# ABB Procontic b

Programmable Control System

<table>
<thead>
<tr>
<th>Planning</th>
<th>8.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software</td>
<td>8.2</td>
</tr>
</tbody>
</table>
System Description

ABB Procontic b
Programmable Control System

Planning

Order number
GATS 1311 04 R2001
replaces
Publication number
D-AT-1923 87 E
Regulations

Regulations concerning the setting up of installations

Apart from the basic "Regulations for the setting up of power units" VDE 0100 and for "The rating of creepage paths and air gaps" VDE 0110 the regulations "The equipment of power units with electrical components" VDE 0160 in connection with VDE 0660, part 500, have to be taken into due consideration. Further attention has to be paid to VDE 0113 in case of the control of working and processing machines. If operating elements are to be arranged near shock-hazard parts with protection against electrical shock, VDE 0106, part 100, is relevant.

The user has to ensure that the units as well as the associated components have to be installed according to these regulations. Respectively valid safety regulations, e.g. regulation for the prevention of accidents and the law concerning technical working material, are valid for machines and units connected as well.

ABB Procontic units have been built according to VDE regulation 0160. The protection against direct touching as demanded by chapter 5.5.1 of this VDE regulation has to be satisfied by the user, e.g. at installing of switch cabinet.

ABB Procontic units have been designed for operation according to insulation class A of VDE 0110. If considerable pollution is expected during operations, the units have to be installed in housings of the respective kind of protection.

*VDE stands for "Association of German Electrical Engineers".

Note: Please observe the national regulations for the installation of electrical equipments, which are valid in your country.

ABB Schalt- und Steuerungstechnik GmbH
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>General Remarks</td>
<td>1-1</td>
</tr>
<tr>
<td>1.1</td>
<td>Address Allocation of the Devices</td>
<td>1-1</td>
</tr>
<tr>
<td>1.2</td>
<td>Address Setting of Units and Bus Connectors</td>
<td>1-2</td>
</tr>
<tr>
<td>1.2.1</td>
<td>07 ET 82 and 07 ET 83</td>
<td>1-2</td>
</tr>
<tr>
<td>1.2.2</td>
<td>07 ET 84</td>
<td>1-2</td>
</tr>
<tr>
<td>1.2.3</td>
<td>07 BV 84 with extended address range</td>
<td>1-2</td>
</tr>
<tr>
<td>1.3</td>
<td>Information on How to Handle PROCONTIC b Devices</td>
<td>1-3</td>
</tr>
<tr>
<td>1.4</td>
<td>The Handling of EPROMs</td>
<td>1-3</td>
</tr>
<tr>
<td>2.</td>
<td>“Little” Systems with Bit Processing only</td>
<td>2-1</td>
</tr>
<tr>
<td>2.1</td>
<td>Possible System Extensions</td>
<td>2-1</td>
</tr>
<tr>
<td>3.</td>
<td>“Medium” Systems with Bit Processing only</td>
<td>3-1</td>
</tr>
<tr>
<td>3.1</td>
<td>Possible System Extensions</td>
<td>3-1</td>
</tr>
<tr>
<td>4.</td>
<td>“Big” Systems with Bit Processing only</td>
<td>4-1</td>
</tr>
<tr>
<td>4.1</td>
<td>Possible System Extensions</td>
<td>4-2</td>
</tr>
<tr>
<td>5.</td>
<td>Systems with Bit and Word Processing</td>
<td>5-1</td>
</tr>
<tr>
<td>5.1</td>
<td>Possible System Extensions</td>
<td>5-1</td>
</tr>
<tr>
<td>6.</td>
<td>Information on the Structure of Switch Cabinets and Wiring</td>
<td>6-1</td>
</tr>
<tr>
<td>6.1</td>
<td>Power Consumption of the Devices</td>
<td>6-2</td>
</tr>
<tr>
<td>6.2</td>
<td>Structure of the Switch Cabinets</td>
<td>6-4</td>
</tr>
<tr>
<td>6.3</td>
<td>Wiring of 0 V and Earthing Connection</td>
<td>6-5</td>
</tr>
<tr>
<td>6.4</td>
<td>PROCONTIC b Surrounding Conditions</td>
<td>6-6</td>
</tr>
<tr>
<td>6.5</td>
<td>Connection of the Starting Unit</td>
<td>6-6</td>
</tr>
<tr>
<td>6.6</td>
<td>Connection of the Programmer</td>
<td>6-6</td>
</tr>
<tr>
<td>6.7</td>
<td>Failure Messages</td>
<td>6-7</td>
</tr>
</tbody>
</table>

This description helps to survey possibilities of how to plan the setting of PROCONTIC b systems. The structure of basic and extended versions of units is described, furthermore the division of the units in various address ranges and the general handling of the system. The description of the system abilities, of the devices and the description of the programming language can be found in the publications:

**PROCONTIC b: System Description**

- Publication 1: General
- Publication 2: Hardware
- Publication 8.2: Software
1 General Remarks

The programmable controller system PROCONTIC b is an automation system which can be harmonized in an ideal way with all needs due to its fine modularity. This applies to the main frames with different capacities with respect to the program storage and a set of instructions adapted to the tasks as well as to various grades of expandability of the operating storage and the complete range of input and output devices.

All devices of the PROCONTIC b system are fully upward compatible, i.e. programmes from "little" systems can be used for "big" systems as well. The control routine can grow with the tasks.

In order to offer a well-aimed help towards the respective most favourable configuration the following systems are divided into four different classes.

The first class:

- little system processing bits with a maximum expandability to 64 resp. 128 I/O channels.
- Size of program storage: maximum of 2 K-program words.

The second class:

- system processing bits and with a maximum expandability to 256 inputs and 256 outputs.
- Size of program storage: maximum of 2 K-program words.

The third class:

- systems processing bits and with a maximum expandability to 768 inputs and 768 outputs (max. 96 slots).
- Size of program storage: maximum of 8 K-program words.

The fourth class:

- systems processing bits and words and with a maximum expandability to 768 inputs and 768 outputs and 128 analog inputs/outputs (max. 96 slots).
- Size of program storage: maximum of 8 K-program words.

1.1 Address Allocations of the devices

In the PROCONTIC b system every I/O slot is precisely allocated to one PROCONTIC b address. This means that if an input/output device is plugged into the unit it gets the address of the respective slot as its group number. Thereby the setting of the address becomes unnecessary. Even in case of an exchange of devices the allocation between the PROCONTIC b address and the device is restored hereby.

Example:

A PROCONTIC b variable (PROCONTIC b address) consists of the following parts:

- channel no., i.e. clamp on the device
- group no., slot of the device
- marks of the variable, i.e. input (E), output (A) etc.

![Diagram](image)

**Figure 1**

Remark:

In case of expandible units being coded with channel numbers > 7 the figure 8 must be added to the channel number (clamp number). E.g. channel no. 00 corresponds to channel no. 08 (cf. chapter 1.2.1).

The clamps of the I/O devices are always numbered from the bottom to the top as far as PROCONTIC b devices are concerned. In case of devices with fewer than 8 channels per device the unoccupied channels cannot be used by the system for other purposes.

Example:

- device mark (type)
  - at the bridge (<7 resp. >7)
  - clamp no. 07 (channel no. 07 resp. 15)
  - clamp no. 06 (channel no. 06 resp. 14)
  - clamp no. 05 (channel no. 05 resp. 13)
  - clamp no. 04 (channel no. 04 resp. 12)
  - clamp no. 03 (channel no. 03 resp. 11)
  - clamp no. 02 (channel no. 02 resp. 10)
  - clamp no. 01 (channel no. 01 resp. 09)
  - clamp no. 00 (channel no. 00 resp. 08)

- +24V connection
- 0V connection
- subtype no. (device variant).
The timing units 07 TI 80 and 07 TI 81 are addressed by the system like the I/O devices, i.e. I/O slots occupied by these devices cannot be occupied by other I/O devices at the same address.

1.2 Address Setting of Units and Bus Connectors

With the PROCONTIC b being able to address a number of inputs and outputs bigger than that of devices to be plugged into a unit it is necessary for the expansibility of the system to several units to allocate an address range (range of variables) to each unit.

For this purpose an address coding field resp. an address coding switch (bus decoder) can be found on the backplane of every unit. The address range of every unit must be set before starting.

Examples for the addresses being set can be found in the chapters 2 to 5. Here the user can only find information on the significance of the single coding possibilities.

1.2.1 07 ET 82 and 07 ET 83

Address coding field on the backplane of the units 07 ET 82 and 07 ET 83 (the address coding field can be found at the I/O slot no. 12 on the 07 ET 83 resp. at the I/O slot no. 05 on the 07 ET 83).

- T - N: If this bridge is plugged, no single timing units can be plugged into this unit. This bridge must be plugged into the 07 TZ 82.
- X - N: If this bridge is plugged, no input devices can be plugged into this unit.
- Y - N: If this bridge is plugged, no output devices can be plugged into this unit.
- <= 7: If this bridge is plugged, all devices of this unit have the channel numbers from 00 to 07.
- > 7: If this bridge is plugged, all devices of this unit have the channel numbers from 08 to 15.

Remark:
Never plug the bridges >7 and <=7 simultaneously, the same applies to the bridges X-N and Y-N. In case of >7 or <=7 just exactly one bridge may be plugged.

1.2.2 07 ET 84

Address coding switch on the backplane of the unit 07 ET 84.

This address coding switch can be found at the system slot 1. It is provided with a protective cover which can be lifted up from the bottom to the top. The switches are numbered from 1 to 8.

Significance of the switches

<table>
<thead>
<tr>
<th>S</th>
<th>function</th>
<th>ON</th>
<th>OFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>binary inputs</td>
<td>free</td>
<td>blocked</td>
</tr>
<tr>
<td>2</td>
<td>binary outputs</td>
<td>free</td>
<td>blocked</td>
</tr>
<tr>
<td>3</td>
<td>timings</td>
<td>free</td>
<td>blocked</td>
</tr>
<tr>
<td>4</td>
<td>input of words</td>
<td>free</td>
<td>blocked</td>
</tr>
<tr>
<td>5</td>
<td>output of words</td>
<td>free</td>
<td>blocked</td>
</tr>
<tr>
<td>6</td>
<td>channels 0 - 7</td>
<td>blocked</td>
<td>free</td>
</tr>
<tr>
<td>7</td>
<td>channels 8 - 15</td>
<td>blocked</td>
<td>free</td>
</tr>
<tr>
<td>8</td>
<td>vacant</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Remark:
The switch S3 must at position OFF on the digital timing unit 07 TZ 82. The switches S6 and S7 must not be positioned at OFF simultaneously.

1.2.3 07 BV 84 with Extended Address Range

If the extended PROCONTIC address range is used for the I/O addresses from I/O 16,00 to I/O 63,15, the respective address range must be set on the bus connectors of the expanded units. The setting of the bus connector 07 BV 84 applies to the unit it is plugged into (cf. pages 3.1 and 4.1). The extended I/O address range cannot be addressed without bus driver 07 BT 84 and bus connector 07 BV 84.

<table>
<thead>
<tr>
<th>bridge</th>
<th>address range</th>
<th>device addresses</th>
</tr>
</thead>
<tbody>
<tr>
<td>A - B*</td>
<td>group no. 00..15</td>
<td>I/O 00,00 .. I/O 15,15</td>
</tr>
<tr>
<td>A - D</td>
<td>group no. 16..31</td>
<td>I/O 16,00 .. I/O 31,15</td>
</tr>
<tr>
<td>A - C</td>
<td>group no. 32..47</td>
<td>I/O 32,00 .. I/O 47,15</td>
</tr>
<tr>
<td>A - E</td>
<td>group no. 48..63</td>
<td>I/O 48,00 .. I/O 63,15</td>
</tr>
</tbody>
</table>

* = state of delivery.
1.3 Information on How to Handle PROCONTIC b Devices

All PROCONTIC b devices are provided with CMOS components. These elements are sensitive with respect to static discharges. Therefore, the respective protective measures have to be regarded, i.e. e.g. PROCONTIC b devices must be treated carefully when being unpacked, they shall only be touched at the edge of the circuit board resp. the front panel. For transport reasons they must be put back in their packings immediately.

It is forbidden to plug PROCONTIC b devices in or off when they are alive because of a possible destruction of the CMOS elements.

It is therefore sensible to have a switch well attainable in the switch cabinet by which the supply voltage can be switched off.

1.4 Treatment of EPROMs

The program storage in form of an EPROM is a central unit of a PROCONTIC b because of the user program stored there. Therefore a careful treating of the program storage is essential. It may be touched at the protective cap only. The program storages 07 PR 82 and 07 PR 84 must not be plugged in or off when being alive. In order to erase the storage it must not be detached from the protective cap, but the window type cover provided for this purpose must be removed. Furthermore, attention must be paid to the storage being erased sufficiently to guarantee an orderly programming.

(Doubling of the time needed for erasure according to the information by the manufacturer for the erase chamber). Attention must also be paid to using storages manufactured by BBC only because they guarantee a good programming and erase ability.
Little systems with bit processing only are restricted to one unit.
Either 8 I/O devices (07 ET 82) or 16 I/O devices (07 ET 83) can be plugged into these units. So the following PROCONTIC addresses are allocated to the I/O devices (the values in brackets apply to the 07 ET 82).

\[
\begin{align*}
E \ 00,00 & \ldots \ E \ 00,07 \ to \ E \ 15,00 \ldots E \ 15,07 \\
( \ E \ 00,00 & \ldots \ E \ 00,07 \ to \ E \ 07,00 \ldots E \ 07,07 ) \\
or
A \ 00,00 & \ldots A \ 00,07 \ to \ A \ 15,00 \ldots A \ 15,07 \\
( \ A \ 00,00 & \ldots A \ 00,07 \ to \ A \ 07,00 \ldots A \ 07,07 ) \\
or
T \ 00,00 & \ldots T \ 00,03 \ to \ T \ 03,00 \ldots T \ 03,03 \\
( \ T \ 00,00 & \ldots T \ 00,03 \ to \ T \ 03,00 \ldots T \ 03,03 )
\end{align*}
\]

Example:

The following elements are necessary:
1. power supply unit  \( 1 \times \ 07 \text{NG80} \)
2. main frame \( 1 \times \ 07 \text{ZE 82} \)
3. blank covers \( 2 \times \ 07 \text{BA 80} \)
4. analog timings \( 1 \times \ 07 \text{TI 80} \)
5. binary inputs \( 4 \times \ 07 \text{XS 80} \)
6. binary outputs \( 3 \times \ 07 \text{YS 80} \)

This shows that a unit with 8 I/O slots is sufficient (07 ET 82).

![figure 2](image_url)

Remark: The address coding socket is at slot 05 on the 07 ET 82.

In a PROCONTIC b system with one unit the bus decoder is always provided with the bridge <=7. All other bridges remain open.

In this configuration the following addresses can be found:

\[
\begin{align*}
T \ 00,00 & \ldots T \ 00,03 \\
E \ 01,00 & \ldots E \ 01,07 \ to \ E \ 04,00 \ldots E \ 04,07 \\
A \ 05,00 & \ldots A \ 05,07 \ to \ A \ 07,07 \ldots A \ 07,07 \\
M \ 00,00 & \ldots M \ 00,15 \ to \ M \ 07,00 \ldots M \ 07,15 \\
(128 markers on the 07 ZE 82).
\end{align*}
\]

2.1 Possible System Extensions

07 RK 80

If additional storage, e.g. step chains, is needed for such a system apart from the 128 markers existing on the main frame 07 ZE 82, this system can be extended by means of the additional storage 07 RK 80. This means the additional availability of 8 step chains with 16 steps each. The additional storage is zero-voltage-proof. Furthermore, there are 64 output maps (A 08,08 ... A 15,15) fit for use as buffered markers on the card. The device 07 RK 80 is plugged into the system slot IV or V and thus does not occupy an I/O slot.

07 AS 82

The operating storage 07 AS 82 helps to extend the marker range of the main frame. The user then has 1024 markers and 64 step chains with 16 steps each available.

\[
\begin{align*}
M^* \ 00,00 & \ldots M^* \ 63,15 \ as \ binary \ markers \\
S^* \ 00,00 & \ldots S^* \ 63,15 \ as \ step \ chains
\end{align*}
\]

All markers and step chains can be buffered with a battery. If the operating storage is used in this configuration, the bridge 1001 must be removed from the main frame 07 ZE 82 because of all markers being on the operating storage. The device is plugged into the system slot IV resp. V. Furthermore, the use of the stronger power supply unit 07 NG 82 is necessary.

07 TZ 82

If the timing and counting unit 07 TZ 82 is applied in a system, the bridge T-N must be plugged additionally onto the bus decoder.
The user then has 60 timings, 16 counters (R201 only) and a serial interface for the editing of the text available. A detailed description of this device can be found in the PROCONTIC b system description. The device is plugged into the slots IV,V,00 resp. V,00,01. Furthermore, the use of the stronger power supply unit 07 NG 82 is necessary.

07 AE 83

By applying the combined I/O device 07 AE 83 the number of the I/O channels can be extended to a maximum of 64 inputs and 64 outputs with the 07 ET 82 resp. to a maximum of 128 inputs and 128 outputs with the 07 ET 83.
Medium systems with bit processing only can be extended to a maximum of 4 units. Systems with the 07 ZE 82 must always be extended with the bus driver 07 BT 82 R3, In case of "older" equipment with 07 BT 82 R1, R2 the 07 BV 82 must be ordered as a replacement.

So the following PROCONTC b addresses are to be found in this configuration:

- **Basic unit:**
  - T 00,00 to T 02,00
  - E 03,07 to E 15,07

- **1st extension unit:**
  - E 00,08 to E 02,15
  - A 03,15 to A 13,15

- **128 markers on the 07 ZE 82:**
  - M 00,00 to M 07,15

### 3.1 Possible System Extensions

#### 07 RK 80
If additional storage, e.g., step chains, is needed for such a system apart from the 128 markers being on the main frame 07 ZE 82, this system can be extended by means of the additional storage 07 RK 80. In that case additional 8 step chains with 16 steps each are available. The additional storage is zero-voltage-proof.

Furthermore, there are 64 output maps (A 08,08 ... A 15,15) fit for use as buffered markers on the card. In order to use this device the bridge 1001 on the bus driver 07 BT 92 must be connected by all means.

The device 07 RK 80 is plugged into the system slot IV or V and thus does not occupy an I/O slot.

**Attention**
The outputs A 08,08 ... A 15,15 have buffered maps which - as the case may be - must be erased or allocated in the first cycle, if they also exist as output devices. If this range is used as marker, the configuration of the equipment should be of that kind that the output devices are allocated in the basic units, the inputs in the extension units.

#### 07 TZ 82
If the timing and counting system 07 TZ 82 is applied, the bridge T-N must be additionally plugged in both units (cf. 1.2.1). The user then has 60 timings, 16 counters (R201 only) and a serial interface for the editing of the text available. A precise description of this device can be found in the PROCONTC b system description.

The device is plugged into the slots IV, V, 00 resp. V, 00,01. Furthermore, the use of the stronger power supply unit 07 NG 82 is necessary.
07 AS 82
The marker range of the main frame can be extended by means of the operating storage 07 AS 82. The user then has 1024 markers and 64 step chains with 16 steps each available.

M' 00,00 ...... M' 63,15  as binary markers
S' 00,00 ...... S' 63,15  as step chains

All markers and step chains can be buffered with a battery. If the operating storage is applied in this configuration, the bridge 1001 must be removed from the main frame 07 ZE 82 because of all markers being on the operating storage. Furthermore, the bridge 1002 has to be removed from the bus driver 07 BT 82. The device is plugged into the system slot IV resp. V. Furthermore, the use of the stronger power supply 07 NG 82 is necessary.

07 AE 83
By using the combined I/O device 07 AE 83 the number of the I/O channels can be extended to a maximum of 256 inputs and 256 outputs.

Extension to 4 units
If the system is to be extended to 4 units (basic unit and 3 extension units), each defined unit must be released for precisely one type of device.

Example:

basic unit:
only output devices: channel no. 00 ... 07
and timing units: channel no. 00 ... 03
(bridge X-N and bridge <=7 plugged)

1st extension unit:
only output units: channel no. 08 ... 15
and timing units: channel no. 08 ... 11
(bridge X-N and bridge >7 plugged)

2nd extension unit:
only input devices: channel no. 00 ... 07
(bridges X-N, T-N and <=7 plugged)

3rd extension unit:
only input units: channel no. 08 ... 15
(bridges Y-N, T-N and >7 plugged).
All PROCON 1IC b systems working with 07 ZE 84 have the extended address range to E 63,15 resp. A 53,15 at their disposal. They can be extended to a size of 6 units (maximum). The use of the operating storage 07 AS 82 and the stronger power supply unit 07 NG 82 is by all means necessary. According to the extension it can be necessary to parallel 2 power supply units 07 NG 82. The system description informs about the power consumption. The following PROCON 1IC addresses are allocated to the I/O devices:

E 00,00......E 00,15 to E 63,00......E 63,15
A 00,00......A 00,15 to A 63,00......A 63,15
T 00,00......T 00,03 to T 03,00......T 03,03
T 00,08......T 00,11 to T 03,08......T 03,11

Remark:
There is no extended address range for the timing units. In systems with the unit 07 ZE 84 the pair 07 BT 84 and 07 BV 84 can always connect the basic unit with the extension units. If nevertheless the bus driver 07 BT 82 is used in the basic unit, the bridge 1002 must be removed there. For this kind of extension the address range for I/O devices remains restricted to the maximum group number 15,...

Example:

The following elements are needed:
- 2 power supply units 2 x 07 NG 82
- 1 main frame 1 x 07 ZE 84
- 1 operating storage 1 x 07 AS 82
- 1 bus driver 1 x 07 BT 84
- 4 bus connectors 4 x 07 BV 84
- 21 blank covers 21 x 07 BA 80
- 320 binary inputs 40 x 07 XS 80
- 240 binary outputs 30 x 07 YS 80
- 40 timings 1 x 07 TZ 82

This means that 5 units with a total of 80 I/O slots are necessary (5 x 07 ET 83).

Remark:
In case of using the timing and counting unit 07 TZ 82 it is necessary to plug the bridge T-N additionally onto all units.

Figure 4

Remark: the address coding socket is at slot 1 (with the 07 ET 83).
In this configuration the following PROCONTIC b addresses are allocated to the devices:

timing unit 07 TZ 82:
  T 00,00 ...... T 00,15 to T 03,00 ...... T 03,11

basic unit:
  E 03,00 ...... E 03,07 to E 15,00 ...... E 15,07

1st extension unit:
  E 00,08 ...... E 00,15 to E 15,08 ...... E 15,15

2nd extension unit:
  E 16,00 ...... E 16,07 to E 26,08 ...... E 26,07

3rd extension unit:
  A 16,08 ...... A 16,15 to A 31,08 ...... A 31,15

4th extension unit:
  A 32,00 ...... A 32,07 to A 45,08 ...... A 45,07

global markers:
  M' 00,00 ...... M' 00,15 to M' 63,00 ...... M' 63,15

local markers:
  M 00,00 ...... M 00,15 to M 07,00 ...... M 07,15

global steps:
  S' 00,00 ...... S' 00,15 to S' 6300 ...... S' 63,15

local steps:
  S 00,00 ...... S 00,15 to S 07,00 ...... S 07,15

4.1 Possible System Extensions

07 RK 80
An extension with 07 RK 80 is impossible. All internal storages which can be addressed are in the unit 07 AS 82.

07 AE 83
In case of an extension with 5 extension units the number of the I/O channels can be extended to a maximum of 768 inputs and 768 outputs if the combined I/O device 07 AE 83 is used.

07 WP 84
If arithmetic operations are needed in the PROCONTIC b system, an extension by the word processor 07 WP 84 is possible. The word inputs and outputs serve as the binary I/O devices. Using the analog I/O devices in systems with the unit 07 ET 83 is impossible.
5 Systems with Bit and Word Processing

A PROCONTIC b system with bit and word processing always consists of the following main frames:

07 NG 82 R3
07 ZE 84 R1
07 AS 82 R1
07 WP 84 R202

As long as peripheral devices are only used as binary I/O devices, the same structure with the 07 ET 83 can be used as described in chapter 4.

For the use of analog I/O devices the unit 07 ET 84 is always prescribed.

Analog devices are always double devices occupying 2 slots. Hereby the left part of the device must always be in a slot with even numbers, e.g. slots 00, 02, 04, 06, 08, 10, 12 and 14.

When using analog devices they are always connected with the extensions units by means of the bus driver 07 BT 84.

Example:

50 timings: 1 x 07 TZ 82
8 analog inputs 12 bit: 1 x 07 EA 81
8 analog inputs 8 bit: 1 x 07 EA 80
4 analog outputs 12 bit: 2 x 07 AA 81
80 binary inputs: 8 x 07 AE 83 and
2 x 07 XS 80
60 binary outputs:
are contained in 8 x 07 AE 83

This means that 2 units with a total of 32 I/O slots are needed (1 x 07 ET 84 and 1 x 07 ET 83)

So the following PROCONTIC addresses are allocated to the I/O devices (cf. figure 5):

EA 06,00 ... EA 06,07 to EA 08,00 ... EA 08,07
AA 10,00 ... AA 10,01 to AA 12,00 ... AA 12,01
E 00,00 ... E 00,07 to E 09,00 ... E 09,07
A 00,00 ... A 00,07 to A 07,00 ... A 07,07

Remark: with the 07 ET 83 the address coding socket is at slot 12. With the 07 ET 84 the address coding switch is at slot 1.

5.1 Possible System Extensions

Extension of the word periphery

The extension of the word periphery takes place according to the same procedure as the extension of the binary inputs/outputs. An extension to 4 units is the maximum. Hereby a division of the units must take place to get one type of device (division into analog input and analog output). A mixed setting of word and bit devices on the unit 07 ET 84 is possible as well.

Extension of the binary inputs/outputs

The extension of the binary inputs/outputs has already been described in chapter 4.1.
Proposals for the structure and the correct wiring of a PROCONTIC b system are made on the next pages. There is only a minimum of requirements concerning current supply and the cabling. To guarantee a function of the system without problems attention has to be paid to the proposals made here.

To be able to leak off disturbances of high frequency better only use flexible wiring for the current supply. Connect the “zero” of the PROCONTIC supply and the earthing connections of the PROCONTIC casings at a central earthing point (cf. chapters 6.2 and 6.3).

Provide the switch cabinet with at least 2 plug sockets of 220 V for the connection of a programmer and a printer. Connect the protective conductor with the central earthing point in the switch cabinet. Power supply units for the supply of the PROCONTIC b meeting the requirements of the system can be found in the system description.

Sufficient dimensions of the section of the lead in are part of a good wiring. Several parallel sources for the leading in of the supply voltage and not its leading through a complete unit are especially necessary for the power outputs. In case of output devices it is not only the connection of the positive supply voltage that has to be regarded, because all PROCONTIC b outputs are provided with a demagnetization switch for inductive charges leading off the demagnetization current towards “zero” volt (for details concerning the demagnetization switches cf. the system description).

Therefore the “zero” volt lead in is provided with a respective section. In case of units equipped with input devices only no 24 V supply needs leading through, because the input devices 07 XS 80, 07 XS 86 and 07 XS 87 only need the “zero” volt connection for their functions. All input currents and all interference currents run via this connection which again makes a sufficient section necessary.

The following regulations have to be regarded by the user:

VDE 0100:
VDE 0110, 0160, 0660: setting up of switch plants for switch devices with electrical components
VDE 0113: for the control of working and processing machines

Supply Voltage
The supply voltage for the PROCONTIC b is 24 V +/−30%. The transformer of the power supply unit must be provided with a screen winding (earthing on protective conductor SE with flexible wiring of 6 mm). The supporting condenser must be laid out that way that the ripple at full load does not exceed 3 V. To suppress interferences a HF anti-interference condenser of 0.1 µF and a parallel resistance of 3.3 kΩ are switched between the positive pole and the negative pole of the power supply unit.

The 24 V - supply voltage to the PROCONTIC b must be secured by a smart fuse, e.g. the power switches manufactured by ABB, according to the power consumption plus a reserve of some 20%. The linkage of 24 V - power supply units with control and short circuit protection may lead to unwanted building-up processes (possible error source). Suitable power supply units can be found in the PROCONTIC b system description.
6.1 Power Consumption of the Devices

The following tables inform about the power consumption of the various devices. Hereby the power consumption from 24 V refers to the lead in of the voltage at the front side. The power consumption from VDD is taken via the PROCONTIC bus from the power supply units 07 NG 80 and 07 NG 82.

