignment

Variable module 1 administers the variables which are used in program module 1 as well as in program module 2; variable module 2 administers the variables which are used in program module 2 and program module 3. This means that variable modules 1 and 2 are assigned two program modules each (1 and 2 or 2 and 3). Program module 2 is assigned two variable modules whereas program modules 1 and 3 are only assigned one variable module each.

3.5 Modular Project Creation

Dividing complex regulation and control projects into functional or logical subtasks makes them clearer and easier to implement. PROPROG provides several methods for this:

3.5.1 TOP DOWN

First, you divide the overall project into independent subtasks (PMs) while organizing the variables such that they do not use up too much memory. The individual subtasks can now be programmed by different employees.

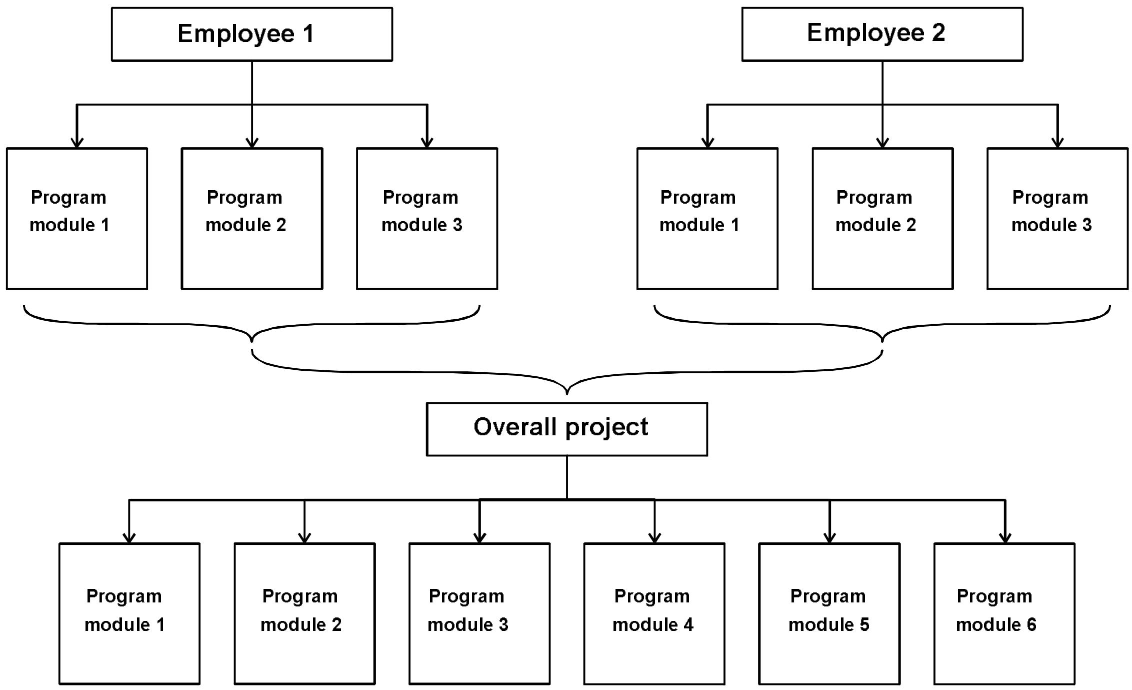


Figure 3.7: TOP DOWN Configuration

3.5.2 BOTTOM UP

You start by programming and create a program with subplans; later you realize that it is necessary to divide the project into modules to get a better overview and to administer the large number of variables. You now divide the program into program and variable modules.