<table>
<thead>
<tr>
<th>devices</th>
<th>intern</th>
<th>extern</th>
<th>U₁₀</th>
</tr>
</thead>
<tbody>
<tr>
<td>07 NG 80</td>
<td>mA</td>
<td>150 mA</td>
<td>mA</td>
</tr>
<tr>
<td>07 NG 82</td>
<td>mA</td>
<td>600 mA</td>
<td>mA</td>
</tr>
<tr>
<td>07 ZE 82</td>
<td>mA</td>
<td>mA</td>
<td>100 mA</td>
</tr>
<tr>
<td>07 ZE 84</td>
<td>mA</td>
<td>mA</td>
<td>290 mA</td>
</tr>
<tr>
<td>07 ZE 86</td>
<td>mA</td>
<td>500 mA</td>
<td>0 mA</td>
</tr>
<tr>
<td>07 RK 80</td>
<td>mA</td>
<td>mA</td>
<td>12 mA</td>
</tr>
<tr>
<td>07 AS 82</td>
<td>mA</td>
<td>mA</td>
<td>500 mA</td>
</tr>
<tr>
<td>07 BT 82</td>
<td>mA</td>
<td>mA</td>
<td>25 mA</td>
</tr>
<tr>
<td>07 BT 84</td>
<td>mA</td>
<td>mA</td>
<td>50 mA</td>
</tr>
<tr>
<td>07 BV 84</td>
<td>mA</td>
<td>mA</td>
<td>10 mA</td>
</tr>
<tr>
<td>07 TI 80</td>
<td>mA</td>
<td>mA</td>
<td>10 mA</td>
</tr>
<tr>
<td>07 TI 81</td>
<td>mA</td>
<td>mA</td>
<td>10 mA</td>
</tr>
<tr>
<td>07 TZ 82</td>
<td>mA</td>
<td>mA</td>
<td>400 mA</td>
</tr>
<tr>
<td>07 WP 84</td>
<td>mA</td>
<td>mA</td>
<td>400 mA</td>
</tr>
<tr>
<td>07 ZG 84</td>
<td>mA</td>
<td>600 mA</td>
<td>30 mA</td>
</tr>
<tr>
<td>07 XS 80</td>
<td>mA</td>
<td>ca. 40 mA</td>
<td>1 mA</td>
</tr>
<tr>
<td>07 XS 81</td>
<td>mA</td>
<td>45-55 mA</td>
<td>1 mA</td>
</tr>
<tr>
<td>07 XS 86</td>
<td>mA</td>
<td>ca. 40 mA</td>
<td>1 mA</td>
</tr>
<tr>
<td>07 XS 87</td>
<td>mA</td>
<td>mA</td>
<td>1 mA</td>
</tr>
<tr>
<td>07 YS 80</td>
<td>mA</td>
<td>dependent on load</td>
<td>5 mA</td>
</tr>
<tr>
<td>07 YS 81</td>
<td>mA</td>
<td>dependent on load</td>
<td>1 mA</td>
</tr>
<tr>
<td>07 YS 82</td>
<td>mA</td>
<td>dependent on load</td>
<td>1 mA</td>
</tr>
<tr>
<td>07 YS 86</td>
<td>mA</td>
<td>100 mA</td>
<td>5 mA</td>
</tr>
<tr>
<td>07 AB 83</td>
<td>mA</td>
<td>dependent on load</td>
<td>1 mA</td>
</tr>
<tr>
<td>07 AE 83</td>
<td>mA</td>
<td>dependent on load</td>
<td>1 mA</td>
</tr>
<tr>
<td>07 EA 80</td>
<td>mA</td>
<td>100 mA</td>
<td>40 mA</td>
</tr>
<tr>
<td>07 EA 81</td>
<td>mA</td>
<td>110 mA</td>
<td>40 mA</td>
</tr>
<tr>
<td>07 AA 80</td>
<td>mA</td>
<td>100 mA</td>
<td>16 mA</td>
</tr>
<tr>
<td>07 AA 81</td>
<td>mA</td>
<td>100 mA</td>
<td>16 mA</td>
</tr>
<tr>
<td>07 AG 80</td>
<td>mA</td>
<td>mA</td>
<td>43 mA</td>
</tr>
<tr>
<td>07 AG 81</td>
<td>mA</td>
<td>mA</td>
<td>43 mA</td>
</tr>
<tr>
<td>07 IE 84</td>
<td>mA</td>
<td>700 mA</td>
<td>mA</td>
</tr>
</tbody>
</table>

* power from $U_e = 8.5 \text{ mA}$ with 24 V / 17 mA with 48 V

power from $U_a = 11 \text{ mA}$ with 24 V / 22 mA with 48 V

+ output load at K11/K12
### Power supply units available for the PROCONTIC logic level:

<table>
<thead>
<tr>
<th>Device</th>
<th>TB*</th>
<th>Position number in</th>
<th>07 ET 84</th>
<th>07 ET 83</th>
<th>07 ET 82</th>
<th>Pieces</th>
<th>I_{ma}</th>
<th>Total I_{ma}</th>
</tr>
</thead>
<tbody>
<tr>
<td>07ZE 82</td>
<td>2</td>
<td>II - III</td>
<td>I - III</td>
<td>II - III</td>
<td>II - III</td>
<td>100 mA</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>07ZE 84</td>
<td>3</td>
<td>II - IV</td>
<td>II - IV</td>
<td>II - IV</td>
<td>II - IV</td>
<td>290 mA</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>07ZE 86</td>
<td>4</td>
<td>II - V</td>
<td>II - V</td>
<td>II - V</td>
<td>II - V</td>
<td>0 mA</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>07RK 80</td>
<td>1</td>
<td>IV - 15</td>
<td>IV - 15</td>
<td>IV - 02</td>
<td>IV - 02</td>
<td>12 mA</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>07AS 82</td>
<td>1</td>
<td>V - 15</td>
<td>IV - 09</td>
<td>IV - 02</td>
<td>IV - 02</td>
<td>500 mA</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>07BT 82</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>07BT 84</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>07BV 84</td>
<td>1</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>50 mA</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>07TI 80</td>
<td>1</td>
<td>00 - 03</td>
<td>00 - 03</td>
<td>00 - 03</td>
<td>00 - 03</td>
<td>10 mA</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>07TI 81</td>
<td>1</td>
<td>00 - 03</td>
<td>00 - 03</td>
<td>00 - 02</td>
<td>00 - 02</td>
<td>10 mA</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>07TZ 82</td>
<td>3</td>
<td>IV - 15</td>
<td>IV - 09</td>
<td>IV - 02</td>
<td>IV - 02</td>
<td>400 mA</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>07WP 84</td>
<td>3</td>
<td>IV - 15</td>
<td>IV - 09</td>
<td></td>
<td></td>
<td>400 mA</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>07ZG 84</td>
<td>2</td>
<td>00 - 15</td>
<td>00 - 15</td>
<td>00 - 07</td>
<td></td>
<td>30 mA</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>07XS 80</td>
<td>1</td>
<td>00 - 15</td>
<td>00 - 15</td>
<td>00 - 07</td>
<td></td>
<td>1 mA</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>07XS 81</td>
<td>1</td>
<td>00 - 15</td>
<td>00 - 15</td>
<td>00 - 07</td>
<td></td>
<td>1 mA</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>07XS 86</td>
<td>1</td>
<td>00 - 15</td>
<td>00 - 15</td>
<td>00 - 07</td>
<td></td>
<td>1 mA</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>07XS 87</td>
<td>1</td>
<td>00 - 15</td>
<td>00 - 15</td>
<td>00 - 07</td>
<td></td>
<td>1 mA</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>07YS 80</td>
<td>1</td>
<td>00 - 15</td>
<td>00 - 15</td>
<td>00 - 07</td>
<td></td>
<td>5 mA</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>07YS 81</td>
<td>1</td>
<td>00 - 15</td>
<td>00 - 15</td>
<td>00 - 07</td>
<td></td>
<td>1 mA</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>07YS 82</td>
<td>1</td>
<td>00 - 15</td>
<td>00 - 15</td>
<td>00 - 07</td>
<td></td>
<td>1 mA</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>07YS 86</td>
<td>1</td>
<td>00 - 15</td>
<td>00 - 15</td>
<td>00 - 07</td>
<td></td>
<td>5 mA</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>07YS 87</td>
<td>1</td>
<td>00 - 15</td>
<td>00 - 15</td>
<td>00 - 07</td>
<td></td>
<td>1 mA</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>07AE 83</td>
<td>1</td>
<td>00 - 15</td>
<td>00 - 15</td>
<td>00 - 07</td>
<td></td>
<td>1 mA</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>07EA 80</td>
<td>2</td>
<td>00 - 15</td>
<td></td>
<td></td>
<td></td>
<td>40 mA</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>07EA 81</td>
<td>2</td>
<td>00 - 15</td>
<td></td>
<td></td>
<td></td>
<td>40 mA</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>07AA 80</td>
<td>2</td>
<td>00 - 15</td>
<td></td>
<td></td>
<td></td>
<td>10 mA</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>07AA 81</td>
<td>2</td>
<td>00 - 15</td>
<td></td>
<td></td>
<td></td>
<td>10 mA</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>07AG 80</td>
<td>2</td>
<td>00 - 15</td>
<td>00 - 07</td>
<td></td>
<td></td>
<td>43 mA</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>07AG 81</td>
<td>2</td>
<td>00 - 15</td>
<td>00 - 07</td>
<td></td>
<td></td>
<td>43 mA</td>
<td>mA</td>
<td></td>
</tr>
</tbody>
</table>

*TB = division width

Total power I_{ma} from 10 V: ______ mA
cable channel for low voltage (60 V), inputs, outputs not mechanically switched, power supply: 24 V - rate and PROCONTIC

screen BT/BV 4 mm

terminal strip

mounting plate for magnetic switches, magnetic valves, etc.

I/O supply 2 x 1.5 mm² each

Sigma-A: static, static, output devices

Sigma switch. power levels

Sigma switchpower supply unit 24 V PROCONTIC

power supply unit 24 V PROCONTIC

Power supply unit 24 V PROCONTIC supply

2.1.5 mm² PROCONTIC power supply at PSU

0 V earthing

24 V PROCONTIC

0 V earthing

Sigma-A: static, static, output devices

Sigma switch, power levels

Sigma switch, power supply unit 24 V PROCONTIC

Power supply unit 24 V PROCONTIC

0 V earthing

24 V PROCONTIC

0 V earthing

level 1

level 2

level 3

level 4

screen IE-cable

cable channel for mechanically plugged rates (relays, switches) and protective conductor

earthing of casing (6 mm²)

earthing of casing (6 mm²)

6.2 Structure of Switch Cabinet
6.3 Wiring of 0 V and Earthing Connection

0 V for logic and power

- BUS connector 07 BV 84 (4 mm²)
- BUS connector 07 BV 84 (4 mm²)
- BUS driver 07 BT 82 or 07 BT 84 (4 mm²)
- 0 V clamp of 07 TZ 82 resp. 07 WP 84 (1.5 mm²)
- 07 NG 80 or 07 NG 82 (2 x 1.5 mm²)
- PROCONTIC b input devices 07 XS ... (2 x 1.5 mm²)
- external logic, e.g. SIGMA®-tronic
- process adaptation of the input signals, e.g. SIGMA®-tronic
- test and programming unit 07 IE 82/64 (2 x 1.5 mm²)

24 V

- PROCONTIC b output device 07 YS ... (2 x 1.5 mm²)
- lights
- power switching steps, e.g. SIGMA®-tronic
- ZB 10: BL output at the coupler (1.5 mm²)
- ZB 10: only in case of a triaxial cable: device connection SRE (2 x 1.5 mm²)*

Protective conductor and screens

- 6 mm²

- casing and screen of the power supply unit for Procontic b and logic
- casing and screen of the power supply unit for power
- ZB 10: in case of coaxial cable: device connection SRE (2 x 1.5 mm²)*
- in case of triaxial cable: external screen (4 mm²)*

- casing
- protective screen SL
- protective screen SL and screen for external devices
- protective screen SL for plug sockets
- earthing connection of the PROCONTIC b levels (6 mm²)

*remark: singular earthing per bus each, e.g. via an additional BNC-T-element (at ZB 10 bus controller)
6.4 PROCONTIC b Surrounding Conditions

The PROCONTIC b is a sturdily built, modular system which can be used in a lot of spheres and applications. There are only minimum requirements to be met concerning the respective surrounding conditions. But they have to be met in any case.

Range of temperature

The range of temperature for the use of the PROCONTIC b is between 0°C and +55°C. This is the temperature measured as the entry air temperature below the unit. This means the use of the device is of no problem in rooms where people work. If the control is to be used in unheated rooms or even in open air, a switch cabinet heating is to be provided apart from a respective protection for the cabinet.

Humidity and dirt

The PROCONTIC b meets the requirements of the VDE 0110 insulation group A. Humidity class F according to DIN (= German Industrial Standards) 40040 mentioned among the technical data means that the control operates up to a humidity of 95%, but covering with dew must not occur (humidity class F: 95% relative humidity at 25°C on 30 days a year, 75% relative humidity at 21°C, no covering with dew). This, however, is only guaranteed if changes of temperature in the switch cabinet remain little. If the control is operated at a high humidity and sinking temperatures can be expected, an additional switch cabinet heating has to be installed which operates permanently. Furthermore, attention must be paid to the possible formation of conductive deposits in the control in surroundings having a lot of dust in the air. They might lead to a breakdown of the system. In such surroundings a switch cabinet with the respective protection has to be used (we recommend protection IP 54 or better).

Furthermore, all slots not in operation should be protected by means of the blank cover 07 BA 80 in order to prevent dust from intruding.

Other operating conditions

The PROCONTIC b casing is built according to the protection IP 10 (DIN 40050), i.e. that the casing must be protected against eventual touches by unauthorized personnel. Installing it in a switch cabinet is by all means recommendable.

Mechanical strains

The PROCONTIC b is fit for stationary operation. If fastened to hat section track it fulfills the requirements according to VDE 0160/DIN 57160.

If a higher mechanical strain is to be expected at the place of operation, the assembly angles 07 MW 80 have to be provided for fastening.

6.5 Connection of the test and programming unit

The system cable 07 SK 85 serves for the connection of the test and programming unit 07 IE 84 with the PROCONTIC b control. If the cable is connected with an operating control, attention must be paid to

- the starting unit being connected with the same voltage supply as the PROCONTIC.
- the starting unit being inactive. The switch on the device 07 ZE 84 must be in position "OFF".
- the two screen conductors being connected to "0 V" before the plugging of the system cable.
- the programmer being plugged into a plug socket whose protective conductor is connected with the earthing point of the PROCONTIC b (switch cabinet plug socket).

6.6 Connection of the test and programming unit

The test unit 07 IE 84 has two interfaces for the connection of the programming units. The connection via a 9-pole trapezoidal plug helps to exploit all functions of the test and programming unit.

The programming unit may be connected to the central units from 07 ZE 86 on directly via the 9-pole trapezoidal plug. All functions of the test – and programming unit are available.

If the test and programming unit (e.g. 07 TD 12) is connected via the 25-pole TD plug, the programming of a maximum 4K-program storage is possible only. Furthermore the functions of TEST cannot be carried out.

Hereby the following details have to be regarded:

- the use of the system cable 07 SK 87 R2 (or a self-produced cable according to figure 6 below)
- 0 V - clamp to be connected to the word processor 07 WP 84 and the digital timing unit 07 TZ 82
- earthing of the PROCONTIC b 0 V
- the test and programming unit being connected to a plug socket whose protective conductor is connected with the central earthing point of the PROCONTIC b (switch cabinet plug socket)
- do not use an extension cable for the power network connection of the test and programming unit.
- if previously mentioned measures are impossible, the programmer should be operated by means of a separating transformer.
Connection of 07 ZE 86, 07 ZE 88 or 07 IE 84 with a programming unit by using an interface-converter:

- This configuration consists of one interface-converter type PSM-V24/V11-P and one power supply Type PSM-NT-220AC/15DC/200, manufacturer: Phoenix
- Put switch to DCE

Cable:
- 07 SK 94 R2 (order number: GJR2370500R2)
- 07 SK 87 R2 (order number: GJR5290300R2)

Configuration of user applied cable:

---

Connection of 07 ZE 88 with a programming unit by using an Interface-converter:

Cable:
- 07 SK 94 R2 (order number: GJR2370500R2)
- 07 SK 87 R2 (order number: GJR5290300R2)

88 iso with galvanic insulation from V24

Manufacturer:
Wiesmann & Theis GmbH
Postfach 201505
5600 Wuppertal 2
6.7 Failure messages

In case of the PROCONTIC systems failing to operate a form being added to every PROCONTIC unit may be filled in to describe the failure. This helps to find technical faults during repairs better and gives the feeling of security to the user that the fault will be found.

figure 6: system cable 07 SK 07 R2
# System Description

## ABB Procontic b

Programmable Control System

### Software

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Langtext</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENDSL</td>
<td>Endschalter links</td>
</tr>
<tr>
<td>EMDSR</td>
<td>Endschalter rechts</td>
</tr>
<tr>
<td>GEBO</td>
<td>Geber oben</td>
</tr>
<tr>
<td>GEGB</td>
<td>Geber B</td>
</tr>
<tr>
<td>MERKB</td>
<td>Merker B</td>
</tr>
<tr>
<td>MERKD</td>
<td>Merker D</td>
</tr>
<tr>
<td>ABB</td>
<td>ABB Logo</td>
</tr>
</tbody>
</table>

[Image of ABB software interface]

ABB Schalt- und Steuerungstechnik GmbH
Regulations

Regulations concerning the setting up of installations

Apart from the basic "Regulations for the setting up of power units" VDE* 0100 and for "The rating of creepage paths and air gaps" VDE 0110 the regulations "The equipment of power units with electrical components" VDE 0160 in connection with VDE 0660, part 500, have to be taken into due consideration. Further attention has to be paid to VDE 0113 in case of the control of working and processing machines. If operating elements are to be arranged near shock-hazard parts with protection against electrical shock, VDE 0106, part 100, is relevant.

The user has to ensure that the units as well as the associated components have to be installed according to these regulations. Respectively valid safety regulations, e.g. regulation for the prevention of accidents and the law concerning technical working material, are valid for machines and units connected as well.

ABB Proconetic units have been built according to VDE regulation 0160. The protection against direct touching as demanded by chapter 5.5.1 of this VDE regulation has to be satisfied by the user, e.g. at installing of switch cabinet.

ABB Proconetic units have been designed for operation according to insulation class A of VDE 0110. If considerable pollution is expected during operations, the units have to be installed in housings of the respective kind of protection.

* VDE stands for "Assosiation of German Electrical Engineers".

Note: Please observe the national regulations for the installation of electrical equipments, which are valid in your country.

ABB Schalt- und Steuerungstechnik GmbH
Table of contents

1 Assortment of commands 1-1 6.5 Examples for bit-module programming 6-5
2 Language structure (semantics) 2-1 6.6 Examples for word-module programming 6-5
3 Operators 3-1 6.7 Logging, text output via serial interface 6-6
3.1 ! IF, start of the records 3-1 6.7.1 The direct call 6-6
3.2 & AND, conjunctive 3-1 6.7.2 The indirect call 6-7
3.3 / OR, disjunction 3-2 6.7.3 Output format for logging modules 6-9
3.4 = THEN, allocation 3-2 6.7.4 Example of a text output with the timer 6-10
3.5 S SET 3-2 6.7.5 Example of a text output with the word processor 6-10
3.6 R RESET 3-2 6.8 Time controlled modules 6-10
3.7 N NOT, negation 3-3 6.9 Function modules 6-12
3.8 Comparison operations 3-3 6.10 Table of the function modules 6-13
3.9 Arithmetic operators 3-4

4 Operators 4-1
4.1 E binary input 4-1
4.2 EA analog input 4-1
4.3 A binary output 4-2
4.4 AA analog output 4-3
4.5 Flags 4-3
4.5.1 General information 4-3
4.5.2 M\textsuperscript{\textregistered} global flags 4-4
4.5.3 M local flags 4-5
4.6 Word flags 4-6
4.6.1 General information 4-6
4.6.2 MM word flag 4-7
4.6.3 MA analog flag 4-7
4.7 Step chains 4-7
4.7.1 General information 4-7
4.7.2 S\textsuperscript{\textregistered} global step chains 4-8
4.7.3 S local step chains 4-9
4.8 T timers 4-11
4.9 Z counters 4-13

5 Special commands 5-1 7 Parameter input 7-1
5.1 Subroutine programs 5-1 8 Attributes 8-1
5.2 Program end 5-1 8.1 Possible format changes 8-1
5.3 Branches 5-2 8.2 Transfer and output of data 8-1
5.3.1 Branches onto a given PROCONTIC address 5-2 8.3 Conversion of an analog value to a binary value 8-2
5.3.2 Branches with branch flags 5-2 8.4 Output of word values through binary output or input units 8-3

6 Modules 6-1 8.5 Attribute d 8-3
6.1 BA-module call 6-1 9 Constants 9-1
6.2 BS-module start 6-1 9.1 General information 9-1
6.3 Type of module 6-4 9.2 Indirect constants 9-1
6.3.1 User module type A 6-4 9.3 Direct Constants 9-1
6.3.2 Logging module type P 6-4 9.4 Bit constants 9-2
6.3.3 Function module type F 6-4 8.2
6.4 Principle of the module programming 6-4
Note for the central unit 07 ZE 88:
The notes for the central unit 07 ZE 86 in this software description are available for the central unit 07 ZE 88 too.
ASSORTMENT OF COMMANDS

OPERATORS

1-14
!
&
/
=
+
-
*
<
>
<=
>=
=*

OPERANDS

1-14
E
EV
EA
A
AW
AA
M
M'
R
RW
RA
Mw
MA
S
S'
T
Z
#B0
#B1
KD

INDIRECT ANALOG CONSTANT
TIME CONSTANT
DIRECT WORD CONSTANT
DIRECT DOUBLE WORD CONSTANT
TEXT CONSTANT (up to 12 digits)
MODULE STATUS

ATTRIBUTES

1-14
a
b
d
g
w

ATTRIBUTES (FORMAT CONVERTERS)

SPECIAL COMMANDS

1-14
PE
PE
P
PE
M
M
MC
M
NOPO
NOPI

PROGRAM END
PROGRAM END conditional
PROGRAM END conditional negated
SUBROUTINE END
SUBROUTINE START
SUBROUTINE START negated
SUBROUTINE START with normalizing
SUBROUTINE START with normalizing negated
CLEAR WORD (ALL BITS 1)
FULL WORD (ALL BITS 0)

MODULES AND BRANCHES

1-14
BS
BA
P
PB
BE
SPM
SP
MR

MODULE START unconditional
MODULE START conditional
MODULE CALL unconditional
MODULE CALL conditional
USER module
LOGGING module
FUNCTION module
MODULE DEFINITION user
MODULE DEFINITION logging device
MODULE END unconditional
MODULE END conditional
unconditional BRANCH onto label
conditional BRANCH onto label
unconditional BRANCH onto address
conditional BRANCH onto address
BRANCH LABEL
" = A, F or P
. = numerical
(1 With the comparison operator "=" and the attributes "a", "w", "c", "b", an input operand "E" cannot be entered

Help:
Assign the input operand to a flag, process the comparison operation with this flag

(2 Currently not realized with PROCONTIC b
(3 Currently not realized with PROCONTIC b hardware
(4 With - (negation) not possible

Help:
Assign the negation to the corresponding word before the comparison is done !: WM01,00 = WM01,01
and proceed with the new word flag.
The PROCONTIC programming language is user-oriented. It describes the functional sequence of a control in a form understandable to the user and is based mainly on the rules of Boolean algebra.

A program consists of a sequence of statements which, as a rule, are processed sequentially by the central processor unit. The smallest independent unit in the program is the single statement which, in the simplest case, consists of a condition and an allocation.

! Input 1 = output 1

This program record means that the output is activated if input 1 has a "1" signal or is logic "1". This is written in the PROCONTIC language as follows:

! E 00,01 = A 00,01

Where ! E 00,01 is the condition and = A 00,01 is the allocation.

The following applies analogously:

---

The word ! E 00,01 consists of the operator (!) and the operand E 00,01. In turn, the operand consists of the operand identifier (E) and parameter (00,01).
3.1 ! IF, Start of the record

The operator ! (IF) is always placed at the start of the record. The operator initiates the polling section.

Examples:

a. Bit Operand

\[
! E 00,00 \quad & E 00,01 \quad & E 00,02 \quad = A 00,00
\]

b. Word Operand

\[
! \downarrow MW01,00 \quad = MW01,01 \quad = MW01,02 \quad \downarrow MW01,03
\]

c. Modules

\[
BA 00,0000 \quad Unconditioned \quad module \quad call
\]

In the special commands for the program end and subroutine end, the operator with the operand stands alone in the record.

Examples:

\[
PE \quad PROGRAM \quad END
\]

\[
BE \quad MODULE \quad PROGRAM \quad END
\]

\[
ME \quad SUBROUTINE \quad END
\]

3.2 \& AND, conjunctive

a. Bit Operand

The operator \& (AND) links binary input signals conjunctively.

Example:

\[
! E 03,03 \quad & E 03,02 \quad = A 01,05
\]

A 1-signal is allocated to the output A 01,05 not before both operands in the command also have a 1-signal.

b. Word Operand

(Only with 07 WP 84 or 07 ZE 86)

Two word variables are linked to each other in an AND.

Example:

\[
! \uparrow MW01,00 \quad & MW01,01 \quad = MW01,02
\]

Application examples:

The upper 8 bits of a card flag are to be unscreened.

\[
! MW01,00 \quad #W00255 \quad = MW01,00
\]

Notice:

All word variables and bit variables with attributes may be linked to each other in logic AND.

The logic AND linkage may be combined with the prefix N. Prefix N generates the 1-complementary of the variable contents.

Example:

\[
&N MW01,03
\]

Attention!

The combination \&N MW... is not possible. The combination =N MW..., is allowed, but cannot be entered by the programming unit 07 PC 30 (hex code for =N: DB93).
The sign (−) generates the 2-complementary. By this it is possible to form the 1-complementary.

Example:

\[
-\,\text{MN01,04} - \#000001 = \text{MN01,05}
\]

2-complementary \(-1\) 1-complementary

3.3 / OR, disjunction

a. Bit operand

The operator / (OR) links binary signals disjunctively.

Example:

\[
M'05,05 / E 03,01 = A 01,07
\]

A 1-signal is allocated to the output A 01,07, if one of the both operands in the command has a 1-signal.

Notice:
In the control AND operators are processed before OR operators.

b. Word operand
(with 07 WP 84 or 07 ZE 86 only)

Two word variables are linked to each other in an OR.

Example:

\[
:\text{MN01,05} / \text{MN01,07} = \text{MN01,08}
\]

Notice:
All word variables and bit variables with attributes may be linked to each other in logic OR.

The logic OR linkage may be combined with the prefix \(N\). The prefix \(N\) generates the 1-complementary of the variable content.

Example:

\[
/N \text{MN01,03}
\]

Attention!
The combination \(N\,MN\ldots\) is not possible. The combination \(N\,MN\ldots\) is allowed but cannot be entered by the programming unit 07 PC 30 (hex code for \(N\): D893).

The sign (−) generates the 2-complementary. By this it is possible to form the 1-complementary.

Example:

\[
-\,\text{MN01,04} - \#000001 = \text{MN01,05}
\]

2-complementary \(-1\) 1-complementary

3.4 = THEN, allocation

The operator = (THEN) means the allocation of a record. It allocates the result of the logic linkage by the polling section to the following operand.

Multiple allocations are possible. In this case the result of the logic linkage by the polling section is allocated to the word of the allocating section.

Example:

\[
E 07,14 = A 00,15 = S A 00,14 = R A 00,13 \quad \text{MA03,01} + \text{EA00,00} = \text{MA03,00} = \text{AA00,01}
\]

3.5 S, SET

The optional operator S (set) stands ahead of the operands A (output), M (local flag) and \(M'\) (global flag).

3.6 R, RESET

The optional operator R (reset) leads to the reset of a set operand.

Fundamentally all flags and outputs available in the PROCONCTIC b may be set and reset.
Example for a flip-flop dominantly resetting:

```
E 00,00 = S A 01,07
E 00,01 = R A 01,07
```

After fulfilling the set condition the operand stays set as long as it is not reset by a fulfilled reset condition. If units without battery buffer are used for the flag section, the central unit resets all flags and outputs after power is applied or at the cycle start. In case the outputs and flags are buffered they are not normalized (reset to 0-signal) after power-up or at the cycle start. The former status is maintained.