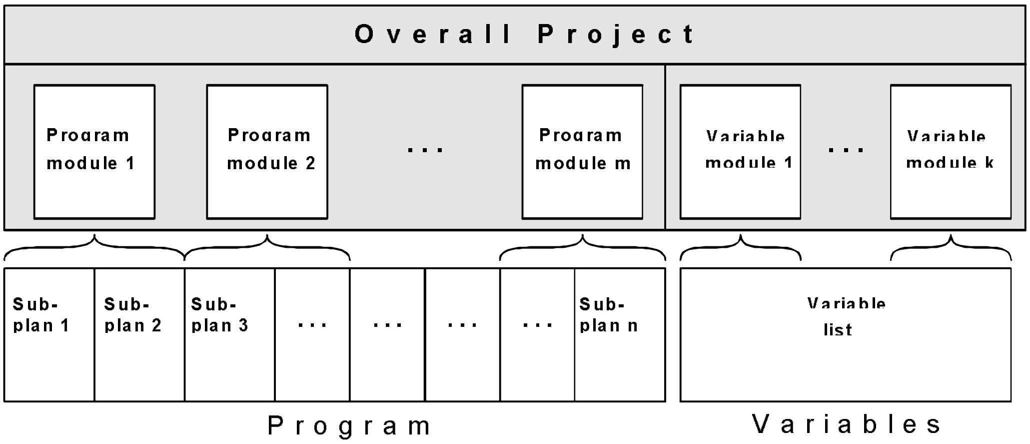


Figure 3.8: BOTTOM UP Configuration

3.5.3 Task-Specific

Different program modules can be created using different editors (EL, FUPCOP, Macro-INL, C, Assembler). This means that you can choose the editor which is most suitable for solving the subtask in question.

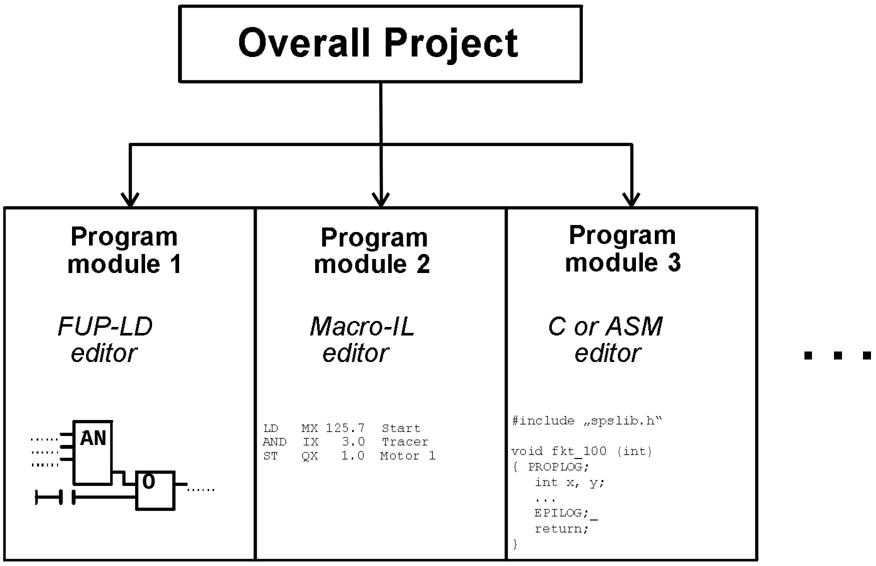


Figure 3.9: Task-Specific Configuration

3.5.4 Subproject Transfer

You can easily adapt and reuse program sections which have already been created for other projects (i.e. one or more program modules or parts of them).

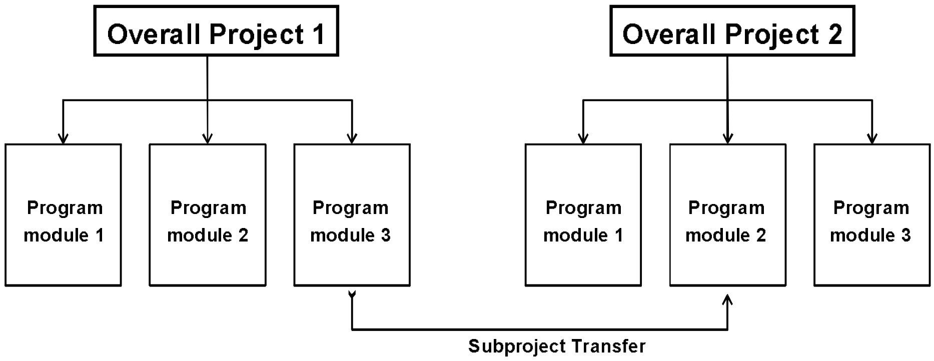


Figure 3.10: Subproject Transfer

3.6 Command Overview

Command	Remarks
LD	Load
LDN	Load negated
LDIR	Load index register
LDI	Load indexed
LDA	Load operand address
ST	Store
STN	Store negated
STI	Store indexed
S	Set
R	Reset
AND	Logical AND
ANDN	Logical AND, negated
OR	Logical OR
ORN	Logical OR, negated
XOR	Logical EXCLUSIVE OR
XORN	Logical EXCLUSIVE OR, negated
NOT	Logical NOT
AND(	Logical AND
ANDN(	Logical AND, negated
OR(	Logical OR
ORN(	Logical OR, negated
XOR(	Logical EXCLUSIVE OR
XORN(	Logical EXCLUSIVE OR, negated
LT	Less than
LE	Less than or equal to
EQ	Equal to
GE	Greater than or equal to
GT	Greater than
NE	Not equal to
LT(	Less than
LE(	Less than or equal to
EQ(	Equal to
GE(	Greater than or equal to
GT(	Greater than
NE(	Not equal to
ADD	Addition
SUB	Subtraction
MUL	Multiplication
DIV	Division
MOD	Modulo
ADD(	Addition
SUB(	Subtraction
MUL(	Multiplication
DIV(	Division
MOD(	Modulo
)	Closing parenthesis

Command	Remarks
INC DEC	Incrementation Decrementation
NEG	Logical negation
ABS	Absolute value
SQRT POW2 POW10 LN LN+1 LOG LG2 EXP EX-1	Square root 2^ operand 10^ operand Natural logarithm Natural logarithm (accumulator + 1) Logarithm base 10 Logarithm base 2 Exponential function (e^x) Exponential function((e^x)-1)
SIN *) COS *) TAN *) ASIN *) ACOS *) ATAN *) SINH *) COSH *) TANH *) ATANH *) SINCOS *)	Sine Cosine Tangent Inverse sine Inverse cosine Inverse tangent Hyperbolic sine Hyperbolic cosine Hyperbolic tangent Inverse hyperbolic tangent Sine and cosine (simultaneous)
SHL SHR ASL ASR ROL ROR	Logical shift left Logical shift right Arithmetic shift left Arithmetic shift right Logical rotate left Logical rotate right
PED NED	Positive edge detection Negative edge detection
BOOL SSINT SINT SDINT REAL *) LREAL *) BCD BCD-D	Change accumulator to data type BOOL Change accumulator to data type SSINT Change accumulator to data type SINT Change accumulator to data type SDINT Change accumulator to data type REAL Change accumulator to data type LREAL Change accumulator to data type BCD Change accumulator to data type BCD-D
JMP JMPC JMPN LABEL	Jump, unconditional Jump, conditional (RLO = 1) Jump, conditional (RLO = 0) Label
DEF END	Definition, start Definition, end
CAL CALC CALN	Call, unconditional Call, conditional (RLO = 1) Call, conditional (RLO = 0)

Command	Remarks
RET RETC RETN	Return, unconditional Return, conditional (RLO = 1) Return, conditional (RLO = 0)
DEF_PV DEF_PA	Definition, parameter value Definition, parameter address
PARV PARA	Parameter transfer, value Parameter transfer, address
COPY FILL TCOPY	Copy memory Set memory Copy text
ERROR	Error module
T-MS T-S T-MIN	Milliseconds timer Seconds timer Minutes timer
C-DOWN C-UP	Downwards counter Upwards counter
OPEN **) CLOSE **) READ **) WRITE **) SEEK **)	Open file Close file Read file Write file Position file pointer