The command written last is dominant, i.e.:

Reset memory dominantly:

```
E 00,07 = S M'06,05
E 00,08 = R M'06,05
```

Set memory dominantly:

```
E 00,10 = R M'06,06
E 00,09 = S M'06,06
```

Polling the operators in this case must always ask for the dominant operator.

Step-chains and timers cannot be reset.

The set command ahead of the operand T means, that a time value follows to be used for setting the timer:

```
M'06,07 = S T 00,00  #0,123E2
```

### 3.7 \( \mathbf{N} \) NOT, negation

\( \mathbf{N} \) stands by the operator and negates the status of the following operand.

In the allocation section ahead of step chains and in connection with a set respective reset condition the use of \( \mathbf{N} \) is not permissible.

By adding a \( \mathbf{N} \) to the operator an AND (\&) can be converted into an NAND and an OR (\|) into a NOR.

**Examples:**

```
E 00,00 \& E 00,01 = N A 01,07
```

### 3.8 Comparison operations

Two word operands may be compared to each other.

<table>
<thead>
<tr>
<th>Comparison operations</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>=?</td>
<td>Polling for equality</td>
</tr>
<tr>
<td>&lt;</td>
<td>Polling for disequality</td>
</tr>
<tr>
<td>&gt;</td>
<td>Polling for &quot;above&quot;</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Polling for &quot;equal or above&quot;</td>
</tr>
<tr>
<td>&lt;</td>
<td>Polling for &quot;below&quot;</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Polling for &quot;below or equal&quot;</td>
</tr>
</tbody>
</table>

**Sign**

| -                | Minus                           |

Table 3.8.1
a. Comparisons with direct word processing

The result is allocated to any operand:

\[ \text{EA02,00} \geq \text{MA04,06} = \text{M'00,02} \]

At the analog input EA02,00 a value is preset which is checked to be above or equal with a value from a analog flag MA04,06. If the answer is "yes", a "1" is allocated to the global flag M'00,02.

With direct word processing after the comparison each bit command may be used:

The comparison with constants is possible.

\[ \text{MW01,05} \rightarrow \text{NW01023} = \text{S'02,07} \]

Multiple stage comparisons are not possible. These are to be converted in the single stage ones and written each after each.

Notice:
The word with the comparison operator may include minus. It is not possible to write directly =NW,..., since the programming units are not suited for these prefixes. The following procedure must be applied:

\[ \neg \text{MW01,00} = \text{MW01,01} \]

A comparison with the new word flag is done.

b. Comparisons in modules ( BS..A..):

The result is allocated to any operand out of the table 3.8.2 below.

\[ \text{EA02,00} \geq \text{MA04,06} = \text{M'00,02} \]

At the analog input EA02,00 a value is present which is checked to be above or equal with a value from an analog flag MA04,06. If the answer is "yes", a "1" is allocated to the global flag M'00,02.

| Table 3.8.2 |
|---|---|
| with 07 WP 84 R101 permissible: | with 07 WP 84 eff. R202 and 07 ZE 86 permissible: |
| ! EA.... \rightarrow MA...... | ! EA.... \rightarrow MA...... |
| = A 00,00 | = A 00,00 |
| = A 15,15 | = A 15,15 |
| = M 00,00 | = M...... |
| = S...... | = S...... |
| = M 07,15 | = S'...... |
| = S 00,00 | = SPM 003 |
| = BE...... | = BE...... |
| = R...... | = R...... |

3.9 Arithmetic operators

The arithmetic operators + (plus), - (minus), * (multiplication) and : (division) belong to the group of word operators and may be processed in 2 different data formats.

Format 1: binary (16 Bit range) +/- 32767

Format 2: analog range (Micas) +/- 7,999

Arithmetic operators can be used ahead of word operands only.

- Any number of operands may be linked by arithmetic operators.
- Only operands of the same format can be processed (respective operands, which were by attributes).

- The linking side applies the "point before line" rule.

\[
\begin{align*}
&! ( \text{MAO1,00} \quad \text{-MAO1,03} ) \\
&+ ( \text{EA03,04} \quad \text{MA07,05} ) \\
&\quad \text{AA05,03} \quad =-\text{MAO1,07}
\end{align*}
\]

The brackets indicate, how the word processor 07 WP 54 and the central unit 07 ZE 86 process the record; they need not to be entered.

- Direct and indirect constants may be processed in the record.

\[
\begin{align*}
&! \text{MW01,02} \quad \text{#WO0003} \quad = \text{MW01,03}
\end{align*}
\]

If MW01,02 has the value of 24, MW01,03 gets the value \(24 \times 3 = 72\).

In place of the direct constant #WO0003 an indirect constant may be included.

\[
\begin{align*}
&! \text{MW01,02} \quad \text{#WO12} \quad = \text{MW01,03}
\end{align*}
\]

The value of the indirect constant is filed in a fixed section of the user program (see memory configuration).

Notice:

\[
\begin{align*}
\text{W0000} & \quad \text{#W20000} \quad \text{W0002} \\
20000 & \quad \times \quad 2 \\
40000 & \\
\text{W0006} & \quad \text{#W0002} \quad = \text{MW01,00} \\
-2 & \quad = \quad 20000
\end{align*}
\]

Attention:

The limit \(+/-32767\) must not be exceeded. Analog values are limited to \(+/-7,999\).

Example:

To an analog measured value a correction value should be added, which is produced by a BCD input. The result is to be displayed as a BCD value and the analog output voltage shall be limited to \(+/-8\,\text{V}\).

\[
\text{W0000} \quad \text{wEA06,00} \quad + \quad \text{CE 11,08} \quad = \quad \text{MW01,00}
\]
\[
\text{W0009} \quad \text{cA 14,08} \\
\text{W0011} \quad \text{MW01,00} \quad > \quad \text{W} \quad 00800 \quad = \quad \text{SPW 001}
\]
\[
\text{W0017} \quad \text{W} \quad 00800 \quad < \quad \text{MW01,00} \quad = \quad \text{SPW 002}
\]
\[
\text{W0023} \quad \text{SPW 003} \\
\text{W0024} \quad \text{MR 001} \\
\text{W0025} \quad \text{W} \quad 00800 \quad = \quad \text{MW01,00}
\]
\[
\text{W0030} \quad \text{SPW 003} \\
\text{W0031} \quad \text{MR 002} \\
\text{W0032} \quad \text{W} \quad 00800 \quad = \quad \text{MW01,00}
\]
\[
\text{W0037} \quad \text{MR 003} \\
\text{W0038} \quad \text{aMW01,00} \quad = \quad \text{AA08,00}
\]

The analog value is converted into the binary format and is added to a value which was also converted before. The next program step checks, if the value of the flag MW01,00 is above or below the word constant #W00800. If this is the case, the constant is used as output. It must be regarded that the value of the word flag (address 38) was converted into the analog format before the output. The word constant 800 is considered as 8 V in the output.

<table>
<thead>
<tr>
<th>Analog signal</th>
<th>Micas</th>
<th>Binary number</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ 10 V</td>
<td>1,000</td>
<td>1000</td>
</tr>
<tr>
<td>+ 10 V</td>
<td>1,000</td>
<td>1000</td>
</tr>
</tbody>
</table>

8.2
4.1 E Binary input

Example:

E...... Channel number 00 to 15
location number 00 to 63
operand E (binary input)

Possible linkage and allocation forms:

! E...... !N E......
& E...... &N E......
/ E...... /N E......

The operand "E" designates the binary inputs of the system (the inputs > E 15, 15 can only be addressed by means of double word addressing and must therefore not be addressed by the central processor unit 07 ZE 82).

The operand E must not be used in the allocation section of a record (inputs may be polled, but not be set by the program).

Example:

E00,00 & E07,13 / E27,13
= A01,04

device location bus connector 07 BV 84
E 00,00 to E 15,... jumper A-B (bank 0)
E 16,00 to E 31,... jumper A-D (bank 1)
E 32,00 to E 47,... jumper A-C (bank 2)
E 48,00 to E 63,... jumper A-E (bank 3)

Notice:
Binary inputs must be plugged into the locations in the rack as used in the program.

4.2 EA Analog input

Example:

EA...... channel number 00 to 15
location number 00, 02, 04, 06,
08, 10, 12 und 14
operand EA (analog input)

Possible linkage and allocation forms:

! EA...... !- EA......
& EA...... &N EA......
&- EA...... / EA......
/N EA...... /- EA......
+ EA...... - EA......
* EA...... ^= EA......
: EA...... := EA......
< EA...... <= EA......
<= EA...... > EA......
=> EA...... >= EA......
=? EA...... >< EA......

The operand EA designates the analog inputs EA.... including the operator can be addressed by double or triple word addressing only. They may be used by the central processor unit 07 ZE 84 in connection with the word processor 07 WP 84 or by the central processor unit 07 ZE 86.

The operand EA must not be used in the allocation section of a record (inputs may be polled, but not be allocated by the program).

The device address of an analog input unit must always be even (00, 02, 04, 06, 08, 10, 12, 14).

device location bus connector 07 BV 84
EA00,... to EA14,... jumper A-B (bank 0)

(The analog processing is possible in bank 0 only.)

... = channel number 00 to 07 with rack <= 7
... = channel number 08 to 15 with rack > 7
Notice:
The analog inputs must be plugged into the locations in the rack as used in the program (even device location numbers only).

The number format for the analog inputs is according to MICAS (fixed point number): +/- 7,999

whereas it is:

10 V = 1,000 = 100 %

4.3 A binary output

Example:

channel number 00 to 15

A ........

location number 00 to 63

operand A (output)

Possible linkage and allocation forms:

= A ........ = N A ........
= S A ........ = R A ........
! A ........ = N A ........
& A ........ & N A ........
/ A ........ / N A ........

The binary outputs of the system are designated by the operand A. (The outputs > A 15, 150 can only be addressed by double word addressing and must therefore not be addressed by the central processor unit 07 ZE 82.)

All outputs can be set (S) and reset (R), even if they do not exist as devices. Thus they can be used as memories. There are, however, hardware configurations in which not all outputs can be addressed by the central processor unit and used as memories. This applies particularly to the configuration in which only one standard rack without bus driver is used with the central processor unit 07 ZE 82 (A00, 00 ... A15, 07). After fulfilling the set condition the operand stays set as long as it is not reset by a fulfilled erase condition. In cases where units with battery buffering are used for the flag section, the central processor unit maintains all output in the previous condition after power is applied or the cycle was started. Without battery buffer the outputs are put into the 0-status.

Example:

E 00, 00 = A24, 00 = A10, 12

Notice:

Binary outputs must be plugged into the locations in the rack as used in the program.

There are two ways of application in the allocation section:

a. Not buffered

Call: = A ........ , = N A ........

The value of the poll section is allocated to the output identified by the location and channel number.

In case a binary output shall be called by various conditions the format below must not be used:

E 01, 02 = A 03, 01, double allocations
E 01, 03 = A 03, 01, are not allowed

Due to the sequential program processing of the section the last record is processed first. If the input E 01, 02 has a 1-signal, the output A 03, 01 will also take over the 1-signal. If now the input E 01, 03 has a 0-signal, by the processing of the second record a 0-signal will also be allocated to the output A 03, 01. The output A 03, 01 now would
have a 1-signal in between the both allocations.

It must be written instead:

\[ E \ 01,02 \ / \ E \ 01,03 = A \ 03,01 \]

b. Buffered

Call: =S A ....... =R A .......

If set condition is fulfilled the binary output is
set to an "1" and not reset before the erase con-dition is fulfilled. Buffered outputs may be re-peatedly set and reset in the program.

The command written last is dominant, f.i.:

\[ E \ 00,07 =S \ A \ 06,05 \]
\[ E \ 00,08 =R \ A \ 06,05 \]
\[ E \ 00,10 =R \ A \ 06,06 \]
\[ E \ 00,09 =S \ A \ 06,06 \]

The operand A may be used in the poll section and
in the allocation section of a record.

In the poll section a program word with the ope-rand A polls the output channel designated by the
location and channel number for its status (0- or
1-signal).

A binary output must not appear in the allocation
section of a program at the same time as straight
output and as buffered output.

Example:

\[ E \ 01,02 = A \ 02,00 \]
\[ E \ 01,03 =S \ A \ 02,00 \]
\[ E \ 01,04 =R \ A \ 02,00 \]

It must be written instead:

\[ E \ 01,02 = M \ 01,02 \]
\[ E \ 01,03 =S \ M \ 02,00 \]
\[ E \ 01,04 =R \ M \ 02,00 \]
\[ M \ 01,02 = M \ 02,00 = A \ 02,00 \]

4.4 AA Analog output

Example:

\[ AA ....... \]

channel number 00 to 15
location number 00, 02, 04, 06, 08, 10, 12 und 14
operand AA (analog output)

Possible linkage and allocation forms:

\[ = AA ....... = AA ....... \]
\[ = AA ....... = AA ....... = AA ....... \]
\[ = AA ....... = AA ....... = AA ....... \]
\[ = AA ....... = AA ....... = AA ....... \]
\[ = AA ....... = AA ....... = AA ....... \]
\[ = AA ....... = AA ....... = AA ....... \]
\[ = AA ....... = AA ....... = AA ....... \]
\[ = AA ....... = AA ....... = AA ....... \]

The analog outputs of the system are designated by
the operand AA. Including the operator the analog
output AA ....... can be addressed by double word
addressing only. They may be addressed by the cen-tral processor unit 07 ZE 84 in connection with
the word processor 07 WP 84 or by the central pro-cessor unit 07 ZE 86.

The device address of an analog output unit must
always be even (00, 02, 04, 06, 08, 10, 12, 14).

<table>
<thead>
<tr>
<th>device location</th>
<th>bus connector 07 BV 84</th>
</tr>
</thead>
<tbody>
<tr>
<td>AA00... to AA14...</td>
<td>jumper A-B (bank 0)</td>
</tr>
</tbody>
</table>

(Analog processing is possible in bank 0 only)

. = channel number 00 to 07 with rack <= 7,
. = channel number 08 to 15 with rack > 7

Notice:
The analog outputs must be plugged into the loca-tions in the rack as used in the program (even de-vice location numbers only).

The number format for the analog outputs is ac-cord-ing to WICAS (fixed point number): +/- 7,999

whereas it is:

10 V = 1,000 = 100 %

4.5 Flags

4.5.1 General information

In the PROCONIC b the entire memory for the flags
is divided into various sections. For programming
the knowledge about the organisation of the flags
is important.

From the picture below it can be seen that the flag
area is arranged in a local and a global sec-
tion. Global flags are designated with \( M' \) and local flags with \( M \).

Example:

\[
\begin{array}{c}
M'_{\ldots} \\
\text{channel number 00 to 15} \\
\text{location number 00 to 63} \\
\text{operand \( M' \) (global flag)}
\end{array}
\]

Possible linkage and allocation forms:

- \( = M', \ldots, =N M', \ldots \)
- \( =S M', \ldots, =R M', \ldots \)
- \( ! M', \ldots, !N M', \ldots \)
- \( & M', \ldots, &N M', \ldots \)
- \( / M', \ldots, /N M', \ldots \)

There are two ways of application in the allocation section:

a. Non buffered

\[ \text{call: } = M', \ldots, =N M', \ldots \]

The value of the poll section is allocated to the global flag designated by the location and channel number.

In case a global flag shall be called by various conditions, the format below must not be used:

\[ \begin{array}{c}
E 01, 02 = M'03, 01 \\
\text{double allocations}
\end{array} \]

\[ \begin{array}{c}
E 01, 03 = M'03, 01 \\
\text{are not allowed}
\end{array} \]

Due to the sequential program processing of the system the first record is processed first. If the input \( E 01, 02 \) has a 1-signal, the flag \( M'03, 01 \) will also take over the 1-signal. If now the input \( E 01, 03 \) has a 0-signal, by the processing of the second record, a 0-signal will also be allocated to the flag \( M'03, 01 \). The flag \( M'03, 01 \) now would have an 1-signal between the both allocation.

It must be written instead:

\[ \begin{array}{c}
E 01, 02 / E 01, 03 = M'03, 01 \\
\text{not allowed}
\end{array} \]

b. Buffered

\[ \text{call: } =S M', \ldots, =R M', \ldots \]

If set condition is fulfilled the binary flag is set to an "1" and not reset before the erase condition is fulfilled. Buffered outputs may be repeatedly set and reset in the program.

The command written last is dominant, i.e.:

\[ \begin{array}{c}
E 00, 07 = S M'06, 05 \\
\text{reset}
\end{array} \]

\[ \begin{array}{c}
E 00, 08 = R M'06, 05 \\
\text{dominant}
\end{array} \]

\[ \begin{array}{c}
E 00, 10 = R M'06, 06 \\
\text{set}
\end{array} \]

\[ \begin{array}{c}
E 00, 09 = S M'06, 06 \\
\text{dominant}
\end{array} \]

The module number switches the local flag section. The local flag section is reset after ! BE only.

Notice:
In the main memory D7 AS 82 the local flag (M) sections are switched in the user modules type A and in the function modules type F.

With the central processor unit D7 ZE 86 the local flag (M) sections are switched only in the user modules type A. They are not switched in the function modules type F.

4.5.2 \( M' \) Global flags

The global flags may be used in the main program as well as in the modules. The global flags occupy two memory cells.
Polling of the operands in this case always must be done for the dominant operator.

The operand M may be used in the polling section as well as in the allocation of a record.

In the polling section a program word with the operand M causes the poll of the flag channel identified by the location and channel number for its status (0- or 1-signal).

A global flag M must not appear in the allocation section of a program at the same time as straight flag and as buffered flag.

Example:

```
1 E 01,02 = M'02,00  Different
1 E 01,03 = M'02,00  allocation forms
1 E 01,04 = R M'02,00  are not allowed
```

It must be written instead:

```
1 E 01,02 = M03,08
1 E 01,03 = M03,08
1 E 01,04 = M'02,00
```

No M flags can be used with the central processor unit 07 ZE 82. This central processor unit can only address the range from M 00,00 ... M 00,15 to M 07,00 ... M 07,15.

The central processor units 07 ZE 84 and 07 ZE 86 do not distinguish the operands M' and M in the first range. In the main bit program they are considered to be identical physical memory cells. Since, however, the flag M 00,00 may appear twice in the program. With different meanings, in certain circumstances, the central processor unit automatically switches to the proper flag section. To avoid confusions, only M flags should be used in the main program.

4.5.3 M Local flags

The local flags M can only be used in modula.

Example:

```
[M ....]
```

channel number 00 to 15

location number 00 to 07

operand M (local flag)

Possible linkage and allocation forms:

```
= M ...... =N M ......
= M' ...... =R M ......
! M ...... =N M ......
& M ...... &N M ......
/ M ...... /N M .......
```

In the allocation section two application forms are possible:

a. Non buffered

```
Call: = M ...... , =N M .......
```

The value of the poll section is allocated to the local flag identified by the location and channel number.

In case a binary flag shall be called by different conditions, the format below must not be used:

```
= M 01,02 = M 03,01  double allocation
! E 01,03 = M 03,01  not allowed
```

Due to the sequential program processing of the system the first record is processed first. If the input E 01,02 has a 1-signal, the flag M 03,01 will also take over the 1-signal. If now the input E 01,03 has a 0-Signal, by the processing of the second record a 0-signal will also be allocated to the flag M 03,01. The flag M 03,01 would now only have a 1-signal in between these both allocation.

It must be written instead:

```
= M 01,02 = M 03,01
```

PROCONTIC b-state 10.88
b. Buffered

call: =S M ..., =R M ....

If set condition is fulfilled the local flag is set to "1" and not reset before the erase condition is fulfilled. Buffered flags may be repeatedly set and reset in the program.

The command written last is dominant, f.i.:

\[
\begin{align*}
E 00,07 & =S \ M 06,05 & \text{reset} \\
E 00,06 & =R \ M 06,05 & \text{dominant} \\
E 00,10 & =R \ M 06,06 & \text{set} \\
E 00,09 & =S \ M 06,06 & \text{dominant}
\end{align*}
\]

Polling of the operands in this case always must be done for the dominant operator.

The operand M may be used in the polling section as well as in the allocation section of a record.

In the polling section a program word with the operand M causes the poll of the flag channel identified by the location and channel number for its status (0- or 1-signal).

A local flag M must not appear in the allocation section of a program at the same time as straight flag and as buffered flag.

Example:

\[
\begin{align*}
E 01,02 & = M 02,00 & \text{different} \\
E 01,03 & = S \ M 02,00 & \text{allocation forms} \\
E 01,04 & = R \ M 02,00 & \text{are not allowed}
\end{align*}
\]

It must be written instead:

\[
\begin{align*}
E 01,02 & = M 03,08 \\
E 01,03 & = S \ M 03,09 \\
E 01,04 & = R \ M 03,08 \\
M 03,08 & = M 03,09 \\
= M 02,00
\end{align*}
\]

The local flags (M) can be used in modules typ A (with 07 WP 84 resp. 07 E 86) only. A total of 16 areas with 128 flags ist available within the range from (M 00,00 to M 00,15) to (M 07,00 to M 07,15). The range of the local flags ist switched by the module number. A multiple call of the same module with different module numbers does not lead to a flag collision.

Example:

\[
\begin{align*}
# \ BA03,A\ldots & \text{the module call designates the flag range 03} \\
# = ! or = \\
\text{Separate flag areas are reserved for 16 module numbers.}
\end{align*}
\]

The modules BA 16,A\ldots, BA 32,A\ldots and BA 48,A\ldots again access the area 00. This applies analogously to BA 17,A\ldots, which selects the area 01. Several modules can thus affect the same flag area. As a remedy, it is possible to reserve the flags for the various modules:

<table>
<thead>
<tr>
<th>module</th>
<th>flag area</th>
</tr>
</thead>
<tbody>
<tr>
<td># BA 00,A\ldots</td>
<td>M 00,00 to M 02,15</td>
</tr>
<tr>
<td># BA 16,A\ldots</td>
<td>M 03,00 to M 05,15</td>
</tr>
<tr>
<td># BA 32,A\ldots</td>
<td>M 06,00 to M 08,15</td>
</tr>
<tr>
<td># BA 48,A\ldots</td>
<td>M 07,00 to M 07,15</td>
</tr>
</tbody>
</table>

\[
# = ! or = 
\]

Local flag can be processed in modules only. If, however, information from the module is needed in the main program, a transmission from the local flags to global flags is required:

Main program:

\[
\begin{align*}
\text{condition} = \text{WA} & = BA,00,0000 \\
M 03,14 & = A 00,00 \\
M 03,14 & = M 03,14 \\
M 00,13 & = M 03,14 \\
M 02,00 & = M 03,14 \\
\end{align*}
\]

Module:

\[
\begin{align*}
PB,0000 \\
M 00,13 & = M 03,14 \\
\text{BE}
\end{align*}
\]

4.6 Word flag

4.6.1 General information

Word flags serve to store results from computing operations. The value can be filled in these flags either in analog or in binary format.
4.6.2 MW Word flag

Example:

```
MW
```

channel number 00 to 15
location number 01, 03, 05, 07, 09, 11, 13 und 15
operand MW (word flag)

Possible linkage and allocation forms:

- \( = MW_{...} \)
- \( ! MW_{...} \)
- \( & MW_{...} \)
- \( \& MW_{...} \)
- \( / MW_{...} \)
- \( /N MW_{...} \)
- \( + MW_{...} \)
- \( = MW_{...} \)
- \( *= MW_{...} \)

The word flag can be used with odd location numbers only and content binary values.

\[ \text{MW} \]

4.6.3 MA Analog flag

Example:

```
MA
```

channel number 00 to 15
location number 00, 02, 04, 06, 08, 10, 12 und 14
operand MA (word flag)

Possible linkage and allocation forms:

- \( = MA_{...} \)
- \( ! MA_{...} \)
- \( & MA_{...} \)
- \( \& MA_{...} \)
- \( / MA_{...} \)
- \( /N MA_{...} \)
- \( + MA_{...} \)
- \( = MA_{...} \)
- \( *= MA_{...} \)

The analog flags can be used with even location numbers only and content analog values.

\[ \text{MA} \]

4.7 Step chains

4.7.1 General information

In the PROCONTIC b the entire memory for the step chains is divided into various sections. For the programming the knowledge about the organisation of the step chains is important.

From the picture below it can be seen that the step chain area is arranged in a global and a local section. Global step chains are designated with S and local step chains with s.
The module number switches the local step chain range. The local step chain range is reset after ! BE only.

**Notice:**
In the main memory O7 AS 82 the local step change (S) sections are switched in the user modules type A and in the function modules type F.

With the central processor unit O7 ZE 86 the local step change (S) sections are switched only in the user modules type A. They are not switched in the function modules type F!

### 4.7.2 S’ Global step chains

The global step chain S’ can be used both in the main bit program and also in modules.

**Example:**

\[
\begin{align*}
\text{S'.....} & \quad \text{channel number 00 to 15} \\
\text{location number 00 to 63} & \quad \text{operand S' (global step chain)}
\end{align*}
\]

Possible linkage and allocation forms:

\[
\begin{align*}
= \text{S'.....} & \quad \text{! S'.....} \\
\text{!N S'.....} & \quad \text{& S'.....} \\
\text{SN S'.....} & \quad \text{/ S'.....} \\
\text{/N S'.....} &
\end{align*}
\]

A step chain consists of 16 steps. It is an essential feature of the step chain, that only one stage can have a 1-signal. It is an 1 out of 16 characteristic. An allocation activates the programmed step and sets the remaining steps of the chain to 0.

In standby mode (start of program with normalization) the step S’00 has an 1-signal.

The operand S’ can be used both in the polling section and in the allocation section of a record. In the polling section, a program word with the operand S’ causes the step chain identified by the location and channel number to be polled for its status (0 or 1). The step chains operate independently of each other. But they can also be used for parallel or sequential operation.

If more than 16 steps are needed within a program, it is possible to use two or more step chains in series connection. The step chain being left has to be set into normal position (0 step).

**Notice:**
Negated allocation of steps is not permitted.

**Example:**

The central assumed shall accept three internal phases in which it has to provide three certain output signals. Transition from one phase to the other is caused by three messages, which may in any one of the 3 phases. In each phase only one certain message is allowed to be the signal for proceeding. After finishing the cycle the control shall be reset by a fourth signal.

The four messages are: E 00,00, E 00,01, E 00,02 and E 00,03.

The three outputs are: A 02,00, A 02,01 and A 02,02.

\[
\begin{align*}
& \text{! S'00,00 & E 00,00 = S'00,01} \\
& \text{(proceed from stand by mode into the mode S'00,01 with the signal E 00,00)} \\
& \text{! S'00,01 & E 00,01 = S'00,02} \\
& \text{! S'00,02 & E 00,02 = S'00,03} \\
& \text{! S'00,03 & E 00,03 = S'00,00}
\end{align*}
\]

These four equations form the proceeding conditions for the individual phases. By linking the messages with the phases it is attained, that per phase only one certain signal causes the proceeding.

\[
\begin{align*}
& \text{! S'00,01 = A 02,00} \\
& \text{! S'00,02 = A 02,01} \\
& \text{! S'00,03 = A 02,02}
\end{align*}
\]

By these three output conditions it is secured that in a certain phase only the certain corresponding output signal is issued.

In coherence the whole program looks like that:

\[
\begin{align*}
& \text{! S'00,00 & E00,00 = S'00,01} \\
& \text{! S'00,01 & E00,01 = S'00,02} \\
& \text{! S'00,02 & E00,02 = S'00,03} \\
& \text{! S'00,03 & E00,03 = S'00,00} \\
& \text{! S'00,01 = A02,00} \\
& \text{! S'00,02 = A02,01} \\
& \text{! S'00,03 = A02,02}
\end{align*}
\]
a. Parallel operation of step chains S'

The individual 64 chains are fully independent of each other. They can be used next to each other in the program without restrictions. If, however, (i.e. in parallel operation), different chains must be synchronized, one chain is used as leading chain. The sequence of the conditions within the remaining chains is made dependent on the condition of the leading chain (by the corresponding linkages).

b. Branching within a step chain S'

Branching within a step chain is permitted. Following the raising or depressing sequence of the individual conditions is not required.

c. Additions of step chains S'

If for realization of a control task, the 15 conditions of a work cycle are not sufficient, without difficulties one chain can be added to the prior one. It must be regarded, however, that one chain only can run at a time. This may be realized as described below:

\[ S'00,00 \rightarrow S'01,00 \rightarrow E00,00 \]

The chain S'00 can leave its standby mode only when S'01 is in standby mode too.

\[ S'00,15 \rightarrow E13,13 = S'00,00 \]

The second chain is started after the first one has finished. By the first chain going into standby mode and the second chain continuing, a restart of the first chain is prevented.

\[ S'01,10 \rightarrow E07,03 = S'01,00 \]

After finishing the control task the second chain goes into standby mode and the first one can be started again.