*) Available on systems with an FPU only

**) Available on systems with mass storage only

Refer to the PROPROG programming manual

3.7 Overview of Operands *)

Operand	Operand Range			Input Type	Remarks
IX	0000.0	.....	2047.7	Dec.	Input, bit
IB	0000	.....	2047	Dec.	Input, byte
IW	0000	.....	2046	Dec. (MOD 2)	Input, word
ID	0000	.....	2044	Dec. (MOD 4)	Input, double word
QX	0000.0	.....	2047.7	Dec.	Output, bit
QB	0000	.....	2047	Dec.	Output, byte
QW	0000	.....	2046	Dec. (MOD 2)	Output, word
QD	0000	.....	2044	Dec. (MOD 4)	Output, double word
MX	0000.0	.....	8191.7	Dec.	Marker, bit
MB	0000	.....	8191	Dec.	Marker, byte
MW	0000	.....	8190	Dec. (MOD 2)	Marker, word
MD	0000	.....	8188	Dec. (MOD 4)	Marker, double word
MR	0000	.....	4092	Dec. (MOD 4)	Marker, real
MLR	0000	.....	4088	Dec. (MOD 8)	Marker, long real
MSX	0000.0	.....	1023.7	Dec.	System marker, bit
MSB	0000	.....	1023	Dec.	System marker, byte
MSW	0000	.....	1022	Dec. (MOD 2)	System marker, word
MSD	0000	.....	1020	Dec. (MOD 4)	System marker, double word
ROSX	0000.0	.....	1023.7	Dec.	Read only system marker, bit
ROSB	0000	.....	1023	Dec.	Read only system marker, byte
ROSW	0000	.....	1022	Dec. (MOD 2)	Read only system marker, word
ROSD	0000	.....	1020	Dec. (MOD 4)	Read only system marker, double word
$X	0000.000	.....	1023.255	Dec.	Step marker, bit
$B	0000	.....	1023	Dec.	Step marker, byte
PX	00	.....	31	Dec.	Parameter, bit
PB	00	.....	31	Dec.	Parameter, byte
PW	00	.....	31	Dec.	Parameter, word
PD	00	.....	31	Dec.	Parameter, double word
PR	00	.....	31	Dec.	Parameter, real
PLR	00	.....	31	Dec.	Parameter, long real
DX	000.000.0	.....	511.255.7	Dec.	Data block, bit
DB	000.000	.....	511.255	Dec.	Data block, byte
DT	000.000	.....	511.255	Dec.	Data block, Text
DW	000.000	.....	511.254	Dec. (MOD 2)	Data block, word
DD	000.000	.....	511.252	Dec. (MOD 4)	Data block, double word
DR	000.000	.....	511.252	Dec. (MOD 4)	Data block, real
DLR	000.000	.....	511.248	Dec. (MOD 8)	Data block, long real
MAB	00	.....	63	Dec.	Address marker, bit
MAW	00	.....	63	Dec.	Address marker, word
MAD	00	.....	63	Dec.	Address marker, double word

*) Operand type and ranges as well as marker ranges may be limited in dependence on the CPU type, memory options and the modules used. Real and long real operands are only available on CPU modules with an FPU, with markers not overlapping the other types of operands. For individual limitations, refer to the descriptions of the respective modules.

Operand	Operand Range	Input Type	Remarks
#	-128 ..... +127	Dec.	Decimal constant, byte
#W	-32768 ..... +32767	Dec.	Decimal constant, word
#D	-2147483648 ... +2147483647	Dec.	Decimal constant, double word
#H	00 ..... FF	Hex	Hexadecimal constant, byte
#WH	0000 ..... FFFF	Hex	Hexadecimal constant, word
#DH	00000000 ..... FFFFFFFF	Hex	Hexadecimal constant, double word
#HS	-7F ..... +7F	Hex	Hexadecimal constant, byte, signed
#WHS	-7FFF ..... +7FFF	Hex	Hexadecimal constant, word, signed
#DHS	-7FFFFFFF ..... +7FFFFFFF	Hex	Hexadecimal constant, double word, sig.
#X	0, 1	Bin	Binary constant, bit
#B	00000000 11111111	Bin	Binary constant, byte
#R	1.18 E-38 < |#R| < 3.40 E38	Exp	Constant, real
#LR	2.23 E-308 <|#LR| < 1.79 E308	Exp	Constant, long real
#PI	(3.14 ...)		Constant, PI
#E	(2.71 ...)		Constant, e
T	000 255	Dec.	Software timer, bit
TW	000 255	Dec.	Software timer, word
C	000 255	Dec.	Software counter, bit
CW	000 255	Dec.	Software counter, word
EDG	000.0 ..... 255.7	Dec.	Edge operand, bit
SX	0000000.0 ..... 8388607.7	Dec.	System bus (peripherals), bit
SB	0000000 8388607	Dec.	System bus (peripherals), byte
SW	0000000 8388606	Dec. (MOD 2)	System bus (peripherals), word
SD	0000000 8388604	Dec. (MOD 4)	System bus (peripherals), double word
BX	000.00000.0 ..... 511.65535.7	Dec.	Peripheral data block, bit
BB	000.00000 ..... 511.65535	Dec.	Peripheral data block, byte
BW	000.00000 ..... 511.65534	Dec. (MOD 2)	Peripheral data block, word
BD	000.00000 ..... 511.65532	Dec. (MOD 4)	Peripheral data block, double word
CIX	0000.0 ..... 8191.7	Dec.	Communication (internal), bit
CIB	0000 8191	Dec.	Communication (internal), byte
CIW	0000 8190	Dec. (MOD 2)	Communication (internal), word
CID	0000 8188	Dec. (MOD 4)	Communication (internal), double word
CEX	00.0000.0 ..... 63.8191.7	Dec.	Communication (external), bit
CEB	00.0000 ..... 63.8191	Dec.	Communication (external), byte
CEW	00.0000 ..... 63.8190	Dec. (MOD 2)	Communication (external), word
CED	00.0000 ..... 63.8188	Dec. (MOD 4)	Communication (external), double word
NX	00000.0 ..... 65535.7	Dec.	Network expansion, bit
NB	00000 65535	Dec.	Network expansion, byte
NW	00000 65534	Dec. (MOD 2)	Network expansion, word
ND	00000 65532	Dec. (MOD 4)	Network expansion, double word
LAB	00000 32767	Dec.	Program label
LAS	00000 32767	Dec.	Labels in the SFC editor
SP	0000 1023	Dec.	Subroutine
FB	0000 1023	Dec.	Firmware block
IRP	000.000.00001..511.127.16384	Dec.	Interrupt Controller Program
ISP	000.000.00001..511.127.16384	Dec.	Interrupt Service Program

3.8 Debugging Table

The table below shows a list of the debugging functions that are allowed with individual operands.

P: .... Status of the operand can be polled, i.e. the operand is valid for the variable status and the trigger reference list

T: .... Operand can be a component of a trigger condition

O:.................... Operand can be overwritten

Operand	Designation	P	T	O
Input, bit	IX	x	x	x
Input, byte	IB	x	x	x
Input, word	IW	x	x	x
Input, double word	ID	x	x	x
Output, bit	QX	x	x	x
Output, byte	QB	x	x	x
Output, word	QW	x	x	x
Output, double word	QD	x	x	x
Marker, bit	MX	x	x	x
Marker, byte	MB	x	x	x
Marker, word	MW	x	x	x
Marker, double word	MD	x	x	x
Marker, real	MR	x		x
Marker, long real	MLR	x		x
System marker, bit	MSX	x	x	x
System marker, byte	MSB	x	x	x
System marker, word	MSW	x	x	x
System marker, double word	MSD	x	x	x
Read only system marker, bit	ROSX	x	x	
Read only system marker, byte	ROSB	x	x	
Read only system marker, word	ROSW	x	x	
Read only system marker, double word	ROSD	x	x	
Step marker, bit	$X	x	x	x
Step marker, byte	$B	x	x	x
Parameter, bit	PX			
Parameter, byte	PB			
Parameter, word	PW			
Parameter, double word	PD			
Parameter, real	PR			
Parameter, long real	PLR			