The control processor unit 07 ZE 82 cannot process step chains S'. Only in connection with the register 07 RK 80 this control processor unit can address the area from (S 00,00 to S 00,15) to (S 07,00 to S 07,15). The control processor units 07 ZE 84 and 07 ZE 86 do not differentiate between the operands S' and S in its first range. In the main program they are considered as the same physical memory cell. Since under circumstances the step chain S 00,00 appears twice in the program with different tasks, the control processor unit automatically switches into the right step chain range. To prevent confusions only S' step chains range should be used in the main program.

4.7.3 S Local step chain

The local step chains can only be used in the modules.

Example:

\[ S \ldots \]

channel number 00 to 15
location number 00 to 07
operand S (local step chain)

Possible linkage and allocation forms:

\[ ! S \ldots \]
\[ & S \ldots \]
\[ / S \ldots \]
\[ = S \ldots \]

The local step chains (S) can only be used in modules of the type A (with 07 WP 84 resp. 07 ZE 86 only). A total of 16 step chain areas is available, whereas each has 8 local step chains with the address from (S 00,00 to S 00,15) to (S 07,00 to S 07,15). The range is selected by the module number. Multiple calls of the same module with different module numbers does not cause a step chain collision.

Example:

\[ # BA 03,A\ldots \]

the module call selects the step chain area 03
\[ # = ! or = \]

Separate step chain areas are reserved for 16 modules numbers.

Again the modules BA 16,A\ldots, BA 32,A\ldots and BA 48,A\ldots access the area 00. This applies analogously to BA 17,A\ldots, which selects area 01. Several modules can thus influence the same step chain area. As a remedy, step chains can be reserved for the various modules:
Local step chains can only be used in modules. If data out of the module are needed in the main program, a transition of the local step chains to global step chains or global flags has to be written:

Main program:

```
! condition = WA = BA 00,4000
! S'03,14 = A 00,00
! M'02,01 = A 00,01
! ME
```

Module:

```
! PB 4000
```

```
! S 00,13 = S'03,14
! S 00,12 = M'02,01
```

```
! BE
```

A step chain consists of 15 steps. It is an essential feature of the step chain, that only one stage have carry an 1-signal. It is an "1 out of 16" characteristic. An allocation activates the step programmed and sets the remaining steps of the chain to 0.

In standby mode (start of program with normalization) the step S ...00 has an 1-signal.

The operand S can be used both in the polling section and in the allocation section of a record. In the polling section a program word with the operand S causes the step chain identified by the location and channel numbers to be polled for its status (0 or 1).

The step chains work independently of each other. But they may also be operated in parallel or in a sequence.

If more than 16 steps are needed within a program, it is possible to use two or more step chains in series connection. The step chain being left has to be set into normal position (0-step).

**Notice:**
Negated allocation of steps is not permitted.

**Example:**

The control selected shall accept three internal phases, in which it has to provide three certain output signals. Transition from one phase to the other is caused by three messages, which may appear in any one of the phases. In each phase only one certain message is allowed to be the signal for proceeding. After finishing the cycle the control shall be reset by a fourth signal.

The four messages are: E 00,00, E 00,01, E 00,02 and E 00,03.

The three local flags are: M 02,00, M 02,01 and M 02,02.

```
! S 00,00 & E 00,00 = S 00,01
(Proceed from standby mode into the condition
S 00,01 with the signal E 00,00).
! S 00,01 & E 00,01 = S 00,02
! S 00,02 & E 00,02 = S 00,03
! S 00,03 & E 00,03 = S 00,00
```

These four equations form the proceeding conditions for the individual phases. By linking the messages with the phases it is attained, that per phases only one certain signal causes the proceeding.

```
! S 00,01 = M 02,00
! S 00,02 = M 02,01
! S 00,03 = M 02,02
```

By these three output conditions it is secured that in a certain phase only the corresponding output signal is issued.

In coherence the whole program looks like that:

```
! S 00,00 & E00,00 = S 00,01
! S 00,01 & E00,01 = S 00,02
! S 00,02 & E00,02 = S 00,03
! S 00,03 & E00,03 = S 00,00
! S 00,01 = M00,00
! S 00,02 = M00,01
! S 00,03 = M00,02
```

**a. Parallel operation of step chains S**

The individual S chains are fully independent of each other. They may be used next to each other within the program without restrictions.
If, however (f.i. in parallel operation), different chains must be synchronized, one chain is used as leading chain. The sequence of the conditions within the remaining chains is made dependent on the condition of the leading chain (by the corresponding linkages).

b. Branching within a step chain S

Branching within a step chain is permitted. Following the raising or degrading sequence of the individual conditions is not required.

c. Additions of step chains S

If for realization of a control task, the 15 conditions of a work cycle are not sufficient, without difficulties one chain can be added to the prior one. It must be regarded. however, that one chain only can run at a time. This may be realized as described below:

\[ S \ 00,00 \ \& \ S \ 01,00 \ \& \ S \ 00,00 = S \ 00,01 \]

The chain S 00 can leave its standby mode only when S 01 is in standby mode too.

\[ S \ 00,15 \ \& \ E13,13 = S \ 00,00 \]

The second chain is started after the first one has finished. By the first chain going into standby mode and the second chain continuing, a restart of the first chain is prevented.

\[ S \ 01,10 \ \& \ E07,03 = S \ 01,00 \]

After finishing the control task the second chain goes into standby mode and the first one can be started again.

4.8 T Timers

Example:

\[ T \ldots \]

channel number 00 to 15
location number 00 to 03
operand T (timer)

Possible linkage and allocation forms:

- T \ldots
- ! T \ldots
- & T \ldots
- / T \ldots
- =S T \ldots

(set time setpoint)

The time delays are called in the various timer units as follows:

Timer unit 07 T1 80:

<table>
<thead>
<tr>
<th>Time delays available</th>
<th>Rack &gt; 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>T 00,00 to T 00,03</td>
<td>T 00,08 to T 00,11</td>
</tr>
<tr>
<td>T 01,00 to T 01,03</td>
<td>T 01,08 to T 01,11</td>
</tr>
<tr>
<td>T 02,00 to T 02,03</td>
<td>T 02,08 to T 02,11</td>
</tr>
<tr>
<td>T 03,00 to T 03,03</td>
<td>T 03,08 to T 03,11</td>
</tr>
</tbody>
</table>

Timer unit 07 T1 81:

<table>
<thead>
<tr>
<th>Time delays available</th>
<th>Rack &gt; 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>T 00,00 to T 00,01</td>
<td>T 00,08 to T 00,09</td>
</tr>
<tr>
<td>T 01,00 to T 01,01</td>
<td>T 01,08 to T 01,09</td>
</tr>
<tr>
<td>T 02,00 to T 02,01</td>
<td>T 02,08 to T 02,09</td>
</tr>
<tr>
<td>T 03,00 to T 03,01</td>
<td>T 03,08 to T 03,09</td>
</tr>
</tbody>
</table>

Fixed cycle times

| T 00,03 to T 00,07    | T 00,11 to T 00,15 |
| T 01,03 to T 01,07    | T 01,11 to T 01,15 |
| T 02,03 to T 02,07    | T 02,11 to T 02,15 |
| T 03,03 to T 03,07    | T 03,11 to T 03,15 |

T \ldots,03 oder T \ldots,11 = 20 Hz = 50 ms
T \ldots,04 oder T \ldots,12 = 10 Hz = 100 ms
T \ldots,05 oder T \ldots,13 = 5 Hz = 200 ms
T \ldots,06 oder T \ldots,14 = 2,5 Hz = 400 ms
T \ldots,07 oder T \ldots,15 = 1,25 Hz = 800 ms

\ldots corresponds to 00, 01, 02 oder 03.

Timer unit 07 T2 82 and control processor 07 ZE 86:

<table>
<thead>
<tr>
<th>Time delays available</th>
</tr>
</thead>
<tbody>
<tr>
<td>T 00,00 to T 00,15</td>
</tr>
<tr>
<td>T 01,00 to T 01,15</td>
</tr>
<tr>
<td>T 02,00 to T 02,15</td>
</tr>
<tr>
<td>T 03,00 to T 03,11</td>
</tr>
</tbody>
</table>

Fixed cycle times

| T 03,12 to T 03,14    |

T 03,12 = 5 Hz = 200 ms
T 03,13 = 2 Hz = 500 ms
T 03,14 = 0,5 Hz = 2 s
Notice:
The control processor unit 07 ZE 86 allows every

time delay to be programmed. Due to the load ca-
cacity of the control processor unit in one pro-
gram only 48 time delays are permitted.

The number of time delays available depends on the
timer unit used.

The timer units are designed as "0-1" delay ele-
ments. Depending on the unit, the delay time can
be set externally or allocated by the program.

The reset time for a time function is less than
the processing time of a word. The set time ex-
pires, when the time T is started (in the alloca-
tion section) and the activation condition is pre-
sent for longer than the delay time. Polling for
the time T results in the value "0", as long as
the activation conditions for T are not fulfilled
or the time T has elapsed. Polling for the time T
results in the value "1" when the time has elapsed
and the input condition is still fulfilled. If the
activation condition is interrupted during the
duration of the delay time, the timer element is
restarted when the activation condition is fulfil-
led again.

\[
\begin{align*}
&= T \ldots \\
&\text{allocation} \\
&\text{! T \ldots} \\
&\text{polling} \\
&\Delta T < \Delta T \\
&\text{T = set time}
\end{align*}
\]

Example:
\[
\begin{align*}
&! \ E 01,00 = T 03,01 \\
&! A 05,05 & T 03,01 = A 01,02
\end{align*}
\]

Notice:
The value of the polling section is allocated to
the time identified by the location and channel
number.

In case a time value should be polled by various
conditions, the following writing mode must not be
used:
\[
\begin{align*}
&! E 02,02 = T 00,01 \quad \text{double allocation} \\
&! E 02,03 = T 00,01 \quad \text{not permitted}
\end{align*}
\]

Due to the sequential program processing of the
system the first record is processed first. If the
input E 02,02 has an 1-signal, the time T 00,01
will also take over 1-signal. If now the input
E 02,03 has a 0-signal, by the processing of the
second record a 0-Signal will be also be allocated
to the time T 00,01. The time T 00,01 will only be
started between the both allocations. Thus the
time will never elapse.

It must be written instead:
\[
! E 02,02 / E 02,03 = T 00,01
\]

Multiple use of timer units:

Multiple use of timer units is possible if

- the same time value is required several times in
  one program,
- the individual activation conditions are disjunc-
  tively (OR) linked to one allocation,
- each time value polling is conjunctively (AND)
  linked to corresponding activation conditions,
- simultaneous activation by several conditions is
  not possible.

Example:
Start of the delay time:
\[
! E 00,00 / N E 00,05 / S'02,02
= T'01,05
\]

Polling for the delay time:
\[
! E 00,00 & T 01,05 = A 00,02 \\
N E 00,05 & T 01,05 = A'03,05 \\
S'02,02 & T 01,05 = A'02,10
\]

Out of the operands E 00,00, N E 00,05 and S' 02,02
only one may have an "1"-signal at the same time.

By chaining extended values can be obtained.

For the timer 07 TI 80 the delay times have to be
set externally by potentiometer and sliding
switches. With the timer unit 07 TI 81 allocation
of the setpoints to the time delays must be renewed
by the program ahead of each cycle start. It must
be noticed, that the delay times in the 07 TI 81
are not buffered.

Examples for operation with the timer unit
07 T2 82 and the control processor unit 07 86

For the timer unit 07 T2 82 and for the control
processor unit 07 ZE 86 the delay times must be
allocated by the program or entered over the serial
interface.
In order to not overload the processor it is recommended to handle these allocations once only at the start of the program or after initialization.

Notice:

The digital timer 07 TZ 82 actualizes the time delays at the end of each cycle (! PE). Hereby the set time value (setpoint) may be increased by a maximum of one PROCONTIC b cycle.

a. Call of time delay elements

The control processor unit 07 ZE 85 actualizes the time values always, when the time has elapsed. The start condition must be present until the time has elapsed, otherwise the time is not being processed.

b. Setting of time values in the program

Input: # 0, abcEx = <Blank> #0,...E
with x = 1 to 5
with abc = 0 to 9

Time value in seconds: 0, abc = 10^x

c. Call of time values in the program

After the set time has elapsed, the status changes from 0 to 1.

Possibilities to call:

/ T 00.00
! T 00.00
& T 00.00
\ T 00.00

Time value over binary input units

Input: cE 03.00 = <Blank> <Shift> [c] E03.00
(only inputs <14.00 without bank change)

If there are only inputs with bank change used, the conversion has to be done before the setting.

!: cE 16.00 = cM 05.00 only the use of local flags are allowed
! E 18.05 = ST 00.01 c M 05.00

With these applications the inputs

E 03.00 ... E 03.03 (x) Exponent 1 to 5
E 03.04 ... E 03.07 (c) 1/1000 0 to 9 permissible
E 04.00 ... E 04.03 (b) 1/100 0 to 9 inputs
E 04.04 ... E 04.07 (a) 1/10 0 to 9

are read as BCD coded values.

Time value in seconds: 0,abc = 10^x

The c (see chapter attributes) indicates, that E 03.00 and the following 7 input channels as well as E 04.00 and the following 7 input channels are being read as BCD values.

Attention:
The highest time value applicable is 0,999 x 10^5 s (seconds). Time values must not be set permanently but for purposes only, to not overload the processor (increase of cycle time).

4.9 Z Counter

The counters can only be used in connection with the timer unit 07 TZ 82 R201 or with the control processor unit 07 ZE 85.

Example:

Possible linkage and allocation forms:

= Z .......
! Z .......
& Z .......
/ Z .......

= V Z .......
= N Z .......
= S Z .......
= R Z .......

(set counter setpoint)
(reset counter setpoint)

Counter
a. Call of counter

! cond. 1 = Z 00.00 count

Attention:
By the digital timer 07 TZ 82 only the positive flanks at the end of the cycle are counted. The maximum counting frequency is the double of the cycle time. The condition for counting must be at zero for at least one cycle, in order to recognize a positive flank. The counter counts up to the setpoint.

b. Set counter to 0

!N cond. 2 =R M'01.02
! cond. 2 &N M'01.02 dynamic flank
>S M'01.02
>S Z 00.00 set counter to 0

By the control processor unit 07 ZE 86 always the positive flanks at the call of the counter are evaluated and the counter status is hereby actualized. The actual value register of the counter is set to 000 000 and the output Q of the counter is set to zero.

Before each restart the counter should be reset to zero and then counted up to the set value.

c. Call of counter values

After reaching the set counter value the status of the counter switches from 0→1.

Possibilities for calling:

! Z 00.00 !N Z 00.00
& Z 00.00 &N Z 00.00
/ Z 00.00 /N Z 00.00

The condition for calling is:

actual value ≥ setpoint

d. Setting of counter values in the program

! cond. 3 =R M'01.01
! cond. 3 &N M'07.01 dynamic flank
>S M'01.01
>S Z 00.00 #M abcd set counter (by
word).

Input: #M abcd = <Blank> #M......

or

=Z Z 00.00 #D abcd ef set counter (by
double word).

Input: #D abcd ef = <Blank> #D......

The setpoint register of the counter is set to the new value. A comparison is done. The output Q of the counter is modified accordingly.

Set value permissible:

word: #M abcd ef = 0 to 32767
double word: #D abcd ef = 0 to 999 999

Attention:
The highest counter value applicable is 999 999. Counter values must not be set permanently but for purposes only, to not overload the processor (increase of cycle time).

f. Setting of counter values over binary input units

W E01.03 =R M'01.02
W E07.03 &N M'01.02 dynamic flank
>S M'01.01
>S Z 00.00 cE 05.00 set counter

With this application the inputs

E 05.00 ... E 05.03 (d) 1
E 05.04 ... E 05.07 (c) 10
E 06.00 ... E 06.03 (b) 100
E 06.04 ... E 06.07 (a) 1000

permissible input for b, c, d: 0...9
permissible input for a: 0...7

are being read as BCD coded values.

Attention:
The highest valent input E 05.07 (binary value 8000) must not be connected, otherwise faulty functions may occur.

The counter value is: abcd

permissible setpoint: 0 to 7999

The "c" means, that E05.00 and the following 16 input channels are being read as BCD values.
Special commands are self contained in the program and do not designate an operand.

5.1 Subroutine

\[
\begin{align*}
= & \text{ MA} \\
= & \text{ MC} \\
! & \text{ ME}
\end{align*}
\]

Program sections, which are processed depending on conditions only, are called subroutines. In case the condition is not fulfilled, the subroutine is processed without allocation. Word commands cannot be used in the subroutine. Therefore only bit commands may be allocated to subroutines. Further the use of branch commands coming out of MA- resp. MC- and ME-blocks are not allowed.

Notice:
With the central processor unit 07 ZE 86 the commands MA and MC influence all PROCONTIC commands excepted the allocation of word outputs (=AW, =AA). The allocation of word outputs are permitted and may, if used within the subroutine, be allocated to a non-defined value.

At wordprocessing within MA- resp. MC- and ME-blocks has the last word bevor ME to be a bit command or a NOPO.

Call of a subroutine:

\[
\begin{align*}
! & \text{ PE} \quad = \text{ MA} \\
! & \text{ MC} \\
! & \text{ ME}
\end{align*}
\]

subroutine start

\[
\begin{align*}
! & \text{ MA} \\
! & \text{ MC} \\
! & \text{ ME}
\end{align*}
\]

subroutine

subroutine end

In case a = MC is used instead of = MA at the start of a subroutine, all allocated operands (except the step chains) are reset (to zero) in each cycle, if the subroutine is not being opened.

Subroutines also may be allocated inversely thus:

\[
\begin{align*}
! & \text{ cond. 1} \\
! & \text{ cond. 2}
\end{align*}
\]

This means, the subroutines are not processed if the condition is fulfilled.

5.2 End of program

\[
\begin{align*}
! & \text{ PE} \\
= & \text{ PE} \\
= & \text{ N PE}
\end{align*}
\]

With ! PE main programs are terminated. The allocation ! PE causes the address counter to jump onto the first address. This jumping of the address counter (may be used to reduce the cycle time) may also be made dependent on conditions, thus:

\[
\begin{align*}
! & \text{ cond. 3} \\
! & \text{ cond. 4} \\
= & \text{ N PE}
\end{align*}
\]

All programs must include at least one ! PE.

A double ! PE must be used at the end of a program. The double ! PE is automatically entered by the programming units 07 TD 12, 07 PC 30 and 07 PC 31.

If indirect constants are used in the program, only exactly one double ! PE can be included. This double ! PE must be placed to the addresses 3326 and 3327 in the 4 K-programs and to the addresses 7422 and 7423 in the 8 K-programs, as it is entered by the programming units.

<table>
<thead>
<tr>
<th>4 K-program</th>
<th>8 K-program</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000 ! BA00,A000</td>
<td>1000 ! BA00,A000</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>2000 ! PE</td>
<td>2000 ! PE</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>2100 ! PB A000</td>
<td>4100 ! PB A000</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>2200 ! BE</td>
<td>4200 ! BE</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>3326 ! PE</td>
<td>7422 ! PE</td>
</tr>
<tr>
<td>3327 ! PE</td>
<td>7423 ! PE</td>
</tr>
</tbody>
</table>
5.3 Branches

By means of branch commands, linear processing of a program can be interrupted and continued at a different location. Branch commands are executed in absolute form (independent of conditions) or conditionally (depending upon conditions). The possible PROCON'TIC b branch commands are:

! SP .... Absolute branch to the specified PROCON'TIC address
= SP .... Conditional branch to the specified PROCON'TIC address
! SPW .... Absolute branch to a defined label
= SPW .... Conditional branch to a defined label
! MR .... Defined label for the commands ! SPW ... or = SPW ...

5.3.1 Branches to a specified PROCON'TIC address

The range of branches is limited to the address area 0000-3327 in 4 K memories. In 8 K memories, it is possible to branch within the 4 K limits, i.e., within the lower range from 0000-4095 or in the upper range from 4096-7423. It is first necessary to branch to the address 4095 if it is intended to branch from the lower area to the upper area. The address 4096 then corresponds to the address 0000 of the upper area meaning that a branch ! SP 200 would lead to the address 4296.

The address, to which the branch is made in the upper 4 K areas is calculated as follows (see below)

Example:

4095
4096

(corresponds to branch address 0000)

4100 ! SP 200

(4296 - 4096 = 200)

4296 ! M'01,00 = A 01,02 (address, to which the branch was made)

Address management for program shifts is carried out by the user, i.e. in the event of program changes the user must check whether or not branch destinations are still right.

The branch onto a label is preferable over the branch onto an address, since branching onto label is managed by the branch distributor of the programming unit.

5.3.2 Branches with branch labels

256 branch labels (numbers 000-255) can be defined.

Central processor unit 07 ZE 84:

With 4 K memories, the branch labels can be used in the address area from 0000-3327, and in the case of 8 K memories, within the address area from 4096 to 7423. For branches to labels, the branch addresses are administered (even in the event of program shifts) by the programming units.

Central processor unit 07 ZE 86:

The branch labels can be used in the whole address area from 0000 to 7423 with 8 K memories.

Example:

0000

0100 ! E 04,07 = SP 0106 (without branch label)
0102 ! M'02,05 & E 04,08
0105 = A 05,03
0106 ! A 04,08 & M'02,07

0500 ! M'01,07 = SPW 017 (with branch label)
0503 ! A 04,07 & E 04,09
0505 = M'02,03
0507 ! MR 017

! PE

3601 ! SP 507

the branch label is automatically created by the programming unit.
Example for branches onto addresses:

In the PROCONTIC b there are 8 K-program with 3 partial programs, which can optionally be selected via the operands E 01,00, E 01,01 and E 01,02. The cycle time by using branches onto addresses.

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>IN E 01,00  = SP 2500</td>
<td>branch to address 2500</td>
</tr>
<tr>
<td>2500</td>
<td>IN E 01,01  = N'M'63,00</td>
<td>branch to address 3583</td>
</tr>
<tr>
<td>2503</td>
<td>-SP 3583</td>
<td>1. part program 2</td>
</tr>
<tr>
<td>3583</td>
<td>-M63,00 = MA</td>
<td></td>
</tr>
<tr>
<td>4093</td>
<td>-ME</td>
<td>2. part program 2</td>
</tr>
<tr>
<td>4094</td>
<td>IN M63,00 = SP 1000</td>
<td>(4096 + 1000 = 5096)</td>
</tr>
<tr>
<td>5096</td>
<td>IN E 01,02 = SP 1904</td>
<td>(4096 + 1904 = 6000)</td>
</tr>
<tr>
<td>5999</td>
<td>NOPO</td>
<td></td>
</tr>
<tr>
<td>6000</td>
<td>PE</td>
<td></td>
</tr>
</tbody>
</table>

module definitions

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>6320</td>
<td>PB12 AG12</td>
<td></td>
</tr>
<tr>
<td>7422</td>
<td>PE</td>
<td></td>
</tr>
<tr>
<td>7423</td>
<td>PE</td>
<td></td>
</tr>
</tbody>
</table>

4 Indirect constants

8191 pointer to module definition

(* Notice: instead of = SP1904 it could also be written = PE).

When branching to an address, the programmer must check whether or not the address allocations have been altered due to program changes, and thus also the branching distance. This is not necessary, when branching to labels. In the event of program changes, the programming units automatically generate the branch labels by means of the branch table.

Example of a program with branches:

The following example shows a BCD counter, which can be used for 3 different counting tasks. The program is proceeded by an enabling circuit to avoid overlaps.

82

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>W000</td>
<td>E 10,08    = S'00,01</td>
<td></td>
</tr>
<tr>
<td>W003</td>
<td>E 10,09    = S'00,02</td>
<td></td>
</tr>
<tr>
<td>W006</td>
<td>E 10,10    = S'00,03</td>
<td></td>
</tr>
<tr>
<td>W009</td>
<td>E 10,11    = S'00,00</td>
<td></td>
</tr>
<tr>
<td>W012</td>
<td>N S'00,01  = SPM 001</td>
<td></td>
</tr>
<tr>
<td>W015</td>
<td>E 09,08    = M'00,00</td>
<td></td>
</tr>
<tr>
<td>W018</td>
<td>E 09,09    = M'00,01</td>
<td></td>
</tr>
<tr>
<td>W021</td>
<td>BA04,0000</td>
<td></td>
</tr>
<tr>
<td>W022</td>
<td>MR 001</td>
<td></td>
</tr>
<tr>
<td>W024</td>
<td>IN S'00,02 = SPM 002</td>
<td></td>
</tr>
<tr>
<td>W027</td>
<td>E 09,10    = M'00,00</td>
<td></td>
</tr>
<tr>
<td>W030</td>
<td>E 09,11    = M'00,01</td>
<td></td>
</tr>
<tr>
<td>W033</td>
<td>BA01,0000</td>
<td></td>
</tr>
<tr>
<td>W035</td>
<td>MR 002</td>
<td></td>
</tr>
<tr>
<td>W036</td>
<td>IN S'00,03 = SPM 003</td>
<td></td>
</tr>
<tr>
<td>W039</td>
<td>E 09,12    = M'00,00</td>
<td></td>
</tr>
<tr>
<td>W042</td>
<td>E 09,13    = M'00,01</td>
<td></td>
</tr>
<tr>
<td>W045</td>
<td>BA02,0000</td>
<td></td>
</tr>
<tr>
<td>W047</td>
<td>MR 003     = NOPO</td>
<td></td>
</tr>
<tr>
<td>W071</td>
<td>PE</td>
<td></td>
</tr>
<tr>
<td>W075</td>
<td>NOPO</td>
<td></td>
</tr>
<tr>
<td>W076</td>
<td>PB12 AG00</td>
<td></td>
</tr>
<tr>
<td>W077</td>
<td>IN M'00,01 = MC</td>
<td></td>
</tr>
<tr>
<td>W080</td>
<td>IN M'00,00 = = S'00,00</td>
<td></td>
</tr>
<tr>
<td>W083</td>
<td>M'00,00 &amp; M'00,00</td>
<td></td>
</tr>
<tr>
<td>W086</td>
<td>= R M 00,00 = MA</td>
<td></td>
</tr>
<tr>
<td>W088</td>
<td>M 00,01    = N M 00,01</td>
<td></td>
</tr>
<tr>
<td>W090</td>
<td>MA</td>
<td></td>
</tr>
<tr>
<td>W091</td>
<td>M 00,02    / M 00,08</td>
<td></td>
</tr>
<tr>
<td>W093</td>
<td>= N M 00,02 = MA</td>
<td></td>
</tr>
<tr>
<td>W095</td>
<td>M 00,04    / M 00,08</td>
<td></td>
</tr>
<tr>
<td>W097</td>
<td>= N M 00,04 = MA</td>
<td></td>
</tr>
<tr>
<td>W099</td>
<td>M 00,08    = N M 00,08</td>
<td></td>
</tr>
<tr>
<td>W101</td>
<td>ME</td>
<td></td>
</tr>
<tr>
<td>W102</td>
<td>M 00,01    = A 14,08</td>
<td></td>
</tr>
<tr>
<td>W104</td>
<td>M 00,02    = A 14,09</td>
<td></td>
</tr>
<tr>
<td>W106</td>
<td>M 00,04    = A 14,10</td>
<td></td>
</tr>
<tr>
<td>W108</td>
<td>M 00,08    = A 14,11</td>
<td></td>
</tr>
<tr>
<td>W110</td>
<td>BE</td>
<td></td>
</tr>
</tbody>
</table>

In the example, a branch label is selected depending on the setting of the step chain S'00. The following program steps are processed accordingly to the selected branch label. The inputs E 09,08,
E 09,10 and E 09,12 switch the clock frequency. Inputs E 09,09, E 09,11 and E 09,13 are the reset inputs.

The BCD counter is located at addresses W0077 to W0100. The counter reading is switched through to the outputs as a BCD value at the addresses W0102 to W0109.

This example can also be set up with absolute addressing. Then, the command SP 0023 takes the place of SPW 001, the command SP 0034 takes the place of SPW 002 and the command SP 0045 takes the place of SPW 003.
Modules can be defined for recurring program sections. Single problems can be written in modules which then only need to be called in the main program and provided with values.

Another possibility is to make use of ABB ready made modules (function modules) produced by and to call these in the main program and provide them with the appropriate parameters (only in conjunction with the word processor 07 WP 84 or the central processing unit 07 ZE 86).

By programming modules, it is also possible to call them several times in the main program with different values. For this purpose, it is distinguished between local and global flags or step chains. The local ones are "private interim values" within a module, while the global ones can be found in the main program only. The module number BA...A... simultaneously contains the identifier for the selection of the flag area.

Complete memory protection can be attained for the modules by separating the flag areas. Certain values can be accessed to the modules and processed by parametrizing.

The modules can be subdivided into module types BA and BS. BA modules are processed immediately after they are called.

BS-modules are processed parallel to the main program by the word processor. Logically, this means that BS modules generally only run in conjunction with the word processor 07 WP 84 or by the central processing unit 07 ZE 86, although they are also accepted by the timer 07 TZ 82 provided they are logging modules.

Logging and text outputs can only be processed in the background, i.e. with BS types. Since BS 000, (F or P)... (07 TZ 82) and BS 16, (F or P)... (07 WP 84 and 07 ZE 86) are reserved for this task, no other programs should run under this number.

From the both module numbers the types F000 and P000 to P127 may be used.

Modules of the typ BA00,A... to BA15,A... can also be used with the 07 ZE 84 only, without additional processor cards (07 WP 84 or 07 TZ 82).

However, there must then be no word commands included in these modules. The transfer of parameters is not possible.

Modules can be called conditionally and unconditionally. Normally, a module is called conditionally:

Conditional call: ! E 00,00 = BA00,A002

If the module is to be processed in each cycle of the main program, the notation can also be unconditional.

Unconditional call: ! BA00,A002

6.1 Module call BA

As already described, modules can be operated by the central processing unit 07 ZE 84 in pure bit processing or by the word processor 07 WP 84 in word processing. The task allocation depends on the module number. In case of the central processing unit 07 ZE 86 the separation of word and bit modules is withdrawn.

BA00 ... BA15 central processor unit: 07 ZE 84 bit processing, immediate
BA16 ... BA63 processor: 07 WP 84 bit and word processing, immediate; modules type A and type F (with 07 WP 84 R202 only)
BA00 ... BA63 processor: 07 ZE 86 bit and word processing, immediate

The module number is an ordinal number and does not influence the process, only the local flag bank is selected. Parameters (32 max.) can be transferred.

During the direct bit processing the bit central is disabled.

6.2 Module start BS

BS modules are processed in parallel by the word processor 07 WP 84 and by the digital timer 07 TZ 82.
With the central processing unit 07 ZE 86 the module starts are not supported, since background operation is not possible by a single processor system. Only the logging modules BS...,FO00 and BS...,P... can be called by module starts.

BS00,P... 07 TZ 82, logging
BS00,FO00 parallel processing
BS16,P... 07 WP 84, logging
BS16,FO00 parallel processing
BS17 - BS62 word processor 07 WP 84, user module type A and F except text module FO00 (only 07 WP 84 R202), parallel processing
BS00 - BS62 central processing unit 07 ZE 86, logging module type P and text module type FO00
B63 queue check

The status bit B63 (status =1) reports whether or not the queue for modules is full. Therefore no modules should be started with number 63 (BA63 has no influence).

If type A is used in conjunction with a BS module > 15, the word processor searches for all module starts in the user program after initialization of the system and stores the corresponding data in an intermediate memory of the word processor (max. 1 K-words).

The definition must include word operands only, except a bit allocation following a comparison. Since, under certain circumstances, the module starts in the user program are faster than the processing of the individual modules, the word processor has a queue for 16 module starts.

If now a module start occurs in the running program, the data important for this start are entered into the queue and stored:

- module no
- module type
- the parameters following the start (type A and type F, 07 WP 84 R202 only)
- max. 16 word parameters and 16 bit parameters (only 07 WP 84 R202) for each module start

The processing of the modules is then made in the corresponding sequence.

For each module no. (16...62) a status bit can be polled in the program (! B ..., !N B ..., / B ..., /N B ..., & B ... oder & N B ...).

The status has a 1-signal as long as a module is in the queue and has not been completely processed. B63 = 1 indicates, that the queue is full.

The status of word inputs or outputs in the module definition is taken into account at the time of processing.

If the 07 WP 84 arrives at a word input or output during processing of the background program, it must actuate the status of this operand.

Word processor 07 WP 84 R101:

At this point, the word processor does not generate an interrupt for bus access, but waits for the next interrupt. This operand is processed after the status exchange. At the next word input or output, the word processor again waits for an interrupt.

The time between 2 interrupts depends on the program. The operand (e.g. 'MW01.00') should be placed at any particular point in the main program to ensure that at least one word operand is processed per cycle.

Word processor effective 07 WP 84 R202:

The word processor generates the interrupt themselves.

Only the following operands must be included in the module definition for word processing (type A), which are called by background modules (call by = BS...,A...):

<table>
<thead>
<tr>
<th>07 WP 84 R101</th>
<th>07 WP 84 eff. R202</th>
</tr>
</thead>
<tbody>
<tr>
<td>all word operands</td>
<td>all word operands</td>
</tr>
<tr>
<td>in the comparison</td>
<td>in the comparison all</td>
</tr>
<tr>
<td>the following bit operands</td>
<td>bit operands are</td>
</tr>
<tr>
<td>are permitted</td>
<td>permitted</td>
</tr>
<tr>
<td>= A ..., = M ...,</td>
<td>=SPM..., =SPM..., = BE</td>
</tr>
<tr>
<td>and = S ...,</td>
<td>not permitted</td>
</tr>
<tr>
<td>A &gt; 15,15</td>
<td>A &gt; 15,15</td>
</tr>
<tr>
<td>B A ..., =R A ...,</td>
<td>=SA ..., =RA ...,</td>
</tr>
</tbody>
</table>
| =SP..., =BE, | =NA ...
| = M', =N A ... | bit processing except |
| bit processing except | comparison |
| comparison | at least one word call |
| at least one word call | direct access to the |
| in the main program for | bus |
| the word processor | |

8.2
<table>
<thead>
<tr>
<th>Call Kind</th>
<th>type</th>
<th>A00-A63</th>
<th>P000-P127</th>
<th>F...</th>
</tr>
</thead>
<tbody>
<tr>
<td>BA</td>
<td>Processor:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Timing:</td>
<td>07 ZE 84</td>
<td>not possible</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Module definition includes:</td>
<td>Main program is interrupted</td>
<td>07 WP 84</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit program</td>
<td>Special case:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>no parameters</td>
<td>BA00,F127 (head module)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BA00,F001 (serial input)</td>
<td></td>
</tr>
<tr>
<td>BS</td>
<td>Processor:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Timing:</td>
<td>07 TZ 82</td>
<td>07 TZ 82</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Module definition includes:</td>
<td>BS00,P...</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>only 07 ZE 84</td>
<td>only BS00,F000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Main program is interrupted for entering the texts, output over serial interface is in parallel text and the following parameter: E ...... and A ...... &lt;=15,15 M ...... and S ...... &lt;=0,15 attributes b and c</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>No. 16-62</th>
</tr>
</thead>
<tbody>
<tr>
<td>BA</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>BS</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>No. 00-62</th>
</tr>
</thead>
<tbody>
<tr>
<td>BA</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>BS</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
6.3 Module type

A module type (e.g., M002) belongs to each module (e.g., M100).

Example:

! BA 00, A002

There are the types A, F and P

6.3.1 User module type A

User modules, which have to be programmed by the user, are available from A00 to A63. This type identifies a module defined by the user. The module definition is always written following the program, thus after the ! PE. The module type A may include bit and word processing.

! PE
! PB A002

Module program

! BE

6.3.2 Logging module type P

The type P identifies a logging operation described as a module definition.

Modules P000 to P127 are available (refer to the description of logging).

6.3.3 Function module type F

The type F identifies function modules from ABB. These modules are permanently contained in the unit and only need to be called.

The description of the individual function modules follows in the chapters 6 and 6.10.

6.4 Principle of module programming

Address Program

W000

W0500 ! E 02,00 = BA18,A003

W0832 ! M 00,03 & N E 02,07 = BS16,P013

W2320 ! PE

W2800 ! PB A03

! MM01,07 = MV03,04

= CA 00,00

! BE

W3120 ! PB P013

" Text 1"

! BE

! PE

W3220 ! PE

W3840 ! SP 3120 target address P013

= SP 2800 target address A03

W4095
Example of module bit programming

In the present example, a counter is to be used for 3 different purposes.

```
W0000 ! E 09,08 = M'00,00
W0012 ! E 09,09 = M'00,01
W0005 ! BA00,A000
W0007 = M'00,02 = A 13,08
W0010 = E 09,10 = M'00,00
W0013 = E 09,11 = M'00,01
W0016 = BA01,A000
W0018 = M'00,02 = A 13,09
W0021 = E 09,12 = M'00,00
W0024 = E 09,13 = M'00,01
W0027 = BA02,A000
W0029 = M'00,02 = A 13,10
W0034 = NOPO
W0035 = PE
W0036 = PB A000
W0038 = N M'00,00 = S M 00,00
W0067 = M'00,00 & M 00,00 = R M 00,00
W0071 = MA
W0072 = S 00,08 = S 00,00
W0074 = S 00,07 = S 00,08
W0076 = S 00,06 = S 00,07
W0078 = S 00,05 = S 00,06
W0080 = S 00,04 = S 00,05
W0082 = S 00,03 = S 00,04
W0084 = S 00,02 = S 00,03
W0086 = S 00,01 = S 00,02
W0088 = S 00,00 = S 00,01
W0090 = MS
W0091 = S 00,08 = M'00,02
W0094 = M'00,01 = S 00,00 = R M'00,02
W0099 = BE
```

The input E 09,08 switches the counting clock for the counter 1, E 09,10 for counter 2 and E 09,12 for counter 3. The inputs E 09,09, E 09,11 and E 09,13 switch the reset conditions for the counters.

The flag M'00,02 reports, when the set counter reading is reached and switches the corresponding outputs A 13,08, A 13,09 and A 13,10.

The module definition PB A000 includes the counter consisting of a step chain.

Notice:
No parameters (bit parameters) are possible in case of bit modules.

```
In this example a module BA16,A000 is processed periodically every 2 s by a clock signal T 03,10. In the module an analog value is converted into a binary value and issued as BCD value over a binary output unit. Also it is checked by the module, whether the analog value is above 800 (8 V) or below 100 (1 V). As a result of this operation the outputs A 13,08 for below, A 13,10 for above and A 13,09 for the value in between are triggered.

In parallel to the indication by the lights a report text is issued over the interface of the word processor.
```

```
W0000 ! N M'00,00 = S M'00,00
W0044 = S T 03,10  # 0.2000E1
W0007 = N T 03,10 = T 03,10
W0010 = M'00,01
W0012 = N M'00,01 = BA16,A000
W0016 = EA06,00 = EA06,00
W0020 = M'00,02 = A 13,08
W0023 = N M'00,02 & N M'00,03
W0024 = A 13,09
W0028 = M'00,03 = A 13,10
W0032 = N A 13,08 = S M'00,05
W0034 = A 13,08 & M'00,05
W0037 & N B 16
W0038 = R M'00,05 = BS16,F000
W0042 = M'027 = E
W0046 = #" RANGE #" TO
W0056 = #" LOW
W0061 = N A 13,10 = S M'00,08
W0064 = A 13,10 & M'00,06
W0067 & N B 16 = R M'00,06
W0070 = BS16,F000
W0072 = # 027 = E
W0077 = #" RANGE #" TO
W0086 = #" HIGH
W0091 = N A 13,09 = S M'00,07
W0094 = A 13,09 & M'00,07
W0097 = N B 16 = R M'00,07
W0100 = BS16,F000
W0102 = # 027 = E
W0106 = #" VALUE #" IN
W0113 = #" ORDER" = NOPO
W0128 = PE
W0136 = PB A000
W0137 = wEA06,00 = ca 14,08
W0143 = wEA 06,00 > #W 00800
W0149 = M'00,03
W0152 = wEA06,00 < #W 00100
W0159 = = M 00,02
W0161 = BE
```

Since the module is processed every 2 s only, the cycle time of the program is influenced by a small margin only.

---

8.2
6.7 Logging, text output over the serial interface

Text may be issued with updated parameters such as numbers and status. The logging basically is called via the module BS and processed as background program in parallel to the main program.

In the case of modules of the type F000 the next is in the main program. In the case of the type P... the text to be transferred is in the module definition. The format of the output characters corresponds to the character set of the so-called ASCII (American Standard Code of Information Interchange) table. This table contains both visible characters (e.g. typewriter character set) as well as invisible characters (e.g. control characters, line feed, carriage return).

Logging in 07 WP 84 resp. 07 TZ 82:

The word processor and the timer have a limited capacity only. A text must therefore consist of no more than 256 ASCII characters. Before each transfer of a text to the word processor or to the timer, a check must be made as to whether the units are ready for transfer or whether they are still busy with the output of former texts. The set baud rate which determines the number of characters to be transferred per time unit to a peripheral unit, e.g. a printer, is a crucial parameter here as regards the time which the word processor or timer are occupied.

The operand B 00 with status zero signifies that the timer is ready for transfer. This applies analogously to the operand B 16 for the word processor.

The command for log or text output should only be issued once. Therefore, the triggering condition is fulfilled by means of signal flank evaluation:

Example:

Serial output via the timer:

```
E 00,00 =S M'03,00
E 00,00 =R M'03,00 flank evaluation  
M'03,02 &N B 00  
BS00,"... module start  
```

. = number  
* = type F or P.

Serial output via the word processor:

```
E 01,00 &N M'03,01  
S M'03,01 = M'03,03 
E 01,00 =R M'03,01 flank evaluation  
M'03,03 &N B 16 
BS16,"... module start  
```

. = number  
* = type F or P.

Logging via the central processor 07 ZE 86:

The text modules with BS...F000 or BS...P... may be called by all numbers from 00 to 62. The module bit B63 indicates that the queue is full.

When called the the modules are entered into a list (max. 50 modules) only. After the program end the next module is fetched out of the list, all corresponding ASCII characters and values are fetched and stored into an output buffer. This means, that the values are of the cycle at which end the module is truly processed. The value are always issued in 5 digits. The specific output over the serial interface is interrupt controlled, e.g. per cycle a maximum of one text module is issued.

The output buffer can store a maximum of 256 ASCII characters. By the 07 ZE 86 a maximum of 63 texts can be called at the same time, each of them with 256 ASCII characters.

Serial output via the central processor 07 ZE 86:

```
E 02,00 &N M'03,02  
S M'03,02 = M'03,04 
E 01,00 =R M'03,02 flank evaluation  
M'03,04 &N B 61  
BS 61,"... module start 
```

. = number  
* = type F or P.

When calling the text module it is distinguished between a direct and an indirect call.

6.7.1 The direct call

In the case of a direct call the text to be issued is placed after the module start in the program.

The module start for the timer 07 TZ 82 must always be as that:

```
BS00,F000 or = BS00,F000 
```
The module start for the word processor 07 WP 84 must always be as that:

\`
! BS16,F000  or  BS16,F000
\`

With the central processor 07 ZE 86 all numbers are permissible:

\`
! BS38,F000  or  BS38,F000
\`

The text follows in the program:

\`
! cond. (flank)&N B 00  =  BS00,F000

#"TEXT 1

or

#"TEXT E 00,00 (selective also with attributes b or c)

or

#"TEXT MW00,00 (word operand, not with the digital timer 07 TZ 82)
\`

A status 0 or 1 is issued if a binary operand is declared.

\`
#"VALVE M'00,00
\`

Output over the serial interface: VALVE 1

If a word operand is declared, the word value is issued.

\`
#"TEMPERATURE MW00,00
\`

Output over the serial interface: TEMPERATURE +01234

6.7.2 The indirect call

When an indirect call is made, a logging module is allocated to the module:

\`
! cond.  &N B 00  flank
  BS00,P000

! PE

! PB P00
#" TEXT 00
! BE

or

! PB P00
#" TEXT A 00,00 (selective also with attributes b or c)
! BE

or

! PB P00
#" TEXT MW00,01 (word operand, not with the digital timer 07 TZ 82)
! BE
\`

Up to 128 logging modules, which can be called several times, can be defined in the program.

Generally it must be noticed that the statement for the module call must be dynamic (flank evaluation). i.e. the start condition may only be issued once as a command as otherwise the module would be restarted with each cycle.

Attention:

Only direct calls can be made if the central processor 07 ZE 82 R3 is installed in the system together with the digital timer 07 TZ 82.
Permissible parameters in the texts:

<table>
<thead>
<tr>
<th>07 WP 84 R101</th>
<th>07 WP 84 R202</th>
<th>07 ZE 86 R301</th>
<th>07 TZ 82</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bit operands</strong></td>
<td><strong>Bit operands</strong></td>
<td><strong>Bit operands</strong></td>
<td><strong>Bit operands</strong></td>
</tr>
<tr>
<td>E 00,00 to E 15,15</td>
<td>E 00,00 to E 63,15</td>
<td>E 00,00 to E 63,15</td>
<td>E 00,00 to E 15,15</td>
</tr>
<tr>
<td>A 00,00 to A 15,15</td>
<td>A 00,00 to A 63,15</td>
<td>A 00,00 to A 63,15</td>
<td>A 00,00 to A 15,15</td>
</tr>
<tr>
<td>M 00,00 to M 15,15</td>
<td>M 00,00 to M 07,15</td>
<td>M 00,00 to M 07,15</td>
<td>M 00,00 to M 07,15</td>
</tr>
<tr>
<td>S 00,00 to S 07,15</td>
<td>S 00,00 to S 07,15</td>
<td>S 00,00 to S 07,15</td>
<td>S 00,00 to S 07,15</td>
</tr>
<tr>
<td>S'00,00 to S'07,15</td>
<td>S'00,00 to S'07,15</td>
<td>Z 00,00 to Z 00,15</td>
<td>T 00,00 to T 03,15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Word operands</th>
<th>Word operands</th>
<th>Word operands</th>
<th>Word operands</th>
</tr>
</thead>
<tbody>
<tr>
<td>EW01,00 to EW15,15</td>
<td>EW01,00 to EW15,15</td>
<td>EWO1,00 to EW15,15</td>
<td>not possible</td>
</tr>
<tr>
<td>EA00,00 to EA14,15</td>
<td>EA00,00 to EA14,15</td>
<td>EA00,00 to EA14,15</td>
<td>not possible</td>
</tr>
<tr>
<td>AW01,00 to AW15,15</td>
<td>AW00,00 to AW15,15</td>
<td>AW00,00 to AW15,15</td>
<td>not possible</td>
</tr>
<tr>
<td>AA00,00 to AA14,15</td>
<td>AA00,00 to AA14,15</td>
<td>AA00,00 to AA14,15</td>
<td>not possible</td>
</tr>
<tr>
<td>MW01,00 to MW15,15</td>
<td>MW01,00 to MW15,15</td>
<td>MW01,00 to MW15,15</td>
<td>not possible</td>
</tr>
<tr>
<td>MA00,00 to MA14,15</td>
<td>MA00,00 to MA14,15</td>
<td>MA00,00 to MA14,15</td>
<td>not possible</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Constants</th>
<th>Constants</th>
<th>Constants</th>
<th>Constants</th>
</tr>
</thead>
<tbody>
<tr>
<td>not possible</td>
<td>not possible</td>
<td>not possible</td>
<td>not possible</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Attributes</th>
<th>Attributes</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>b and c</td>
<td>b and c</td>
<td>b and c</td>
<td>b and c</td>
</tr>
</tbody>
</table>

All other parameters are not permitted.
6.7.3 Output formats with logging modules

In the cases of the word processor 07 WP 84 R302 and the central processor unit 07ZE 86 R401 the output of values within a text in the logging modules P000 resp. P000 ... P127 can be done in various freely selectable formats.

By using the attributes the content of the low byte of a word flag can be issued as ASCII character without conversion. See also section 8, attributes, concerning this subject.

By the special format identification "#nn" the selection of different output formats is possible. The internally stored values are prepared according to their format identification and issued in the selected display format. There are no differences between analog and word values in issuing formatted values.

The identification "#00" is automatically allocated to the texts with value outputs without format identification.

Output formats:

<table>
<thead>
<tr>
<th>No.</th>
<th>sign</th>
<th>format</th>
<th>internal value</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>±</td>
<td>..........</td>
<td>integer</td>
<td>+00654</td>
</tr>
<tr>
<td>00</td>
<td>±</td>
<td>..........</td>
<td>Analog</td>
<td>+1,000</td>
</tr>
</tbody>
</table>

format with digits to be indicated:

| 01  | .    | +00654  | 4              |
| 02  | .    | +00654  | 54             |
| 03  | .    | +00654  | 654            |
| 04  | .    | +00654  | 0654           |
| 05  | .    | +00654  | 00654          |

format with leading null:

| 06  | .    | +02215  | 0221,5         |
| 07  | .    | +02215  | 022,15         |
| 08  | .    | +02215  | 02,215         |
| 09  | .    | +02215  | 0,2215         |
| 10  | .    | +02215  | .2215          |
| 11  | ±    | +00331  | +00331         |
| 12  | ±    | +00331  | +0033,1        |
| 13  | ±    | +00331  | +003,31        |
| 14  | ±    | +00331  | +0,0331        |
| 15  | ±    | +00331  | +0,0331        |
| 16  | ±    | +00331  | +0,0331        |

leading null replaced by a blank:

| 17  | .    | +00234  | 7654321        |
| 18  | .    | +00234  | 4              |
| 19  | .    | +00234  | 34             |
| 20  | .    | +00234  | 234            |
| 21  | .    | +00234  | 234            |
| 22  | .    | +00347  | 34,7           |
| 23  | .    | +00347  | 3,47           |
| 24  | .    | +00347  | 331            |
| 25  | .    | +00347  | 0,0331         |
| 26  | .    | +00347  | 0,0331         |
| 27  | ±    | +00347  | +347           |
| 28  | ±    | +00347  | +34,7          |

additional formats by suppressing leading null:

| 29  | .    | +00839  | 839            |
| 30  | .    | +00839  | 83,9           |
| 31  | .    | +00839  | 8,39           |
| 32  | .    | +00839  | 839            |
| 33  | ±    | +00839  | +83,9          |
| 34  | ±    | +00839  | +8,39          |
| 35  | ±    | +00839  | +839           |
| 36  | ±    | +00839  | +8,39          |

Example:
The actual filling status is to be displayed on the screen in 4 digits and without digits following the comma, together with a text.
- The filling capacity is stored in the word flag MN01,00.
- As format #04 is selected.
- Content MN01,00 = 02351

Program:

<table>
<thead>
<tr>
<th>! PB P000</th>
<th>! THE</th>
<th>! ACTUAL</th>
<th>! FILL.STATUS</th>
<th>! IS</th>
<th>! #04</th>
<th>MN01,00</th>
<th>#</th>
<th>#letters</th>
<th>! BE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>44 45 52 20</td>
<td>41 48 54 55 45 4C 4C 45 20</td>
<td>46 55 45 4C 4C 53 54 41 4E 44 20</td>
<td>42 45 54 52 41 45 47 54 20</td>
<td>33 32 35 31</td>
<td>20</td>
<td>4C 49 54 45 52 20</td>
<td>00</td>
<td></td>
</tr>
</tbody>
</table>

= blanc
Screen output after the module is called:

THE ACTUAL FILLING STATUS IS 3251 LITERS

6.7.4 Example of a text output with the timer

W0000  !N 09,08 =S M 00,00
W0002  !E 09,08 & M 00,00 =R M 00,00
     = BS00,P000
W0007  "#TEXT  "#IS  "#ISSUED
     "#DIRECTLY
W0029  "#AFTER  "#THE  "#FULFILLED
     "#CONDITION
W0061  ":#THIS  ":#IS
     ":#ALSO
W0077  ":#POSSIBLE  ":#WITH  ":#THE
     ":#07
W0090  ":#ZEB2  ":#R3  ":#".
     ":#"
W0110  NOPO NOPO NOPO

A flank evaluation, which starts the module BS00, P000, is located in the addresses W0000 to W0004. The text to be issued upon the module call is located in the address W0007 to W0109.

6.7.5 Example of a text output with the word processor 07 WP 84 or with the central processor 07 ZE 86

In the following part of the program, 3 different print tasks are started by the flank evaluation depending on the inputs E09,09, E09,10 and E09,11.

W0120  !N 09,09 =S M 00,01
W0122  !E 09,09 & M 00,01 =R M 00,01
     = BS16,P000
W0127  NOPO NOPO NOPO
W0130  !N 09,10 =S M 00,02
W0132  !E 09,10 & M 00,02 =R M 00,02
     = BS16,P001
W0137  NOPO NOPO NOPO
     NOPO
W0141  !N 09,11 =S M 00,03
W0143  !E 09,11 & M 00,03 =R M 00,03
     = BS16,P002
W0148  NOPO NOPO NOPO
     NOPO
W0152  NOPO NOPO
W0154  !PE NOPO
W0173  !PB P000  "#TEXT  "#OUTPUT
     "#WITH
W0190  "#LOGGING  "#MODULE  "#"

The module BS16,P001 is called at address W0135. This module leads to a text output via the word processor.
The logging module P000 is processed at address W0173.
A pure text is issued.

At address W0244 a text is issued, which describes the status of an operand.

At the address W0300 a text is issued, which is linked to a value, which is applied as a BCD value to an input unit.

The number 27 is processed in the text. The 27 followed by e corresponds in the ASCII table to the control character ESC E, which leads to deletion of the monitor in the programming unit 07 TD 12.
Other control commands can be found in the description of the connecting terminal.

6.8 Time controlled modules

For certain function modules, e.g. regulator modules, it is required, that the function module is called within certain time intervals (scanning time). This scanning time is generated by the internal timer of the word processor.

The processing of the time controlled function modules is done by a head module. The head module includes the interrupt time for the time controlled function modules as well as the label for the module record.

The head module is located after the ! PE in the program. Hereby no interrupt is generated in the running control.
The head module and the module record are entered when normalizing. The head module must be ahead of the module records.

Example:

Main program

; PE

Head module

! BA00,F127
! BA00,A020
#0,abcE1
NOPO
! BE

(*) = In the 07 WP B4 R202 in 10 ms steps and in the case of 07 ZE 86 in 20 ms steps

Module record: (included in the range of module definitions)

P8 A020
! B418,F022
EA00,00
MA00,00
! B419,F023
MA01,00
MA02,00
! B420,F040
EA00,01
MA00,00
#W 00203
#W 00001
! BE

The number F127 is allocated as head module.
A20, table of the module record (order number)
Clock time, constant value = time value in s (*).

For the word processor 07 WP B4 the number of modules (n) is calculated to:

\[ n \leq \frac{\text{shortest scanning time (in ms)}}{10} \]

Shortest scanning time required: 40 ms

\[ n = \frac{40}{10} = 4 \]

module records are possible.

There are 8 module records needed.

\[ \text{shortest scanning time} = n \times 10 = 8 \times 10 = 80 \text{ ms} \]

For the central processor 07 ZE 86 the number of modules (n) is calculated to:

\[ n \leq \frac{\text{shortest scanning time (in ms)}}{20} \]

Shortest scanning time required: 40 ms

\[ n = \frac{40}{20} = 2 \]

module records are possible.

There are 8 module records needed.

\[ \text{shortest scanning time} = n \times 20 = 8 \times 20 = 160 \text{ ms} \]

The module records must be arranged in a raising sequence according to their scanning times. Only module records of the type A.. are permitted. The run time per module record cannot exceed 10 ms. This time is monitored in RUN-mode; if faulty an error message is issued and the control is brought to a STOP; similar behavior as in the case of cycle monitoring.

A maximum of 16 module records is possible. The module records are filed together with the background modules. The program capacity of all modules of the type A.. must not exceed 1 K words. This limit is not monitored by the programming units. The word processor monitors this limit, in a given case an error message is issued while normalizing. Out of a time controlled module only modules of the type BA..,F... may be called.
Priority of processing

Operation of the serial interface (interrupt)
Time controlled modules
Direct word processing
BS...F... background A...
Bit processing

1. Processing of bit program and background modules in parallel
2. Interrupt of the bit program for arithmetic processing by the word processor
3. Interrupt of the word processing by time controlled modules
4. Processing of bit program and background modules in parallel (not with 07 ZE 86).