Operand	Designation	P	T	O
Data block, bit	DX	x	x	x
Data block, byte	DB	x	x	x
Data block, Text	DT	x	x	x
Data block, word	DW	x	x	x
Data block, double word	DD	x	x	x
Data block, real	DR	x		x
Data block, long real	DLR	x		x
Address marker, byte	MAB	x	x	x
Address marker, word	MAW	x	x	x
Address marker, double word	MAD	x	x	x
Decimal constant, byte, word, double word	#, #W, #D			
Hexadecimal constant, byte, word, double word	#H, #WH, #DH			
Hexadecimal constant, byte, signed	#HS			
Hexadecimal constant, word, signed	#WHS			
Hexadecimal constant, double word, signed	#DHS			
Binary constant, bit, byte	#X, #B			
Constant, real, long real	#R, #LR			
Constant, PI	#PI			
Constant, e	#E			
Software timer, bit	T	x	x	x
Software timer, word	TW	x	x	
Software counter, bit	C	x	x	
Software counter, word	CW	x	x	x
Edge operand, bit	EDG	x	x	
System bus (peripherals), bit	SX	x	x	x
System bus (peripherals), byte	SB	x	x	x
System bus (peripherals), word	SW	x	x	x
System bus (peripherals), double word	SD	x	x	x
Peripheral data block, bit	BX	x	x	x
Peripheral data block, byte	BB	x	x	x
Peripheral data block, word	BW	x	x	x
Peripheral data block, double word	BD	x	x	x
Communication (internal), bit	CIX	x	x	x
Communication (internal), byte	CIB	x	x	x
Communication (internal), word	CIW	x	x	x
Communication (internal), double word	CID	x	x	x
Communication (external), bit	CEX	x	x	x
Communication (external), byte	CEB	x	x	x
Communication (external), word	CEW	x	x	x
Communication (external), double word	CED	x	x	x
Network expansion, bit	NX	x	x	x
Network expansion, byte	NB	x	x	x
Network expansion, word	NW	x	x	x
Network expansion, double word	ND	x	x	x
Program label	LAB			
Labels in the SFC editor	LAS			
Subroutine	SP			
Firmware block	FB			
Interrupt Controller Program	IRP			
Interrupt Service Program	ISP			

3.9 Overview of Link Elements

Group:	Link Element Name	Comment
(Common)	:=	Direct assignment bit
	:=BY	Direct assignment byte
	:=DW	Direct assignment double word
	:=FL	Direct assignment float
	:=LFL	Direct assignment long real
	:=WO	Direct assignment word
Arithmetic	ABS	Absolute value
	ABS_B	Absolute value byte
	ABS_D	Absolute value double word
	ACOS_LR	Inverse cosine long real
	ACOS_R	Inverse cosine real
	ADD_B	Adder byte
	ADD_D	Adder double word
	ADD_LR	Adder long real
	ADD_R	Adder real
	ADD_W	Adder word
	ASIN_LR	Inverse sine long real
	ASIN_R	Inverse sine real
	ATANH_LR	Inverse hyperbolic tangent long real
	ATANH_R	Inverse hyperbolic tangent real
	ATAN_LR	Inverse tangent long real
	ATAN_R	Inverse tangent real
	COSH_LR	Hyperbolic cosine long real
	COSH_R	Hyperbolic cosine real
	COS_LR	Cosine long real
	COS_R	Cosine real
	DEC	Decrement
	DIV	Divider
	DIV-D/W	Division double word/word
	DIV_B	Divider byte
	DIV_D	Divider double word
	DIV_LR	Divider long real
	DIV_R	Divider real
	DIV_W	Divider word
	EX-1_LR	e-function 1 long real
	EX-1_R	e-function 1 real
	EXP_LR	e-function long real
	EXP_R	e-function real
	INC	Increment
	JUNC_D	Summation with overflow control
	LG2_LR	Logarithm base 2 long real
	LG2_R	Logarithm base 2 real
	LN+1_LR	Natural logarithm+1 long real
	LN+1_R	Natural logarithm+1 real
	LN_LR	Natural logarithm long real
	LN_R	Natural logarithm real