6.9 Function modules

Function modules allow the use of more complex functions, which may be executed by the machine on a simple call. These modules are "hard wired" functions, stored in the operating systems of the word processor 07 WP 84 or of the central processor unit 07 ZE 86. All the user has to do is to provide the corresponding parameters. The specific module program of the function modules is included in the EPROM's of the word processor 07 WP 84 or of the central processor unit 07 ZE 86.

The function modules can be called conditioned or unconditioned by BA (direct processing) and BS (parallel processing, with the word processor 07 WP 84 only).

Example:

Parameter list

Parameter list

NOPO

NOPO

Parameter list

Parameter list

NOPO

NOPO

If a function module is needed several times in a program, it has to be combined with different module numbers, in case there are values of the past (is true f.i. with regulators).
6.10 List of the function modules

There are the following function modules included in the word processor 07 WP 84 and in the central processor unit 07 ZE 86:

<table>
<thead>
<tr>
<th>Module number</th>
<th>Module Meaning name</th>
<th>Processor</th>
<th>Cells (module number and module type)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>07 WP 84</td>
<td>07 TZ 82</td>
</tr>
<tr>
<td>F000</td>
<td>TEX Logging and text module</td>
<td>eff.R101</td>
<td>eff.R101</td>
</tr>
<tr>
<td>F001</td>
<td>SIP Serial input</td>
<td>eff.R202</td>
<td>–</td>
</tr>
<tr>
<td>F002</td>
<td>SIN Interface initialization</td>
<td>eff.R202</td>
<td>–</td>
</tr>
<tr>
<td>F003 (1)</td>
<td>EMAS Receipt of ASCII characters</td>
<td>eff.R202</td>
<td>–</td>
</tr>
<tr>
<td>F020</td>
<td>IDS Indirect write</td>
<td>eff.R202</td>
<td>–</td>
</tr>
<tr>
<td>F021</td>
<td>IDL Indirect read</td>
<td>eff.R202</td>
<td>–</td>
</tr>
<tr>
<td>F022 (*)</td>
<td>BETR Absolute value former</td>
<td>eff.R202</td>
<td>–</td>
</tr>
<tr>
<td>F023 (*)</td>
<td>BEG Limiter</td>
<td>eff.R202</td>
<td>–</td>
</tr>
<tr>
<td>F024 (*)</td>
<td>KPL Complementary former</td>
<td>eff.R202</td>
<td>–</td>
</tr>
<tr>
<td>F040(*) (1)</td>
<td>PIR PI-regulator</td>
<td>eff.R202</td>
<td>–</td>
</tr>
<tr>
<td>F060</td>
<td>ZYK Cycle monitoring</td>
<td>eff.R202</td>
<td>eff.R201</td>
</tr>
<tr>
<td>F127</td>
<td>KOPF Head module</td>
<td>eff.R202</td>
<td>–</td>
</tr>
</tbody>
</table>

(*) = may also be used as time controlled module.
(1 = Note:
Are modules with values of past used (EMAS and PIR), they need different module numbers.
(2 = := or := BS00 to BS63
(3 = := or := BA16 to BA63
(4 = := or := BA16 to BA63 and/or := BS16 to BS63
(5 = := or := BA00 to BA63
Text module

Module type

Run time:
- call and end: 250 µs
- per each ASCII character: 35 µs
- parameter: 100 µs
- attributes c/b together: 200 µs
Value of the past: 0 words

F000

\[ \begin{array}{c}
\text{parameter} \\
\text{ASCII-character} \\
\end{array} \]

<table>
<thead>
<tr>
<th>Element</th>
<th>Kind of data</th>
<th>Meaning of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BS16, F000</td>
<td>module number and module type</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>text, ASCII character</td>
</tr>
<tr>
<td></td>
<td></td>
<td>resp.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>variables</td>
</tr>
</tbody>
</table>

The following operands are for the planning of the module:

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
<th>Operands permitted with</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>module number and module type</td>
<td>07 TI 82 = BS, 00</td>
</tr>
<tr>
<td>2</td>
<td>text, ASCII character</td>
<td>07 WP 84 = BS, 16</td>
</tr>
<tr>
<td></td>
<td>resp.</td>
<td>07 ZE 86 = BS, BS, 00–63</td>
</tr>
<tr>
<td></td>
<td>variables</td>
<td>all bit variables</td>
</tr>
<tr>
<td></td>
<td></td>
<td>all word variables</td>
</tr>
<tr>
<td></td>
<td></td>
<td>attributes b, c</td>
</tr>
</tbody>
</table>

8.2

6-14
Text module
Module type

Run time:
- call and end: 250 µs
- per each ASCII character: 35 µs
- Parameter: 100 µs
- Attributes c/b together: 200 µs
Value of the past: 0 words

The call of a time controlled module is not permitted.

Notice:
A text module may contain up to 256 ASCII characters. One bit parameter counts as 1 ASCII character, one word parameter as 7 ASCII characters. The module bit B.. indicates, whether or not the text buffer is empty (B.. = 0).

See also sections 6.7 and 8.

Example:

: Cond.1 = BSO0, F000
"TEXT "IS "ISSUED "DIRECTLY
"AFTER "THE "CONDITION "IS
"FULFILLED ". "THIS "IS
"ALSO "POSSIBLE "WITH "THE
"07 "ZEB2 "R3 "

NOPO NOPO NOPO
Serial input

Module type

Run time:
- entering and issuing of the parameters: 200 µs
- Receipt of the value: 700 µs
- Receipt of the analog value: 550 µs
Value of the past: 0 words

F001

acknowledgment
target variable
invalid
ready

<table>
<thead>
<tr>
<th>Element</th>
<th>Kind of data</th>
<th>Meaning of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td># BA,F001</td>
<td>module number and module type</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>input E2 (bit operand): acknowledgment</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>output E3 (word operand): target variable</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>output A4 (bit operand): invalid message</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>output A5 (bit operand): ready message</td>
</tr>
</tbody>
</table>

The following modules are for the planning of the modules:

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
<th>Operands allowed with</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>O7 WP 64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>! BA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>00</td>
</tr>
</tbody>
</table>
| 1       | module number and module type                    | each bit variable allowed
| 2       | 0 = module enabled for receiving data
|         | 1 = module disabled for receiving data           |                        |
| 3       | data received are filed onto this variable       |                        |
| 4       | 0 = data received are valid                      |                        |
|         | 1 = data received are invalid                    |                        |
| 5       | 0 = data not yet received and filed              |                        |
|         | 1 = data received and filed                      |                        |

# = ! or =
* = letter
. = number

8.2
Serial input

Module type

Run time:
- entering and issuing of
  the parameters: 200 µs
- Receipt of the value: 700 µs
- Receipt of the analog value: 550 µs
Value of the past: 0 words

The function module SIP for the serial interface receives per interrupt values from the interface and transfers them to a word flag.

The module initially accepts all ASCII characters and reflects a corresponding echo. Only the last 6 characters ahead of the termination character (〈CRL〉) are evaluated. These 6 characters must be within the number range of ±32766 resp ±7,999. Each input is signalized by the 'READY' message. In case other characters were transferred, the message 'INVALID' is generated and the last ASCII character is filled onto the target variable in ASCII. With proper input the numerical value is filled onto the variable indicated.

Input errors (typing error before the termination character) may be corrected by repeating the input. The last 6 characters of the input are valid.

The data generated and transferred by the word processor are transferred according to the selection in the module SIN.

Depending on its status the module transmits XON resp. XOFF.

The following statements are for the word processor 07 WP 84:

The special SIP module is placed after ! PE. The SIP module needs the module cycle monitoring (FO60) for triggering the acknowledgment. The cycle monitoring must not be switched on.

Example:

<table>
<thead>
<tr>
<th>07 WP 84</th>
<th>07 ZE 86</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module</td>
<td>Content</td>
</tr>
<tr>
<td>BA00,F001</td>
<td>M'01,00</td>
</tr>
<tr>
<td>M'01,00</td>
<td>0</td>
</tr>
<tr>
<td>MW01,01</td>
<td>000251</td>
</tr>
<tr>
<td>M'00,00</td>
<td>0</td>
</tr>
<tr>
<td>A 01,00</td>
<td>1</td>
</tr>
</tbody>
</table>

The module is always active. It may be disabled via the acknowledgment bit by the user program. The acknowledgment bit is always polled by the SIP module, when the module cycle monitoring (FO60) is called. The acknowledgment bit should be present for one cycle only (until the control passed the module cycle monitoring in the program).

The special processing of the SIP module, after receiving the data over the serial interface, is handled on the same priority level as the background modules. I.e. after each module processing an order of the SIP module is processed.
### Interface initialization

**Serial input**
- **Module type**

Run time: 1200 µs
Value of the past: 0 words

---

**F002**

2. Interface identification
3. Baud rate
4. Stop bit
5. Length of the character
6. Parity Enable
7. Even Parity
8. Echo
9. Send Break Character
10. Termination character

---

<table>
<thead>
<tr>
<th>Element</th>
<th>Kind of data</th>
<th>Meaning of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td># BA...F002</td>
<td>Module number and module type</td>
</tr>
<tr>
<td>2</td>
<td><strong>...</strong></td>
<td>Input E 2 (word operand): Interface identification</td>
</tr>
<tr>
<td>3</td>
<td><strong>...</strong></td>
<td>Input E 3 (word operand): Baud rate</td>
</tr>
<tr>
<td>4</td>
<td><strong>...</strong></td>
<td>Input E 4 (word operand): Stop bit (1 or 2)</td>
</tr>
<tr>
<td>5</td>
<td><strong>...</strong></td>
<td>Input E 5 (word operand): Length of the character (7 or 8)</td>
</tr>
<tr>
<td>6</td>
<td><strong>...</strong></td>
<td>Input E 6 (bit operand): Parity bit</td>
</tr>
<tr>
<td>7</td>
<td><strong>...</strong></td>
<td>Input E 7 (bit operand): Even/odd</td>
</tr>
<tr>
<td>8</td>
<td><strong>...</strong></td>
<td>Input E 8 (bit operand): Echo</td>
</tr>
<tr>
<td>9</td>
<td><strong>...</strong></td>
<td>Input E 9 (bit operand): Send break</td>
</tr>
<tr>
<td>10 *)</td>
<td><strong>...</strong></td>
<td>Input E10 (word operand): Termination character</td>
</tr>
</tbody>
</table>
Serial input
Module type

Run time: 1200 µs
Value of the past: 0 words

The following operands are for the planning of the module:

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
<th>Operands allowed with</th>
<th>07 WP 84</th>
<th>07 ZE 86</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>! BA&lt;sub&gt;i&lt;/sub&gt; = BA</td>
<td>16-63</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>! BA&lt;sub&gt;i&lt;/sub&gt; = BA</td>
<td>00-63</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>module number and module type</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>interface identification (for 07 WP 84 R202 always 1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>300, 600, 1200, 02400, 4800 or 9600 baud</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1 = 1 stop bit, 2 = 2 stop bits</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>7 = 7 data bits, 8 = 8 data bits</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1 with parity bit 1, 0 without parity bit 1 (1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1 = even, 0 = odd</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0 = with echo, 1 = without echo</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0 = standard mode, 1 = TX0 goes to 0 until the SIN is recalled</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>(2) termination character &lt;CR&gt; = 13, &lt;ETX&gt; = 3 and &lt;SP&gt; = 30</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(1) Such identified parameters are not evaluated by the word processor 07 WP 84 R202. The word processor 07 WP 84 R202 always sends even parity.

The initialization and the configuration of the serial interface can be done by the user by means of the module SIN.

The module initializes the interface, i.e. the receiver and the transmitter buffers are reset and an XON is send.

The interface configuration is buffered. After applying power without the buffer battery the following standard setting is valid:

300 baud
1 stop bit
7 characters + even parity (no parity in the cases of 07 WP 84 R302 and 07 ZE 85 R401)
characters received are echoed

The SIN module is located in the main program and is called like a normal function module.

Example: for the standard setting

<table>
<thead>
<tr>
<th>Module</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>! BA20,F002</td>
<td></td>
</tr>
<tr>
<td>#W00001</td>
<td>00001</td>
</tr>
<tr>
<td>#W00002</td>
<td>00300</td>
</tr>
<tr>
<td>#W00004</td>
<td>00300</td>
</tr>
<tr>
<td>#W00007</td>
<td>00007</td>
</tr>
<tr>
<td>#B1</td>
<td>1</td>
</tr>
<tr>
<td>#B2</td>
<td>1</td>
</tr>
<tr>
<td>#B0</td>
<td>0</td>
</tr>
<tr>
<td>#B0</td>
<td>0</td>
</tr>
<tr>
<td>#W00013</td>
<td>00013 (CR)</td>
</tr>
<tr>
<td>ND0</td>
<td></td>
</tr>
</tbody>
</table>

A steadily fulfilled = BA condition leads to a disliked permanent initialization of the interface.
Receipt of ASCII-Zeichen

Module type

Run time:
Basic run time 1200 μs
Optional run time see text
Value of the past: 3+n words (n = number of the useful informations)

EMAS F003

<table>
<thead>
<tr>
<th>Element</th>
<th>Kind of data</th>
<th>Meaning of the elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td># BA...F002</td>
<td>module number and module type</td>
</tr>
<tr>
<td>2</td>
<td>&quot;        &quot;</td>
<td>input E 2 (bit operand): acknowledgment</td>
</tr>
<tr>
<td>3</td>
<td>&quot;        &quot;</td>
<td>input E 3 (word operand): interface identification</td>
</tr>
<tr>
<td>4</td>
<td>&quot;        &quot;</td>
<td>input E 4 (word operand): number of useful informations</td>
</tr>
<tr>
<td>5</td>
<td>&quot;        &quot;</td>
<td>output A 5 (bit operand): invalid</td>
</tr>
<tr>
<td>6</td>
<td>&quot;        &quot;</td>
<td>output A 6 (bit operand): ready</td>
</tr>
<tr>
<td>7</td>
<td>&quot;        &quot;</td>
<td>output A 7 (word operand): telegram number</td>
</tr>
<tr>
<td>8</td>
<td>&quot;        &quot;</td>
<td>output A 8 (word operand): 1. useful information</td>
</tr>
<tr>
<td>n</td>
<td>&quot;        &quot;</td>
<td>output A n (word operand): n. useful information</td>
</tr>
<tr>
<td>01</td>
<td>&quot;#01.&quot;</td>
<td>input E01 (text constant): 1. comparison text</td>
</tr>
<tr>
<td>nn</td>
<td>&quot;#nn.&quot;</td>
<td>input Enn (text constant): n. comparison text</td>
</tr>
</tbody>
</table>
Receipt of ASCII characters

Module type

Run time:
Basic run time 1200 µs
Optional run time see text
Value of the past: 3*n words (n = number of the useful informations)

The following operands are for the planning of the module:

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>module number and module type</td>
</tr>
<tr>
<td>2</td>
<td>module released for receiving, 0 = enabled, 1 = disabled setting of the serial interface, standard value = 01.</td>
</tr>
<tr>
<td>3</td>
<td>number of output parameter for the useful data to be filed n = max. number of the W-blocks + the W-characters</td>
</tr>
<tr>
<td>4</td>
<td>invalid, indicates whether the telegram received matches a filed comparison format. 1 = invalid, 0 = valid</td>
</tr>
<tr>
<td>5</td>
<td>ready, 1 = ready, 0 = not ready</td>
</tr>
<tr>
<td>6</td>
<td>identification of the comparison telegrams (&lt; 99)</td>
</tr>
<tr>
<td>7</td>
<td>useful data, to be filed onto flags (n&lt;128)</td>
</tr>
<tr>
<td>8...8+n</td>
<td>texts, to be compared with the data received (nn &lt; 128)</td>
</tr>
</tbody>
</table>

# = ! or =
* = letter
. = number

If there is a conformity between the telegram received and the comparison format, the numerical values resp. the ASCII characters and the telegram number are filed onto the corresponding word flags one after the other.

When issuing the "ready" signal (ready = 1) the output "invalid" = 0 indicates a format conformity. The message "invalid" = 1 is issued, if the telegram received does not conform with any of the comparison formats filed. The "ready" message is generally set at the end of the evaluation and stays active until the output "acknowledgment" = 1 occurs. The output "acknowledgment" = 1 causes the output "ready" and "invalid" to be reset as well as a release for the receipt of further characters. The outputs "ready" and "invalid" are filed as values of the past.

Comparison texts have the structure 
#"nn "text (termination character) and are filed directly following the function module EMAS. The two-digit number following the start-text character indicates the telegram number. The character # is for the number identification * is reserved for the text characters identification.
Receipt of ASCII-Zeichen
Module type
Run time:
Basic run time: 1200 µs
Optional run time: see text
Value of the past: 3+n words (n = number of the useful informations)

Each string must be terminated by a termination character, to identify the end.
The first character of a string must not be a $. The function modules EMAS and SIN cannot be called simultaneously.

<table>
<thead>
<tr>
<th>Module</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>! BA20, F003</td>
<td></td>
</tr>
<tr>
<td>W'01,00</td>
<td>0 or 1</td>
</tr>
<tr>
<td>W'00001</td>
<td>1</td>
</tr>
<tr>
<td>W'0003</td>
<td>3</td>
</tr>
<tr>
<td>A 01,01</td>
<td>0 or 1</td>
</tr>
<tr>
<td>A 01,02</td>
<td>0 or 1</td>
</tr>
<tr>
<td>W'03,00</td>
<td>0 to 99 numbers or ASCII characters</td>
</tr>
<tr>
<td>W'01,00</td>
<td>numbers or ASCII characters</td>
</tr>
<tr>
<td>W'03,00</td>
<td>numbers or ASCII characters</td>
</tr>
<tr>
<td>W'03,02</td>
<td>numbers or ASCII characters</td>
</tr>
<tr>
<td>01Text 1###,**Z&lt;CR&gt;</td>
<td></td>
</tr>
<tr>
<td>02Text 2###,**T&lt;CR&gt;</td>
<td></td>
</tr>
<tr>
<td>03Text 3</td>
<td></td>
</tr>
<tr>
<td>NOPO</td>
<td></td>
</tr>
</tbody>
</table>

Structure of the comparison text:

nn <text> ###<text> **<text> <CR> (general structure)
Q1 ABCDEFG ### HIJKLM ** NOPQRSTUVWXYZ<CR> (comparison text)

termination character

text character in ASCII, 1 word flag/ASCII-Zeichen

numerical value as binary number on a word flag (≤ 32767)
two-digit telegram number

# = number identification (max. 5 # per number)
* = character identification
<CR> = freely selectable termination character, f.i. CR

Timing diagram for EMAS:

message
acknowledgment
ready
invalid
enabled
XON
XOFF
1. F003
2. F003
waiting cycle
3. F003
1. run
4. F003
2. run
5. F003
acknowledgment cycle
6. F003
7. F003
waiting loop for acknowledgment

8.2

6-22
PROCONIC 5 state 2.88
Receipt of ASCII characters

Module type

Run time:
1200 µs

Optional run time: see text

Value of the past: 3+n words (n = number of the useful informations)

Phases of EMAS

<table>
<thead>
<tr>
<th>call</th>
<th>action</th>
<th>acknowledgment</th>
<th>ready</th>
<th>invalid</th>
<th>telegram number</th>
<th>usable information</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>enable receipt</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>pre-status</td>
<td>pre-status</td>
</tr>
<tr>
<td>2+x</td>
<td>wait for telegram</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>pre-status</td>
<td>pre-status</td>
</tr>
<tr>
<td>3</td>
<td>receive telegram</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>pre-status</td>
<td>pre-status</td>
</tr>
<tr>
<td>4</td>
<td>comparison result</td>
<td>0</td>
<td>1</td>
<td>0/1</td>
<td>0/n</td>
<td>memory content</td>
</tr>
<tr>
<td>5+y</td>
<td>wait for acknowledgment</td>
<td>0</td>
<td>1</td>
<td>0/1</td>
<td>0/n</td>
<td>memory content</td>
</tr>
<tr>
<td>6+z</td>
<td>process of acknowledgment</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0/n</td>
<td>memory content</td>
</tr>
<tr>
<td>1</td>
<td>enable receipt</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0/n</td>
<td>memory content</td>
</tr>
</tbody>
</table>

Cycle time of EMAS:
The processing of a complete order requires a binary call of the module.

The calculation of the typical cycle time is based on the following cycles.

a. Comparison

Basic run time: \( t_1 = 1200 \mu s \)

Cycle time for a number: \( t_2 = 800 \mu s \)
Numbers received are converted into a binary number and filed in an intermediate memory.

Cycle time for ASCII characters: \( t_3 = 70 \mu s \)
ASCII characters received are filed in an intermediate memory.

Cycle time for direct comparison: \( t_4 = 40 \mu s \)
Characters received are compared with the filed comparison telegram.

Cycle time for search: \( t_5 = 35 \mu s \)
In case of an invalid comparison the next telegram is searched for.

The total run time of a comparison cycle (in relation to 1 comparison text) is:

\[ T = t_1 + i \times t_2 + j \times t_3 + k \times t_4 \]

where \( i \) = number of numbers,

\( j \) = number of ASCII characters and

\( k \) = number of direct comparison.

b. Output

Basic run time: \( t_6 = 1200 \mu s \)

Cycle time for the value output: \( t_7 = 100 \mu s \)
Storage of the filed values onto the target variable.

The total run time of the output cycle is:

\[ T = t_6 + m \times t_7 \]

whereas \( m \) = number of the values outputs.

c. Waiting cycle

Basic run time: \( t = 800 \mu s \)
Wait for characters resp. for the acknowledgment.

d. Acknowledgment cycle

Basic run time: \( t = 1022 \mu s \)
Reset of the signals "invalid" and "ready", release of the module to receive further characters.

Example:

01AB###

Comparison:

\[ T = t_1 + 2 \times t_4 + 1 \times t_3 + 3 \times t_2 = 2290 \mu s \]

Output:

\[ T = t_6 + 4 \times t_7 = 1600 \mu s \]

Modules with values of the past have to be used with different module numbers.
Indirect write

Module type

Run time: 500 μs
Value of the past: 0 words

```
2
source address
3
distance
4
target address
```

<table>
<thead>
<tr>
<th>Element</th>
<th>Kind of data</th>
<th>Meaning of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td># B...F020</td>
<td>module number and module type</td>
</tr>
<tr>
<td>2</td>
<td>&quot;....&quot;</td>
<td>input E2 (word operand): source address</td>
</tr>
<tr>
<td>3</td>
<td>&quot;....&quot;</td>
<td>input E3 (word operand): distance (offset)</td>
</tr>
<tr>
<td>4</td>
<td>&quot;....&quot;</td>
<td>input E4 (word operand): target address</td>
</tr>
</tbody>
</table>

The following operands are for the planning of the module:

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
<th>Operands allowed with</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Module number and module type</td>
<td>07 WP 84</td>
</tr>
<tr>
<td></td>
<td></td>
<td>! BA, = BA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16-63</td>
</tr>
<tr>
<td>2</td>
<td>Source address</td>
<td>07 ZE 86</td>
</tr>
<tr>
<td></td>
<td></td>
<td>! BS, = BS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16-63</td>
</tr>
<tr>
<td></td>
<td></td>
<td>! BA, = BA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>00-63</td>
</tr>
<tr>
<td>3</td>
<td>Distance (offset). Values from 0 to 127 permitted; if the distance is higher,</td>
<td>random word and bit operands with</td>
</tr>
<tr>
<td></td>
<td>the leading digits are cut off; the analog value is considered to be integral.</td>
<td>attributes permissible,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>random word and bit operands with</td>
</tr>
<tr>
<td></td>
<td></td>
<td>attributes permissible.</td>
</tr>
<tr>
<td>4</td>
<td>Target address</td>
<td>MW..., MA..., AW..., AA...</td>
</tr>
</tbody>
</table>

# corresponds to ! or =
^ corresponds to a letter
. corresponds to a number
Indirect write
Module type

Run time: 500 µs
Value of the past: 0 words

When calculating the group number the type of the variable (MA resp. MW) is taken into account. Analog variables have even group numbers. Integer variables have odd group numbers.
The use as time controlled module is not permitted.

For the administration of data and parameter fields a simple addressing of the field addresses is required. The addressing is not done by a PRO-CONTIC b variable, but by a relative address (distance).
The module transfers the content of a source address onto a variable (target address), identified by basis address and distance.

Calculation of the target address:

\[ \text{basis address} \times 00 + \text{distance} = 5 \]
\[ \text{target address} \times 05 \]

Example:

<table>
<thead>
<tr>
<th>Module</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>BA20,F020</td>
<td>123456</td>
</tr>
<tr>
<td>MW09,00</td>
<td>000005</td>
</tr>
<tr>
<td>MW01,00</td>
<td></td>
</tr>
<tr>
<td>MW03,00</td>
<td></td>
</tr>
</tbody>
</table>

The such parametrized module IDS is equivalent to the command:

\[ \text{MW 09,00} = \text{MW 03,05} \]

with the same content as in the above shown example.
Indirect read

Module type

Run time: 500 μs
Value of the past: 0 words

<table>
<thead>
<tr>
<th>Element</th>
<th>Kind of data</th>
<th>Meaning of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td># B*...F021</td>
<td>module number and module type</td>
</tr>
<tr>
<td>2</td>
<td>**...</td>
<td>input E2 (word operand): offset</td>
</tr>
<tr>
<td>3</td>
<td>**...</td>
<td>input E3 (word operand): basis address</td>
</tr>
<tr>
<td>4</td>
<td>**...</td>
<td>input E4 (word operand): target address</td>
</tr>
</tbody>
</table>

The following operands are for the planning of the module:

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
<th>Operands allowed with 07 WP 84</th>
<th>Operands allowed with 07 ZE 86</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Module number and module type</td>
<td>16-63</td>
<td>! BA, = BA</td>
</tr>
<tr>
<td>2</td>
<td>Distance (offset). Values from 0 to 127 permitted. If the distance exceeds that, the leading digits are cut off. The analog value is interpreted as integral.</td>
<td>16-63</td>
<td>! BS, = BS</td>
</tr>
<tr>
<td>3</td>
<td>Basis address</td>
<td>00-63</td>
<td>! BA, = BA</td>
</tr>
<tr>
<td>4</td>
<td>Target address</td>
<td>random word and bit operands with attributes permitted.</td>
<td>random allocatable word and bit operands with attributes permitted.</td>
</tr>
</tbody>
</table>

# corresponds to ! or =
* corresponds to a letter
. corresponds to a number
Indirect read
Module type

Run time: 500 µs
Value of the past: 0 words

The use as time controlled module is not permitted.

The module transfers the content of a word variable, defined by basis address and offset (distance), onto a given target address.

When calculating the group number the type of the variable (WA resp. MW) is put into account. Analog variables have even group numbers, integer variables have odd group numbers.

Calculation of the group numbers:

<table>
<thead>
<tr>
<th>basis address</th>
<th>MW01.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ Distance</td>
<td>17</td>
</tr>
<tr>
<td>= target address</td>
<td>MW03.01</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>BA17,F021</td>
<td>000004</td>
</tr>
<tr>
<td>MW01.02</td>
<td></td>
</tr>
<tr>
<td>EA00.00</td>
<td></td>
</tr>
<tr>
<td>MA02.00</td>
<td>EA00.04</td>
</tr>
</tbody>
</table>

Example:
Absolute value former

Module type

Run time: 400 μs
Value of the past: 0 words

<table>
<thead>
<tr>
<th>Element</th>
<th>Kind of data</th>
<th>Meaning of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td># B*..F022</td>
<td>module number and module type</td>
</tr>
<tr>
<td>2</td>
<td>**.....</td>
<td>input E2 (word operand): X</td>
</tr>
<tr>
<td>3</td>
<td>**.....</td>
<td>output A3 (word operand): Abs(X)</td>
</tr>
</tbody>
</table>

The following operands are for the planning of the module:

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
<th>Operands allowed with</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>module number and module type</td>
<td>O7 WP 84: ! BA = BA ! BS = BS 16-63 00-63</td>
</tr>
<tr>
<td>2</td>
<td>random value</td>
<td>O7 ZE 85: ! BA = BA random word operands and bit operands with attributes</td>
</tr>
<tr>
<td>3</td>
<td>absolute value of the input</td>
<td></td>
</tr>
</tbody>
</table>

# corresponds to ! or =
* corresponds to a letter
. corresponds of number
Absolute value former

Module type

Run time: 400 µs
Value of the past: 0 words

The absolute value is formed from the input E2 and the result is issued over the output A3.