Group:	Link Element Name	Comment
Arithmetic	LOG_LR	Logarithm base 10 long real
	LOG_R	Logarithm base 10 real
	MOD_B	Modulo byte
	MOD_D	Modulo double word
	MOD_LR	Modulo long real
	MOD_R	Modulo real
	MOD_W	Modulo word
	MUL	Multiplier
	MUL-D/W	Multiplication double word/word
	MUL_B	Multiplier byte
	MUL_D	Multiplier double word
	MUL_LR	Multiplier long real
	MUL_R	Multiplier real
	MUL_W	Multiplier word
	NEG	Negator
	POW10_B	10^ operand byte
	POW10_D	10^ operand double word
	POW10_LR	10^ operand long real
	POW10_R	10^ operand real
	POW10_W	10^ operand word
	POW2_B	2^ operand byte
	POW2_D	2^ operand double word
	POW2_LR	2^ operand long real
	POW2_R	2^ operand real
	POW2_W	2^ operand word
	SINCOS	Sine/cosine
	SINH_LR	hyperbolic sine long real
	SINH_R	hyperbolic sine real
	SIN_LR	Sine long real
	SIN_R	Sine real
	SQRT_LR	Square root long real
	SQRT_R	Square root real
	SUB_B	Subtractor byte
	SUB_D	Subtractor double word
	SUB_LR	Subtractor long real
	SUB_R	Subtractor real
	SUB_W	Subtractor word
	TANH_LR	Hyperbolic tangent long real
	TANH_R	Hyperbolic tangent real
	TAN_LR	Tangent long real
	TAN_R	Tangent real
Boards	IBS-CKCF	Interbus: check configuration
	IBS-INIT	Interbus: initialize and start
	IBS-PCP	INT-S to PCP communication
	IBS-RDCF	Interbus: read configuration
	IBS-RDMS	Interbus: read command
	IBS-RES	Interbus: controlled RESET
	IBS-STRT	Interbus: start ring
	IBS-WAT	Interbus: monitoring

Group:	Link Element Name	Comment
Boards	IBS-WRCM	Interbus: write comment
	IZB-EVAL	Actual value recording
	IZB-INIT	Initialize IZB
	IZB-REFP	Encoder reference point
	MIO-INI	Initialize MIO
	MIO-TINI	MIO: Timer interrupt initialize
	MIO-TVAL	MIO: Timer display
	MIO-XINI	MIO: Bit interrupt initialize
	MIO_B	Output MIO byte input
	MIO_W	Output MIO word input
	MIO_X	Output MIO bit input
Comparators	EQ	Equal to
	GE	Greater than or equal to
	GT	Greater than
	LE	Less than or equal to
	LT	Less than
	NE	Not equal to
Convert	CONV_B	Data type conversion to byte
	CONV_BCD	Data type conversion to BCD
	CONV_BDD	Data type conversion to BCD-D
	CONV_D	Data type conversion to DW
	CONV_LR	Data type conversion to long real
	CONV_R	Data type conversion to real
	CONV_W	Data type conversion to word
	CONV_X	Data type conversion to bit
Counter	C-DB-DEC	Decrement counter DB
	C-DB-GET	Output counter-status
	C-DB-INI	Initialize counter DB
	C-DB-SET	Start counter
	C-DOWN1	Decrementer, edge-sensitive
	C-DOWN2	Decrementer, static
	C-DOWN3	Decrementer, edge-sensitive/static
	C-UP1	Incrementer, edge-sensitive
	C-UP2	Incrementer, static
	C-UP3	Incrementer, edge-sensitive/static
Error	ERROR	Entry from error catalogue
File Operations	CLOSE	Close a file
	OPEN	Open a file
	READ	Read from a file
	SEEK	Set file pointer
	WRITE	Write to a file
Logic	AND_B	Logically AND byte
	AND_D	Logically AND double word
	AND_W	Logically AND word
	AND_X	Logically AND bit
	ASL	Shift arithmetic to left
	ASL-OV_D	ASL with overflow control DW
	ASL_D	Shift arithmetic to left
	ASR	Shift arithmetic to right

Group:	Link Element Name	Comment
Logic	NED	Falling edge
	NOT	Bitwise negation of accumulator
	OR_B	Logically OR byte
	OR_D	Logically OR double word
	OR_W	Logically OR word
	OR_X	Logically OR bit
	PED	Rising edge
	R	Reset bit
	R/S	RS flip-flop S-dominant level-triggered
	R/S-E	RS flip-flop S-dominant edge-triggered
	ROL	Rotate to left
	ROR	Rotate to right
	S	Set bit
	S/R	RS flip-flop R-dominant level-triggered
	S/R-E	RS flip-flop R-dominant edge-triggered
	SHL	Shift to left
	SHL_W	Shift to left word
	SHR	Shift to right
	XOR_B	Logically XOR byte
	XOR_D	Logically XOR double word
	XOR_W	Logically XOR word
	XOR_X	Logically XOR bit
Memory Operations	COPY	Copy memory
	FILL	Set memory
	TCOPY	Copy memory bytewise
Program Operations	CAL	Unconditional memory call
	CALC	Conditional memory call (RLO = 1)
	CALN	Conditional memory call (RLO = 0)
	DEF	Memory definition
	DEF_PA	Definition reference parameter
	DEF_PV	Definition value parameter
	END	Memory end
	ET	Time value for controller run time
	JMP	Unconditional jump
	JMPC	Conditional jump (RLO = 1)
	JMPN	Conditional jump (RLO = 0)
	LABEL	Label definition
	PARA	Transfer reference parameter
	PARV	Transfer value parameter
	RET	Unconditional return
	RETC	Conditional return (RLO = 1)
	RETN	Conditional return (RLO = 0)
Closed-Loop Control	DIFFER_D	Differentiator double word
	EXTR1_D	Extrapolator 1st type double word
	EXTR2_D	Extrapolator 2nd type double word
	EXTR3_D	Extrapolator 3rd type double word
	GETRIEB1	Electronic gearing position
	GETRIEB2	Electronic gearing speed

Group:	Link Element Name	Comment
Closed-Loop Control	GETRIEB3	Electronic gearing position
	GETRIEB4	Electronic gearing speed
	INTEGR_D	Integrator double word
	LIM+/-_B	Symmetric limitation byte
	LIM+/-_D	Symmetric limitation double word
	LIM+/-_W	Symmetric limitation word
	LIM_B	Limitation byte
	LIM_D	Limitation double word
	LIM_W	Limitation word
	MAXMIN_D	Maximum/minimum double word
	MAX_D	Maximum double word
	MIN_D	Minimum double word
	P-REG_D	P controller double word
	PEEK_D	Save maximum/minimum
	PI-PNA_D	PI parameter stand. absolute double word
	PI-PNR_D	PI parameter stand. relative double word
	PI-REG_D	PI controller double word
	POS-DIF1	Position difference calculation double word
	POS-DIF2	Position difference relative to RPM speed
	POS-GEN1	Position generator 1
	POS-GEN2	Position generator 2 offset
	R-GEN1_D	Ramp generator 1 double word
	R-GEN2_D	Ramp generator 2 double word
	R-GEN3_D	Ramp-function generator 1 double word
	R-GEN4_D	Ramp-function generator 2 double word
	T-NORM_D	Time standardization double word
	U-NORM_D	Speed standardization double word
Standard	B-SWITCH	Bus switch
	DELAY_B	Delay byte
	DELAY_D	Delay double word
	DELAY_W	Delay word
	DELAY_X	Delay bit
	LATCH_EB	Latch byte edge triggered
	LATCH_ED	Latch double word edge triggered
	LATCH_EW	Latch word edge triggered
	LATCH_EX	Latch bit edge triggered
	LATCH_TB	Latch byte transparent
	LATCH_TD	Latch double word transparent
	LATCH_TW	Latch word transparent
	LATCH_TX	Latch bit transparent
	SWITCH	Switch
	SWITCH_B	Switch byte
	SWITCH_D	Switch double word
	SWITCH_W	Switch word
	SWITCH_X	Switch bit
	TR-INI_B	Trace initialize byte
	TR-INI_D	Trace initialize double word
	TR-INI_W	Trace initialize word

Group:	Link Element Name	Comment
Standard	TRACE_B	Trace write byte
	TRACE_D	Trace write double word
	TRACE_W	Trace write word
System Function	DB-LOAD	Flash data to DB
	DB-STORE	DB data to flash
	INTR-SET	Set up system interrupt
Timer	K-IP-MS	Constant pulse
	MN-IP-MS	Minimum pulse
	MX-IP-MS	Maximum pulse
	OFDEL-MS	Switch-off delay
	ONDEL-MS	Switch-on delay
	OSDEL-MS	Save switch-on delay
	T-MIN1	Minute timer edge initialization
	T-MIN2	Minute timer static initialization
	T-MS1	Millisecond timer edge initialization
	T-MS2	Millisecond timer static initialization
	T-S1	Second timer edge initialization
	T-S2	Second timer static initialization