Example:

<table>
<thead>
<tr>
<th>Module</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>! BA16,F022</td>
<td></td>
</tr>
<tr>
<td>MW00,00</td>
<td>- 12345</td>
</tr>
<tr>
<td>MW00,01</td>
<td>12345</td>
</tr>
<tr>
<td>NOPO</td>
<td></td>
</tr>
</tbody>
</table>
Limiter
Module type

Run time: 700 µs
Value of the past: 0 words

```
F023
  2
   X
  3
  high limit
  4
  low limit
  5

<table>
<thead>
<tr>
<th>Element</th>
<th>Kind of data</th>
<th>Meaning of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>B*...,F023</td>
<td>module number and module type</td>
</tr>
<tr>
<td>2</td>
<td>**...,</td>
<td>input E2 (word operand): comparison value</td>
</tr>
<tr>
<td>3</td>
<td>**...,</td>
<td>input E3 (word operand): hi limit</td>
</tr>
<tr>
<td>4</td>
<td>**...,</td>
<td>input E4 (word operand): lo limit</td>
</tr>
<tr>
<td>5</td>
<td>**...,</td>
<td>output A5 (word operand): limited value</td>
</tr>
</tbody>
</table>

The following operands are for the planning of the module:

```
<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
<th>Operands allowed with</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>module number and module type</td>
<td>07 WP 84</td>
</tr>
<tr>
<td>2</td>
<td>input</td>
<td>07 ZE 86</td>
</tr>
<tr>
<td>3</td>
<td>value of the hi limit</td>
<td>BA = 16-63</td>
</tr>
<tr>
<td>4</td>
<td>value of lo limit</td>
<td>BS = 16-63</td>
</tr>
<tr>
<td>5</td>
<td>output, as input value, but within the given</td>
<td>BA = 00-63</td>
</tr>
<tr>
<td></td>
<td>limits</td>
<td></td>
</tr>
</tbody>
</table>

# corresponds to ! or =
* corresponds to a letter
. corresponds to a number
Limiter

Module type

Run time: 700 µs
Value of the past: 0 words

The input value E2 is limited to a set high or low value of the inputs E3 resp. E4, and the value is issued over the output A5. If the input E2 is beyond the given limits, the corresponding limit is issued.

Example:

<table>
<thead>
<tr>
<th>Module</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>! BA16,F023</td>
<td>000350</td>
</tr>
<tr>
<td>MW00.00</td>
<td>001200</td>
</tr>
<tr>
<td>MW00.01</td>
<td>-000200</td>
</tr>
<tr>
<td>MW00.02</td>
<td>000350</td>
</tr>
</tbody>
</table>
Complementary former (1-complement)

Module type

Run time: 500 µs
Value of the past: 0 words

<table>
<thead>
<tr>
<th>Element</th>
<th>Kind of data</th>
<th>Meaning of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td># B*,...,F024</td>
<td>module number and module type</td>
</tr>
<tr>
<td>2</td>
<td>**,...</td>
<td>input E2 (word operand or bit operand): X</td>
</tr>
<tr>
<td>3</td>
<td>**,...</td>
<td>output A3 (word operand or bit operand): Y</td>
</tr>
</tbody>
</table>

The following operands are for the planning of the module:

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
<th>Operands allowed with</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>module number and module type</td>
<td>07 WP 84</td>
</tr>
<tr>
<td></td>
<td></td>
<td>! BA, = BA 16-63</td>
</tr>
<tr>
<td>2</td>
<td>random value</td>
<td>07 ZE 86</td>
</tr>
<tr>
<td></td>
<td></td>
<td>! BS, = BS 16-63</td>
</tr>
<tr>
<td>3</td>
<td>1-complement of the input value</td>
<td>random word or bit operands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>random word or bit operands</td>
</tr>
</tbody>
</table>

# corresponds to ': or =
* corresponds to a letter
. corresponds to a number
Complementary former (1-complement)

Module type

Run time: 500 µs
Value of the past: 0 words

The 1-complementary is formed of the input E2 and the result is issued over the output A3.

<table>
<thead>
<tr>
<th></th>
<th>B15</th>
<th>B14</th>
<th>B13</th>
<th>B12</th>
<th>B11</th>
<th>B10</th>
<th>B09</th>
<th>B08</th>
<th>B07</th>
<th>B06</th>
<th>B05</th>
<th>B04</th>
<th>B03</th>
<th>B02</th>
<th>B01</th>
<th>B00</th>
</tr>
</thead>
<tbody>
<tr>
<td>MW01,00 (input E2)</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MW01,02 (output A3)</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Example:

If bit operands are used, the module operates as an inverter.

<table>
<thead>
<tr>
<th>Module</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>BA16,F024</td>
<td>00000 (Dec.) 0000 (HEX.)</td>
</tr>
<tr>
<td>MW01,00</td>
<td>-1 (Dec.) FFFF (HEX.)</td>
</tr>
<tr>
<td>MW01,01</td>
<td></td>
</tr>
<tr>
<td>NPO</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>BA16,F024</td>
<td></td>
</tr>
<tr>
<td>E03,04</td>
<td>0</td>
</tr>
<tr>
<td>A05,07</td>
<td>1</td>
</tr>
<tr>
<td>NPO</td>
<td></td>
</tr>
</tbody>
</table>
PI-Regulator

Module type

Run time: 3400 µs
Value of the past: 2 words

---

**Diagram**

- **F040**
  - W
  - X
  - KP
  - TN/TZ
  - I
  - S
  - R
  - OG (OG)
  - UG (UG)

---

**Table**

<table>
<thead>
<tr>
<th>Element</th>
<th>Kind of data</th>
<th>Meaning of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td># B*...F040</td>
<td>module number and module type</td>
</tr>
<tr>
<td>2</td>
<td><strong>...</strong></td>
<td>input E2 (word operand): W (setpoint)</td>
</tr>
<tr>
<td>3</td>
<td><strong>...</strong></td>
<td>input E3 (word operand): X (actual value)</td>
</tr>
<tr>
<td>4</td>
<td><strong>...</strong></td>
<td>input E4 (word operand): KP (variable parameter)</td>
</tr>
<tr>
<td>5</td>
<td><strong>...</strong></td>
<td>input E5 (word operand): TN/TZ</td>
</tr>
<tr>
<td>6</td>
<td><strong>...</strong></td>
<td>input E6 (word operand): initial value</td>
</tr>
<tr>
<td>7</td>
<td><strong>...</strong></td>
<td>input E7 (bit operand): set to initial value</td>
</tr>
<tr>
<td>8</td>
<td><strong>...</strong></td>
<td>input E8 (bit operand): reset to 0</td>
</tr>
<tr>
<td>9</td>
<td><strong>...</strong></td>
<td>input E9 (word operand): OG high limit</td>
</tr>
<tr>
<td>10</td>
<td><strong>...</strong></td>
<td>input E10 (word operand): UG low limit</td>
</tr>
<tr>
<td>11</td>
<td><strong>...</strong></td>
<td>output A11 (word operand): Y controller output</td>
</tr>
<tr>
<td>12</td>
<td><strong>...</strong></td>
<td>output E12 (bit operand): high limit reached</td>
</tr>
<tr>
<td>13</td>
<td><strong>...</strong></td>
<td>output E13 (bit operand): low limit reached</td>
</tr>
</tbody>
</table>
The following operands are for the planning of the module:

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
<th>Operands allowed with</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>module number and module type</td>
<td>07 WP 84</td>
</tr>
<tr>
<td>2</td>
<td>set point</td>
<td>16-63</td>
</tr>
<tr>
<td>3</td>
<td>actual value</td>
<td>16-63</td>
</tr>
<tr>
<td>4</td>
<td>gain(l)</td>
<td>! BA, = BA</td>
</tr>
<tr>
<td>5</td>
<td>relationship, integral-action time and cycle time</td>
<td>15-63</td>
</tr>
<tr>
<td>6</td>
<td>initial value</td>
<td>! BS, = BS</td>
</tr>
<tr>
<td>7</td>
<td>set to initial value</td>
<td>00-63</td>
</tr>
<tr>
<td>8</td>
<td>reset to 0</td>
<td>! BA, = BA</td>
</tr>
<tr>
<td>9</td>
<td>high limit</td>
<td>random word operands</td>
</tr>
<tr>
<td>10</td>
<td>low limit</td>
<td>random word operands</td>
</tr>
<tr>
<td>11</td>
<td>controller output</td>
<td>random word operands</td>
</tr>
<tr>
<td>12</td>
<td>high limit reached</td>
<td>random bit operands</td>
</tr>
<tr>
<td>13</td>
<td>low limit reached</td>
<td>random bit operands</td>
</tr>
</tbody>
</table>

1) analog value  
2) integral number  
3) binary value

# corresponds to ; or =  
* corresponds to a letter  
. corresponds to a number

If the module is in multiple use, it must be called by different module numbers.

The regulator is capable of being adjusted and reset. If KP is changed an immediate adjustment is done at the output to prevent jumps. A change of TN also does not cause the output to jump.

KP: P-amplification  
TN: integral-action time  
TZ: cycle time

Transmission function:

\[ F = (W - X) \times KP + 1 + 1/T \times KP \times (W - X) \times dt \]

whereas \[ F(p) = K_p \times (1 + pTN/pTN). \]

Priorities: set before reset.

Modules with values of the past have to be used with different module numbers.

Example:

<table>
<thead>
<tr>
<th>Module</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>! BA16,F040</td>
<td>01230</td>
</tr>
<tr>
<td>MW00,00</td>
<td>01200</td>
</tr>
<tr>
<td>MW00,01</td>
<td>- 00100</td>
</tr>
<tr>
<td>MW00,02</td>
<td>00020</td>
</tr>
<tr>
<td>MW00,03</td>
<td>00150</td>
</tr>
<tr>
<td>MW00,04</td>
<td></td>
</tr>
<tr>
<td>M 00,00</td>
<td>0</td>
</tr>
<tr>
<td>M 00,01</td>
<td>0</td>
</tr>
<tr>
<td>MW00,05</td>
<td>- 02000</td>
</tr>
<tr>
<td>MW00,06</td>
<td>00150</td>
</tr>
<tr>
<td>MW00,07</td>
<td>01200</td>
</tr>
<tr>
<td>MW00,08</td>
<td>- 0</td>
</tr>
<tr>
<td>M 00,02</td>
<td>0</td>
</tr>
<tr>
<td>NDPO</td>
<td></td>
</tr>
</tbody>
</table>

Notice:  
TN/TZ < 10 causes instability of the regulator (10 < TN/TZ <= 32766). The higher TN/TZ is, the lower the 1-gain becomes.
Cycle monitoring

Module type

Run time: 270 µs
Value of the past: 0 words

<table>
<thead>
<tr>
<th>Element</th>
<th>Kind of data</th>
<th>Meaning of the element</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>= BA...,F060 <strong>...</strong></td>
<td>module number and module type</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>input E2 (bit operand): turn cycle on/off</td>
</tr>
</tbody>
</table>

The following operands are for the planning of the module:

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
<th>Operands allowed with</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>module number and module type</td>
<td>07 WP 84</td>
</tr>
<tr>
<td></td>
<td></td>
<td>07 TZ 82</td>
</tr>
<tr>
<td></td>
<td></td>
<td>07 2E 86</td>
</tr>
<tr>
<td></td>
<td></td>
<td>! BA = BA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>! BS = BS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>! BA = BA 16-63</td>
</tr>
<tr>
<td></td>
<td></td>
<td>00</td>
</tr>
<tr>
<td>2</td>
<td>turn cycle monitoring on/off</td>
<td>00-63</td>
</tr>
</tbody>
</table>

* corresponds to a letter
. corresponds to a number
Cycle monitoring
Module type

Run time: 270 µs
Value of the past: 0 words

The module F060 is required for cycle monitoring.
If this module is not included in the PROCONTC program, cycle monitoring is not done.
The module may be located at any point in the program.
The monitoring may be turned ON and OFF by the bit parameter (f.i. in break point operation). Turning OFF over the serial interface is not possible. The status of the bit parameter can be simulated by the programming unit.
The bit status of the bit parameter means:

Bit = 0; monitoring ON
Bit = 1; monitoring OFF

The monitoring time is fixed to 250 ms. If this time is exceeded, all output flags (word and bit) are set to zero (the N-signal goes to 1). The following error message is issued over the interface:

Example:

ERROR: PROCONTC CYCLE EXCEEDED 250 MS
The control will not restart before the RUN/STOP switch is actuated. If the switch is turned to STOP, the interface will repeat the error message.

When the SIP module is used, the F060 must be included in the program, independent of the cycle monitoring being turned ON or OFF.
The F060 is also used to control the module bits B16 - B63. When the F060 is present, only one background module is processed per each PROCONTC cycle.

Example:

<table>
<thead>
<tr>
<th>Module</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>! BA16,F060</td>
<td>E 00,00</td>
</tr>
<tr>
<td></td>
<td>NOP0</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Head module
Module type

Run time: 120 μs
Value of the past: 0 words

F127

2 module set
3 clock time
4 NOP0

Element | Kind of data | Meaning of the element
--- | --- | ---
1 | ! BAO0,F127 | module number and module type
2 | ! BAO0,A.. | module set
3 | # 0,abc Ex | clock time, variable
4 | NOP0 | clear word

16 module sets are allowed.
The following operands are for the planning of the module:

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
<th>Operands allowed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>module number and module type</td>
<td>07 WP 84</td>
</tr>
<tr>
<td>2</td>
<td>module set</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>clock time</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>NOPO</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>module set</td>
<td>07 ZE 86</td>
</tr>
<tr>
<td>6</td>
<td>clock time</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>NOPO</td>
<td></td>
</tr>
<tr>
<td>.</td>
<td>module set</td>
<td></td>
</tr>
<tr>
<td>.</td>
<td>clock time</td>
<td></td>
</tr>
<tr>
<td>.</td>
<td>NOPO</td>
<td></td>
</tr>
<tr>
<td>.</td>
<td>! BE</td>
<td></td>
</tr>
</tbody>
</table>

a, b, c = 1 ... 9
x = 1 ... 3
. corresponds to a number

The head module is located following the main program, i.e. after ! PE. The clock time is set by the same constant (0, abcEx) as in case of the setpoint allocation of timers. The shortest interval is 10 ms (0,010EO) for the 07 WP 84 and 20 ms (0,002E1) for the 07 ZE 86. The max. clock time is 655 s (0,655E3). The clock times of the individual module sets must be an integral multiple of the shortest clock time.

Word processor 07 WP 84:

For the number of the module sets (n) it is:

\[
\text{shortest clock time (in ms)} \leq \frac{n}{10}
\]

Shortest required clock time: 30 ms

\[
n = \frac{30}{10} = 3
\]

There are 3 module sets possible.

8 module sets are required:

\[
\text{shortest clock time} = n \times 10
\]

\[
= 8 \times 10
\]

\[
= 80 \text{ ms}
\]

Central processor 07 ZE 86:

For the number of the module sets (n) it is:

\[
n \leq \frac{\text{shortest clock time (in ms)}}{20}
\]

Shortest required clock time: 40 ms

\[
n = \frac{40}{20} = 2
\]

There are 2 module sets possible.

8 module sets are required:

\[
\text{shortest clock time} = n \times 20
\]

\[
= 8 \times 20
\]

\[
= 160 \text{ ms}
\]
Head module
Module type

Run time: 120 µs
Value of the past: 0 words

The module sets are to be arranged in an increasing sequence according to their clock times. Only module sets of the type A.. are permitted. The run time module set must not exceed 10 ms. This time is monitored in the running control, in a given incident an error message is issued and the control is brought to a stop.

The module sets are filed together with the background modules. The program size of all modules of the type A.. may be as big as 1 K-words. This limit is not monitored by the programming units (in case of the word processor 07 WP 84 R202 it is monitored, in a given incident an error message is issued during normalization).

Example:

<table>
<thead>
<tr>
<th>07 WP 84</th>
<th>07 ZE 86</th>
</tr>
</thead>
<tbody>
<tr>
<td>BA00,F127</td>
<td>BA00,F127</td>
</tr>
<tr>
<td>BA00,A000</td>
<td>BA00,A000</td>
</tr>
<tr>
<td>#0,010E1</td>
<td>#0,020E1</td>
</tr>
<tr>
<td>NOP0</td>
<td>NOP0</td>
</tr>
<tr>
<td>BA00,A001</td>
<td>BA00,A001</td>
</tr>
<tr>
<td>#0,020E1</td>
<td>#0,040E1</td>
</tr>
<tr>
<td>NOP0</td>
<td>NOP0</td>
</tr>
<tr>
<td>BA00,A002</td>
<td>BA00,A002</td>
</tr>
<tr>
<td>#0,030E1</td>
<td>#0,060E1</td>
</tr>
<tr>
<td>NOP0</td>
<td>NOP0</td>
</tr>
<tr>
<td>BE</td>
<td>BE</td>
</tr>
</tbody>
</table>

Explains see section 5.8.
The formal parameters RA.. and RW.. (word parameters) and R.. (bit parameters, only in connection with O7 WP 84 eff. R02) are location occupiers and are replaced by the actual parameters in the processing, according to the given sequence. The operands may be defined by parameters. As actual parameters only word variables (EA....., EW....., AA....., AW....., MW....., MA....., KW..... und KA.....) are accepted. The actual parameters are transferred between module call and module definition. Thus it is possible to fix the input and output values after the module call, so to use a module definition by multiple module calls (BA16 ...BA63) with different actual parameters. Herewith the corresponding module definition must be written once only. The actual parameters are transferred to the corresponding formal parameters in the module definition.

The module call and the following definition of the actual parameters, i.e. the value actually assigned by a word variable, must meet a certain format.

Example:

<table>
<thead>
<tr>
<th>Kind of data</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>BA17,A12</td>
<td>Start of module</td>
</tr>
<tr>
<td>EA04,03</td>
<td>RA00</td>
</tr>
<tr>
<td>EA02,04</td>
<td>RA01</td>
</tr>
<tr>
<td>MA02,04</td>
<td>RA02</td>
</tr>
<tr>
<td>MW01,05</td>
<td>RW03</td>
</tr>
<tr>
<td>NOP0</td>
<td>end of parameter list</td>
</tr>
</tbody>
</table>

A maximum of 32 formal parameters (RA and RW accumulated) may be assigned. The formal parameters must be written in increasing sequence from 00 to 31. The actual parameters are allocated to the corresponding formal parameters in the sequence from the top down. Multiple use of a formal parameter in a module call is possible. Always the agreed-on actual parameter is taken over.

The formal parameters RA00 to RW03 were loaded with the actual parameters EA04,03 to MW01,05 in the sequence as listed by the latter. The processor jumps into the module definition at the end of the program and processes the module program with the loaded parameters included there.

In the module definition it would be written:

```plaintext
PB A012
MW01,00 + RW03 = RW03  
MA00,00 - RA01 + RA00 = RA02  
BE
```

The altered values of the formal parameters RA02 and RW03 are allocated accordingly. The sequence of the formal parameters in the module definition is random. Each formal parameter can be polled or allocated multiple.

This parametrization is identical with:

```plaintext
PB_A12
MW01,00 + MW01,05 = MW01,05  
MA00,00 - EA02,04 + EA04,03 = MA02,04  
BE
```

Since the actual parameters cannot include attributes, such data must be transferred by word flags.

```plaintext
BE 02,00 = MW01,07  
BA17,A012
MW01,07 (poll of RW03)
NOP0
MW01,07 = BA 04,00 (allocation)
BE
PB A012
RW00 + MW01,05 = RW00
BE
```

At this point it has to be considered, that BS modules are processed in parallel and the expected result MW01,07 is polled by a module start (BA17,A12 or BS17,A12) at a time, when the word processor has not yet reached the result.
Allocations outside of modules, based on results generated by the word processor, are possible after module call (! BA..,A.. or = BA..,A..) only.

Table for the operands accepted as parameters by the word processor

<table>
<thead>
<tr>
<th>O7 WP 84 R101</th>
<th>O7 WP 84 R202</th>
<th>O7 ZE 86</th>
</tr>
</thead>
<tbody>
<tr>
<td>word operands</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EW...</td>
<td>EW...</td>
<td>EW...</td>
</tr>
<tr>
<td>EA...</td>
<td>EA...</td>
<td>EA...</td>
</tr>
<tr>
<td>AW...</td>
<td>AW...</td>
<td>AW...</td>
</tr>
<tr>
<td>AA...</td>
<td>AA...</td>
<td>AA...</td>
</tr>
<tr>
<td>MW...</td>
<td>MW...</td>
<td>MW...</td>
</tr>
<tr>
<td>MA...</td>
<td>MA...</td>
<td>MA...</td>
</tr>
<tr>
<td>KA...</td>
<td>KA...</td>
<td>KA...</td>
</tr>
<tr>
<td>KW...</td>
<td>KW...</td>
<td>KW...</td>
</tr>
<tr>
<td>#W...</td>
<td>#W...</td>
<td>#W...</td>
</tr>
<tr>
<td>attributes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>none</td>
<td>a, w</td>
<td>none</td>
</tr>
<tr>
<td></td>
<td>b, c</td>
<td></td>
</tr>
<tr>
<td>bit parameters</td>
<td></td>
<td></td>
</tr>
<tr>
<td>none</td>
<td>E 00,00...E 15,15</td>
<td>all</td>
</tr>
<tr>
<td></td>
<td>A 00,00...A 15,15</td>
<td></td>
</tr>
<tr>
<td></td>
<td>M 00,00...M 07,15</td>
<td></td>
</tr>
<tr>
<td></td>
<td>not allowed:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>E ....&gt;15,15</td>
<td></td>
</tr>
<tr>
<td></td>
<td>A ....&gt;15,15</td>
<td></td>
</tr>
<tr>
<td></td>
<td>M' ....</td>
<td></td>
</tr>
<tr>
<td></td>
<td>a#W...</td>
<td></td>
</tr>
</tbody>
</table>

Notice:
The parameter list must be terminated by a NOPO.
The parameters are entered without linkage sign.

There are 32 formal parameters RAOO to RA31 and 32 formal parameters RM00 to RM31.

RA and RW are exclusive, i.e. a total of 32.

At a module start (BS) a maximum of 16 parameters may be transferred. Effective O7 WP 84 R202 additional 16 bit parameters R.. are accepted.

At a module call (BA) a maximum of 32 word parameters may be transferred. Effective O7 WP 84 R202 and in case of the O7 ZE 86 additional 32 bit parameters R.. are accepted.

Example:

In the following example a module BA16,A000 is called unconditioned, i.e. it is forced to be processed in each cycle.

After the module call the parameters RAOO with EAO6,00, RA01 with AA08,00 and RW02 with MWO1,00 are loaded. In the module address W007B the first parameter (RAOO), which has an analog value, is compared with a constant previously converted into an analog format, to be above 800 (8 V) or below 100 (1 V). The binary flag M00,00 indicates that the value is above 800 (8 V). Flag M00,01 indicates that the value is below 100 (1 V).

RA00 is copied into the parameter 2 (RAO1) and issued. After that the value RAO0 is converted into a binary value and filed in a 3. parameter RW02 and the value of this is copied into the word flag MWO1,00.

In the addresses 6 to 15 the result of the comparison is issued. A 13,08 means, that the value is lower. Equivalent to that the result in the case of A 13,10 is higher. At A 13,09 the result is within the given comparison values. It can be selected via the input E 09,08 to issue the binary value of the flag MWO1,00 as BCD value over binary output channels.

The following modules BA17 and BA18 work in an analogous way, with the difference, that different input and output channels are used, which are processed by the same processing module.

The module BA19 compares two analog values with each other and indicates, if the first value exceeds the second.
Attributes result in format adaption of the operands to the format used in computation.

The word processor can calculate in 2 different data formats:
- 16 bits binary range: decimal +/- 32767
- 16 bits analog range: analog +/- 7,999 (MICAS)

In the first case the range serves the purpose of arithmetic operations in the integer range.

The second case serves the purpose of processing analog values, whereby the number itself indicates a percentage of the analog value.

Analog value: 10 V equals to 1,000 (MICAS) = 100%

An arithmetic operation out of integer and analog values is not permitted. A format adaption has to be carried out if this is necessary. To do so the attributes "a, w" serve as aiding means.

### 8.1 Possible format changes

<table>
<thead>
<tr>
<th>Condition</th>
<th>Allocation</th>
<th>Conversion from</th>
<th>Conversion to</th>
</tr>
</thead>
<tbody>
<tr>
<td>!, +, −, &quot;&quot;:</td>
<td>w</td>
<td>analog word</td>
<td>binary word</td>
</tr>
<tr>
<td></td>
<td>w</td>
<td>binary word</td>
<td>analog word</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
<td>binary word</td>
<td>analog word</td>
</tr>
<tr>
<td></td>
<td>a</td>
<td>analog word</td>
<td>binary word</td>
</tr>
<tr>
<td>b</td>
<td>b</td>
<td>binary word</td>
<td>16 single binary bits</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>16 single binary bits</td>
<td>binary word</td>
</tr>
<tr>
<td>c</td>
<td>c</td>
<td>16 bits BCD</td>
<td>binary word</td>
</tr>
<tr>
<td></td>
<td>c</td>
<td>binary word</td>
<td>16 bits BCD</td>
</tr>
</tbody>
</table>

When issuing ASCII characters over the serial interface by the logging modules F000 and P000 to P127 the attribute d can be used.

### Example:

![Image](image.png)

The analog value of the analog input EA04,01 is added to the value in the word flag MW03,07. Since this is only available in the binary range, it is converted to the analog format by means of the attribute "a".

Binary numbers can be obtained by binary input units. The input value applied to these units can be both BINARY and BCD coded (decade selector switch). BINARY-BCD format adaption can be carried out by using the attributes "b, c".

### 8.2 Transfer and output of data

Example:

![Image](image.png)

Transfer:

The binary value in the word flag MW01,03 is added to the BCD value at the following inputs:

<table>
<thead>
<tr>
<th>Operand</th>
<th>Bit</th>
<th>4. digit</th>
<th>3. digit</th>
<th>2. digit</th>
<th>1. digit</th>
<th>Sign</th>
</tr>
</thead>
<tbody>
<tr>
<td>E 00,00</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E 00,01</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E 00,02</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E 00,03</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E 00,04</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E 00,05</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E 00,06</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E 00,07</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E 01,00</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E 01,01</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E 01,02</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E 01,03</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E 01,04</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E 01,05</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E 01,06</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E 01,07</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The attribute "c" means that the content of 16 single bits with the start address E 00,00, which are available in BCD coded form, are recorded in binary form and packed into one word.

Transfer:
In the allocation to the output unit, the total is applied in binary form to the output channels as a 16-bit-word.

<table>
<thead>
<tr>
<th>Operand</th>
<th>Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>A 03,00</td>
<td>00</td>
</tr>
<tr>
<td>A 03,01</td>
<td>01</td>
</tr>
<tr>
<td>A 03,02</td>
<td>02</td>
</tr>
<tr>
<td>A 03,03</td>
<td>03</td>
</tr>
<tr>
<td>A 03,04</td>
<td>04</td>
</tr>
<tr>
<td>A 03,05</td>
<td>05</td>
</tr>
<tr>
<td>A 03,06</td>
<td>06</td>
</tr>
<tr>
<td>A 03,07</td>
<td>07</td>
</tr>
<tr>
<td>A 04,00</td>
<td>08</td>
</tr>
<tr>
<td>A 04,01</td>
<td>09</td>
</tr>
<tr>
<td>A 04,02</td>
<td>10</td>
</tr>
<tr>
<td>A 04,03</td>
<td>11</td>
</tr>
<tr>
<td>A 04,04</td>
<td>12</td>
</tr>
<tr>
<td>A 04,05</td>
<td>13</td>
</tr>
<tr>
<td>A 04,06</td>
<td>14</td>
</tr>
<tr>
<td>A 04,07</td>
<td>15</td>
</tr>
</tbody>
</table>

The attributes "b" and "c" do not consider the area limit (bank change). The group number are not automatically raised over the area limit.

E.g. ! bM'47,00 à M'47,00 and M'48,00
not allowed

M'47,07 M'48,07
Data are correct not correct

Following addresses (area limit) are not allowed:
M'...... E ...... oder A......
07, ...
15, ...
23, ...
31, ...
39, ...
47, ...
55, ...
63, ...

8.3 Conversion of an analog value to a binary value

\[ wEAO2,00 + MW01,04 = MW01,05 \]

(binary) (binary) (binary)

1254 + 1254 =

after Conversion with w:

00540 + 1254 = 1794

Explanation:

10 V = 100 % = 1,0 (MICAS)
1000dec = 100 % = 1,0 (MICAS)
The content of the analog input EA02,02 is converted to a binary value and added to a binary value located in the flag MW01,04. The result of the binary addition is passed directly onto the word flag MW01,05.

\[
\begin{align*}
! & \text{ EA04,01 } + & \text{ aMW03,07 } = & \text{ AA08,00} \\
5.4 \text{ V } + & 120_{\text{deg}} = & 0,540 \\
0.540 & + 0.120 = & 0,660 \\
54 \% & + 12.0 \% = & 66 \% \\
5.4 \text{ V } + & 1.2 \text{ V } = & 6,6 \text{ V}
\end{align*}
\]

The attribute "a" converts the content of the word flag (16 bit signed integer) to the analog format used in the calculation.

### Conversion bit-word resp. word-bit:

<table>
<thead>
<tr>
<th>Condition</th>
<th>Allocation</th>
<th>Conversion from</th>
<th>Conversion to</th>
</tr>
</thead>
<tbody>
<tr>
<td>! + - * :</td>
<td>=</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>16 single binary bits</td>
<td>binary word</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>binary word</td>
<td>16 single binary bits</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>16 bits BCD</td>
<td>binary word</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>binary word</td>
<td>16 bits BCD</td>
<td></td>
</tr>
</tbody>
</table>

### 8.5 Attribute d

(ef. 07 WP 84 R302 and 07 ZE 86 R401)

By use of the attribute d the content of the low-byte of a word flag can be issued over the serial interface as ASCII character without conversion. The attribute can only be used in connection with logging and text output modules. The word flags MW...., MA...., AW.... and AA.... can be converted by the attribute d.

Example:

In the word flag MW01,00 0030H (ASCII character for zero) is included.

### Output of word values via binary output and input units

<table>
<thead>
<tr>
<th>MW01,03</th>
<th>= CE00,00 = MW01,04</th>
</tr>
</thead>
<tbody>
<tr>
<td>setpoint1 correction setpoint 2</td>
<td></td>
</tr>
<tr>
<td>A 03,00</td>
<td>= CA05,00</td>
</tr>
<tr>
<td>Output 1</td>
<td>Output 2</td>
</tr>
</tbody>
</table>

The attribute "c" resp. "b" (with the corresponding operation) causes the word processor to process 16 single bits with the specified start address as one binary value.

In the given example the output channels (E 00,00 to E 00,07 and E 01,00 to E 01,07) resp. the output channels (A 03,00 to A 03,07 and A 04,00 to A 04,07) are combined. The operand specified in the program (E 00,00 resp. A 05,00) identifies the least significant bit of the binary word.

The attribute "b" passes on the bit pattern to a 1:1 scale, while the attribute "c" additionally carries out a BCD conversion. (Internally calculation is always in binary or analog.)
9 Constants

9.1 General Information

Indirect and direct constants are distinguished.

Word constants may be linked to all word operators (except = (THAN)).

9.2 Indirect constants

K*... constant number
W = word and A = analog constant

The following constant numbers are allowed for indirect constants:

KW... indirect word constant 00 to 63
KA... indirect analog constant 000 to 127

The numerical values allowed for the indirect constants are:

at KW = -32767 ... +32767
at KA = -7,999 ... +7,999

Indirect constants are filed in the constant memory. During system initialization, f.i. after applying power, the word processor fetches the content of the constant memory to its processing memory.

The number of possible indirect constants is limited. Constants are fixed values in the word range, which are needed recurrently in the program. If, for example, regulators are programmed, whose behavior is not defined until during the commissioning phase, a constant must be inserted in the program at the points at which the dynamic behavior of the regulator is defined. The constant is entered directly into the programming unit during commissioning.

Depending on the type of the indirect constant the numerical range changes.

KW053 = -12345
KA053 = 0,943

The control then processes with this value at the corresponding points in the program. The following is now standing in the program:

! EA00,00 + KA053 = AA00,00

So it is possible, changing the constant, as for example a regulator parameter, to change the dynamical characteristics of a complex programming part.

Behind the program end the programming units automatically generate the constant memory. The constant memory is limited.

9.3 Direct constants

# 0,ab Ex direct timer constant, timer value 0,01 s ... 9900s (= 16 min 30 s)

whereby a,b = 0 ... 9, x = 0 ... 3

# 0,abcE x direct timer constant, timer value 0,01 s ... 99900 s (= 27 h 45 min)

whereby a,b,c = 0 ... 9, x = 1 ... 5

#W abcd direct word constant, also used as counter constant.
numerical value /-32767 ... +32767

#D abcdefg Double word constant.
numerical value /-9999999 .../+9999999

"text Text constant (ASCII constant).
All ASCII signs or hexadecimal numbers between 01H and 7FH are permitted.

There are no direct analog constant.
With an attribut can a word constant be processed as an analog constant (a#W......).

Example

a#W01000 = + 1,000 = 100%
a#W00500 = + 0,500 = 50%
Direct constants are numerical values designated in the program by the character # and filed directly in it. The number of possible direct constants in the program is limited only by the available memory capacity.

Thus it is possible to write the following:

! EA00,00 + a#W 00010 = EA00,00

where #W 00010 represents the constant.

The direct constants also include text constants. It is thus possible to firmly anchor texts in the program, such as malfunction messages, and issue outputs of them when required.

! E 00,00 = BS16,FO00 #"NOTAUS

where #"NOTAUS (emergency stop) represents the text constant.

9.4 Bit constants

#B0 bit constant with status 0
#B1 bit constant with status 1

By bit constants (#B0 and #B1) a certain status can be allocated to operands, i.e., independent of outside conditions a certain status can be allocated to an output, for example after having processed a module. Besides that the bit constants are reasonably used as actual parameters in the case of function modules.
There are various possibilities for issuing data over the serial interface.

- Logging functions
- SIP module
- EMAS module
- monitoring functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>07 TZ 82 R101</th>
<th>07 TZ 82 eff. R201</th>
<th>07 WP 84 R101</th>
<th>07 WP 84 eff. R202</th>
<th>07 ZE 86 eff. R301</th>
</tr>
</thead>
<tbody>
<tr>
<td>F000 resp. P000...P127</td>
<td>logging functions</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$P</td>
<td>logging format</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$B...&lt;CR&gt;</td>
<td>set baud rate</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$ZY=&lt;CR&gt;</td>
<td>poll cycle time</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$Z=&lt;CR&gt; (.. = 0 bzw 1)</td>
<td>turn off cycle time</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$I1,...=abcEx&lt;CR&gt;</td>
<td>set time actual value</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$I1,...=&lt;CR&gt;</td>
<td>poll time actual value</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$ST,...=abcEx&lt;CR&gt;</td>
<td>set time setpoint</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$ST,...=&lt;CR&gt;</td>
<td>poll time setpoint</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$I1,...=&lt;D&gt;</td>
<td>poll time setpoint (2)</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$ST,...=&lt;D&gt;</td>
<td>poll time setpoint (2)</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$I2,...=abcdef&lt;CR&gt;</td>
<td>set counter actual value</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$I2,...=&lt;CR&gt;</td>
<td>poll counter actual value</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$SZ,...=abcdef&lt;CR&gt;</td>
<td>set counter setpoint</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$SZ,...=&lt;CR&gt;</td>
<td>poll counter setpoint</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$I2,...=&lt;D&gt;</td>
<td>poll actual countervalue (2)</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$SZ,...=&lt;D&gt;</td>
<td>poll actual countervalue (2)</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F001 SIP</td>
<td>input of word variables</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F003 EMAS</td>
<td>input of ASCII characters</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$EA,...=&lt;CR&gt;</td>
<td>display analog input</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$EA,...=2,123&lt;CR&gt;</td>
<td>set analog input</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$AA,...=&lt;CR&gt;</td>
<td>display analog output</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$AA,...=2,123&lt;CR&gt;</td>
<td>set analog output</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$MA,...=&lt;CR&gt;</td>
<td>display analog flag</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$MA,...=2,123&lt;CR&gt;</td>
<td>set analog flag</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$AW,...=&lt;CR&gt;</td>
<td>display word output (*)</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$AW,...=-23456&lt;CR&gt;</td>
<td>set word output (*)</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$MW,...=&lt;CR&gt;</td>
<td>display word flag</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$MW,...=2,123&lt;CR&gt;</td>
<td>set word flag</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M'...=&lt;CR&gt;</td>
<td>display bit flag</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M'=0 or&lt;CR&gt;</td>
<td>set bit flag</td>
<td>(1)</td>
<td></td>
<td>(1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$F&lt;CR&gt;</td>
<td>error polling</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$V&lt;CR&gt;</td>
<td>version number</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

a, b, c, d, e, f, g, x and . = numbers
(* = not available as hardware unit. May be used as word flag.
(1 = possible effective word processor
07 WP 84 R302.
(2 = continuous
10.1 Logging functions

Output of texts and values over logging modules is possible. The call for that is done by the program in the PROCONTIC b via the text module F000 resp. the logging modules P000 ... P127. All ASCII characters may be transmitted. Thus it is possible to control printers, cassette recorders or screens.

Change of the baud rate:

$530<CR> corresponds to 300 Baud
$560<CR> corresponds to 600 Baud
$512<CR> corresponds to 1200 Baud
$824<CR> corresponds to 2400 Baud
$848<CR> corresponds to 4800 Baud
$960<CR> corresponds to 9600 Baud

The baud rate is changed immediately after entering. There is no output of an acknowledgment. A faulty input does not cause a change but the output of a "?". The baud rate selected is filed in the buffered RAM (in case of the central processor 07 ZE 86 the selector switch BAUD must be in position P).

10.2 Logging format

The interface of the digital timer is in standard mode with the transmission format 0 (8 data bits, no parity check and 1 stop bit) after power is applied the first time resp. as long as no other mode was selected.

By input of $P=1<CR>$ over the serial interface the transmission format 1 (7 data bits, even parity, 2 stop bits) is selected.

By input of $P=0$ the transmission format 0 is re-selected.

A faulty input does not cause the transmission rate to change but a "?" is issued over the serial interface.

The transmission format entered is filed in the buffered RAM until it is changed by a new input.

10.3 Selection of the baud rate

The interface is in standard mode with the baud rate of 300 baud after power is applied the first time resp. as long as no other mode was selected.

Attention:
In order for the programming unit to exchange data with the interface the baud rates of the corresponding unit and of the interface must match. This means that at the first time use of a unit in the head ($S$HEAD) of the programming unit the baud rate 300 baud must be selected. In TTY mode the question for half-duplex operation (J/N, Y/N) must be answered by a N.

Poll: $ZY<CR>$ Output: 020 MS

10.4 Polling of the maximum cycle time

The run time of each PROCONTIC cycle is registered by the word processor 07 WP 64 effective version R201 resp. by the central processor 07 ZE 86. The maximum cycle time is filed. Thus the user may poll the longest cycle time occurred from starting the control up to the polling point. The time is issued with a tolerance of 10 ms. This function is based on the use of the function module F060 ZYK (cycle monitoring).

10.5 Cycle monitoring function enable/disable

The digital timer 07 TZ 82 monitors the cycle time not to exceed 250 ms. The monitoring function can be enabled resp. disabled over the serial interface.

$Z<CR>$ cycle monitoring disable
$Z<CR>$ cycle monitoring enabled

10.6 Setting of time values over the serial interface

$ST \ 00,00 = abc \times <CR>$ setpoint
$ST \ 00,00 = abc \times <CR>$ actual value

where x = 1 to 5 and abc = 0 to 9

Time value in seconds: $0,abc \times 10^{x}$

If a new setpoint is entered, the CRT displays the following generated by the digital timer 07 TZ 82 R101:

$SS \ T \ xx,yy = value \ 1$

with yy = 00 to 15 and xx = all group numbers allowed for the operand.
If a new actual value or setpoint is entered, the CRT displays the following caused by the digital timer 07 TZ B2 R201 resp. the central processor 07 ZE 86:

$ NN xx,yy = value 1
   value 2

with NN = ST resp. IT, yy = 0 to 15 and xx = all group numbers allowed for the operand.

The first line shows the characters entered. Value 2 shows the value, which is to be issued caused by a read command. Value 2 must be even with value 1, unless a faulty input happened (to many characters, leading nulls missing etc.)

10.7 Polling of time values over the serial interface

Input:
$S T 00,00 = <CR> setpoint
$T T 00,00 = <CR> actual value polling

10.8 Continuous polling of the time values over the serial interface

$S T 00,00 = <CR> setpoint
$T T 00,00 = <CR> actual value polling

If there is no processing (selector switch of the central processor in position 'stop'), the actual value is issued once only. If the selector switch is turned to 'RUN' or the command is given by the running control, an output is generated every 2 seconds in the form

ST 00,00 = <value>

or

IT 00,00 = <value>

The continuous reading of a time is interrupted by actuation of any key.

10.9 Setting of counter values over the serial interface

$ SZ 00,00 = abcd <CR> (setpoint)
$ Z 00,00 = abcd <CR> (actual value)

where abcd = 0 to 9

The counter value is: abcd

If a new setpoint or actual value is entered, the CRT displays the following:

$ NN xx,yy = value 1
   value 2

with NN = SZ resp. IZ, yy = 0 to 15 and xx = all group numbers allowed for the operand.

The first line shows the characters entered. Value 2 shows the value, which is to be issued caused by a read command. Value 2 must be even with value 1, unless a faulty input happened (to many characters, leading nulls missing etc.)

10.10 Polling of counter values over the serial interface

$ SZ 00,00 = <CR> set point polling
$ IZ 00,00 = <CR> actual value polling

10.11 Continuous polling of the counter values over the serial interface

$ SZ 00,00 = D <CR> set point polling
$ IZ 00,00 = D <CR> actual value polling

If there is no processing (selector switch on the central processor in position 'stop'), the actual value is issued once only. If the selector switch is turned to 'RUN' or the command is given by the running control, an output is generated every 2 seconds in the form

SZ 00,00 = <value>

or

IZ 00,00 = <value>

The continuous reading of a time is interrupted by actuation of any key.

10.12 Modules

SIP module:

Input of a word variable (see module description).

EMAS module:

Input of ASCII characters for evaluation by the module (see module description).
10.13 Inputs and outputs of word variables

The following variables may be called over the serial interface:

EW, EA, MW, MA, AW, AA, and M

<table>
<thead>
<tr>
<th>Input by keyboard</th>
<th>Display on the screen</th>
</tr>
</thead>
<tbody>
<tr>
<td>$EAO0, 00&lt;CR&gt;</td>
<td>2,456</td>
</tr>
<tr>
<td>$AA00, 00&lt;CR&gt;</td>
<td>1,010</td>
</tr>
<tr>
<td>$MA01, 00&lt;CR&gt;</td>
<td>2,731</td>
</tr>
<tr>
<td>$AW00, 00&lt;CR&gt;</td>
<td>12010</td>
</tr>
<tr>
<td>$AW00, 00&lt;CR&gt;</td>
<td>21010</td>
</tr>
</tbody>
</table>

Change of a bit flag content

<table>
<thead>
<tr>
<th>Input by keyboard</th>
<th>Display on the screen</th>
</tr>
</thead>
<tbody>
<tr>
<td>$EAO1, 00&lt;CR&gt;</td>
<td>0 or 1</td>
</tr>
</tbody>
</table>

10.14 Inputs and outputs of bit variables

The following variables may be called over the serial interface: M

<table>
<thead>
<tr>
<th>Input by keyboard</th>
<th>Display on the screen</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M'01, 00&lt;CR&gt;</td>
<td>1</td>
</tr>
</tbody>
</table>

10.15 Error polling

Error data, which can be polled by $F<CR>, are filed in the memory of the central processor 07 ZE 86 at three different check-up times:

1. Check-up of the hardware after each power-on:
   - check-up of the PROCONTIC b bus
   - check-up of the bit processor
   - check-up of the CMOS-RAM's
   - check-up of the operating system check sum

2. Check-up of the program at transition from STOP to RUN:
   - parity check of the user program
   - setpoint errors for time values

3. Check-up of the running program (a list of the error messages is included in chapter 2 section 4.5).

The error messages are filed in a stack register. When polling by $F<CR>$ the last error is displayed. Another polling shows the next earlier error until the display "KEIN FEHLER IM STACK" (No error in the stack) appears. This information indicates that all errors filed were issued.

10.16 Software version number

The display of the software version number is issued by the following command:

$V<CR>

10.17 Monitoring function

The monitoring function allows the input and output of variables (see table 10.1), baud rate setting, polling of the maximum cycle time etc.

Monitoring is called by the input of $. A complete command is terminated by carriage return (CDH). Each character entered, including the termination character, is echoed by the word processor 07 WP 84 resp. the central processor 07 ZE 86, unless the echo function was turned off by the module SIN. A faulty input is acknowledged with < ?, CR, LF >. The correction of an input is done by the same procedure. The user must interrupt his faulty input by a non-valid character. There is no delete function.

Notice:
Processing of the monitoring command is done on the level of the background modules, i.e. several background modules are in the queue, the monitoring command is executed in between two modules.

10.18 XON/XOFF mode

The software handshake with XON/XOFF is provided in the following kind:

After receiving a command with the termination <CR> the monitor transmits a XOFF. The processing of the command is done. After the command is processed the monitor transmits XON.

Attention!
If the opposite station does not respond with XON on time, the text buffers of the digital timer, of the word processor and of the central processor may overflow. Do secure by program!
Priority of the interface programs

- Monitor
- SIP module
- EMAS module
- logging
The total PROCNTIC b processing cycle consists of the processing time for bit and word commands and of the run time of the individual variants. Bit commands are mostly single or double words. On the other hand, word commands are mostly double or triple words. Refer to the tables below for the processing time of the individual commands.

### 11.1 Run time for bit processing

<table>
<thead>
<tr>
<th>Commands</th>
<th>No. of words</th>
<th>Processing time (in us) for 07 ZE 82</th>
<th>07 ZE 84</th>
<th>07 ZE 86</th>
<th>07 ZE 88</th>
<th>07 TI 82</th>
</tr>
</thead>
<tbody>
<tr>
<td>E &lt;= 15,15</td>
<td>1 (a) 2.44</td>
<td>2.44</td>
<td>4.88</td>
<td>4.88</td>
<td>4.88</td>
<td>4.88</td>
</tr>
<tr>
<td>A &lt;= 15,15</td>
<td>1 (a) 2.44</td>
<td>2.44</td>
<td>4.88</td>
<td>4.88</td>
<td>4.88</td>
<td>4.88</td>
</tr>
<tr>
<td>E &gt; 15,15</td>
<td>2 (a) 4.88</td>
<td>4.88</td>
<td>150</td>
<td>9.76</td>
<td>9.76</td>
<td>9.76</td>
</tr>
<tr>
<td>A &gt; 15,15</td>
<td>2 (a) 4.88</td>
<td>4.88</td>
<td>150</td>
<td>9.76</td>
<td>9.76</td>
<td>9.76</td>
</tr>
<tr>
<td>M ......</td>
<td>1 (a) 2.44</td>
<td>2.44</td>
<td>4.88</td>
<td>4.88</td>
<td>4.88</td>
<td>4.88</td>
</tr>
<tr>
<td>M'......</td>
<td>2 (a) 4.88</td>
<td>4.88</td>
<td>150</td>
<td>9.76</td>
<td>9.76</td>
<td>9.76</td>
</tr>
<tr>
<td>S ......</td>
<td>1 (a) 2.44</td>
<td>2.44</td>
<td>4.88/150 (b)</td>
<td>4.88/150 (b)</td>
<td>4.88/150 (b)</td>
<td>4.88/150 (b)</td>
</tr>
<tr>
<td>S'......</td>
<td>2 (a) 4.88</td>
<td>4.88</td>
<td>150</td>
<td>9.76</td>
<td>9.76</td>
<td>9.76</td>
</tr>
<tr>
<td>T ......</td>
<td>1.2.44</td>
<td>2.44</td>
<td>120</td>
<td>120</td>
<td>4.88</td>
<td>4.88</td>
</tr>
<tr>
<td>=S T ......</td>
<td>1.2.44</td>
<td>2.44</td>
<td>4.88</td>
<td>4.88</td>
<td>4.88</td>
<td>4.88</td>
</tr>
<tr>
<td>with #0,abcEx</td>
<td>2</td>
<td>4.88 (c)</td>
<td>4.88 (c)</td>
<td>9.76/240 (b)</td>
<td>9.76/240 (b)</td>
<td>200 (d)</td>
</tr>
<tr>
<td>with cE......</td>
<td>2</td>
<td>4.88 (c)</td>
<td>4.88 (c)</td>
<td>9.76/500 (b)</td>
<td>9.76/500 (b)</td>
<td>400 (d)</td>
</tr>
<tr>
<td>with cM'......</td>
<td>2</td>
<td>4.88 (c)</td>
<td>4.88 (c)</td>
<td>9.76/500 (b)</td>
<td>9.76/500 (b)</td>
<td>400 (d)</td>
</tr>
<tr>
<td>Z ......</td>
<td>1.2.44</td>
<td>2.44</td>
<td>110/140 (b)</td>
<td>110/140 (b)</td>
<td>9.76/270 (b)</td>
<td>250 (d)</td>
</tr>
<tr>
<td>=S Z ......</td>
<td>1.2.44</td>
<td>2.44</td>
<td>4.88 (c)</td>
<td>4.88 (c)</td>
<td>9.76/270 (b)</td>
<td>250 (d)</td>
</tr>
<tr>
<td>with #W......</td>
<td>1</td>
<td>4.88 (c)</td>
<td>4.88 (c)</td>
<td>9.76/270 (b)</td>
<td>9.76/270 (b)</td>
<td>250 (d)</td>
</tr>
<tr>
<td>with #O......</td>
<td>1</td>
<td>4.88 (c)</td>
<td>4.88 (c)</td>
<td>19.52/320 b</td>
<td>19.52/320 b</td>
<td>300 (d)</td>
</tr>
<tr>
<td>= MA</td>
<td>1.2.44</td>
<td>2.44</td>
<td>4.88</td>
<td>4.88</td>
<td>4.88</td>
<td>4.88</td>
</tr>
<tr>
<td>=N MA</td>
<td>1.2.44</td>
<td>2.44</td>
<td>4.88</td>
<td>4.88</td>
<td>4.88</td>
<td>4.88</td>
</tr>
<tr>
<td>= MC</td>
<td>1.2.44</td>
<td>2.44</td>
<td>4.88</td>
<td>4.88</td>
<td>4.88</td>
<td>4.88</td>
</tr>
<tr>
<td>=N MC</td>
<td>1.2.44</td>
<td>2.44</td>
<td>4.88</td>
<td>4.88</td>
<td>4.88</td>
<td>4.88</td>
</tr>
<tr>
<td>! ME</td>
<td>1.2.44</td>
<td>2.44</td>
<td>4.88</td>
<td>4.88</td>
<td>4.88</td>
<td>4.88</td>
</tr>
<tr>
<td>! PE</td>
<td>1.2.44</td>
<td>4.88</td>
<td>190</td>
<td>190</td>
<td>190</td>
<td>190</td>
</tr>
<tr>
<td>= PE</td>
<td>1.2.44</td>
<td>2.44</td>
<td>190</td>
<td>190</td>
<td>190</td>
<td>190</td>
</tr>
<tr>
<td>=N PE</td>
<td>1.2.44</td>
<td>2.44</td>
<td>190</td>
<td>190</td>
<td>190</td>
<td>190</td>
</tr>
<tr>
<td>! SP ......</td>
<td>1</td>
<td>4.88</td>
<td>140</td>
<td>140</td>
<td>140</td>
<td>140</td>
</tr>
<tr>
<td>= SP ......</td>
<td>1</td>
<td>2.44</td>
<td>140</td>
<td>140</td>
<td>140</td>
<td>140</td>
</tr>
<tr>
<td>! SPM ......</td>
<td>1</td>
<td>2.44</td>
<td>140</td>
<td>140</td>
<td>140</td>
<td>140</td>
</tr>
<tr>
<td>= SPM ......</td>
<td>1</td>
<td>2.44</td>
<td>140</td>
<td>140</td>
<td>140</td>
<td>140</td>
</tr>
<tr>
<td>! MR ......</td>
<td>1</td>
<td>2.44</td>
<td>140</td>
<td>140</td>
<td>140</td>
<td>140</td>
</tr>
<tr>
<td>= BA...A... (bit module)</td>
<td>2</td>
<td>4.88/9.76 (b)</td>
<td>140</td>
<td>140</td>
<td>4.88/140 (b)</td>
<td>4.88/140 (b)</td>
</tr>
<tr>
<td>! BA...A... (bit module)</td>
<td>2</td>
<td>2.44</td>
<td>19.52</td>
<td>19.52</td>
<td>460</td>
<td>460</td>
</tr>
<tr>
<td>B...</td>
<td>1</td>
<td>2.44</td>
<td>2.44</td>
<td>4.88</td>
<td>4.88</td>
<td>4.88</td>
</tr>
</tbody>
</table>

(a) with attribute + 1 word
(b) 1. time for polling or not fulfilled allocation
and 2. time for fulfilled allocation
(c) for not fulfilled allocation
(d) for fulfilled allocation
11.2 Run time for word processing

11.2.1 Word operators

<table>
<thead>
<tr>
<th>Commands</th>
<th>Number of words</th>
<th>Processing time in (μs)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>linkages</td>
<td></td>
</tr>
<tr>
<td></td>
<td>! &amp; / + -</td>
<td>*</td>
</tr>
<tr>
<td>AW..... (c)</td>
<td>2 (a)</td>
<td>170</td>
</tr>
<tr>
<td>MW..... (a)</td>
<td>170</td>
<td>270</td>
</tr>
<tr>
<td>EA..... (a)</td>
<td>170</td>
<td>350</td>
</tr>
<tr>
<td>AA..... (a)</td>
<td>170</td>
<td>350</td>
</tr>
<tr>
<td>MA..... (a)</td>
<td>170</td>
<td>350</td>
</tr>
<tr>
<td>KW..... 2</td>
<td>170</td>
<td>270</td>
</tr>
<tr>
<td>KA..... 2</td>
<td>170</td>
<td>350</td>
</tr>
<tr>
<td>a          1</td>
<td>140 (60)</td>
<td>170</td>
</tr>
<tr>
<td>w          1</td>
<td>170 (150)</td>
<td>140 (70)</td>
</tr>
<tr>
<td>b          1</td>
<td>230 (350)</td>
<td></td>
</tr>
<tr>
<td>with M' or 1</td>
<td>230 (350)</td>
<td></td>
</tr>
<tr>
<td>E/A &gt;15,15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c          1</td>
<td>460 (640)</td>
<td>700 (1200)</td>
</tr>
<tr>
<td>with M' or 1</td>
<td>460 (640)</td>
<td>980 (1200)</td>
</tr>
<tr>
<td>E/A &gt;15,15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BA...A... 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BS...F... (b) 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BS...P... (b) 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MW..... 3 (a)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HD...... 5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RW..... 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RA..... 2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The numbers in brackets are for the central processor 07 ZE 86
(a. with attribute + 1 word
(b. logging and text modules are also possible

11.2.2 Function and logging modules

The run time for the individual function module is shown in the description (see section 6.10.)

<table>
<thead>
<tr>
<th>Kind of call</th>
<th>Processing time (in μs) for</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>07 WP 84</td>
</tr>
<tr>
<td>-call and end of module P...</td>
<td>350</td>
</tr>
<tr>
<td>-call and end of module F000</td>
<td>250</td>
</tr>
<tr>
<td>-per each ASCII characters</td>
<td>35</td>
</tr>
<tr>
<td>-parameter</td>
<td>100</td>
</tr>
<tr>
<td>-attribute c/b in addition</td>
<td>200</td>
</tr>
<tr>
<td>-if module is not called (per each word)</td>
<td>2,44</td>
</tr>
</tbody>
</table>

(1 = reading-in of the ASCII characters, parameters and attributes is done at the end of the cycle.)
11.2.3 Times for the direct word processing by the word processor 07 WP 84

The direct word processing is located in the main program or in a module definition processed with a BA...

The bit central processor is in standby mode during direct word processing. The standby-time of the bit central processor with the word processor 07 WP 84 resp. the switch-over time from bit to word processing in the case of the central processor 07 ZE 86 for the direct word processing is composed of the following partial times:

- switch-over time from bit to word processing (75 µs)
- run time (see sequence 11.2.3)
- switch-over time from word to bit processing (75 µs)

Example for 07 WP 84:

! E 00,00 = A 01,00

\[ \text{time 1:} \quad 75 \, \mu s \]

! MO1,00 + cm 00,00 = MO1,02
\[ 170 \, \mu s \ (460 \, \mu s + 170 \, \mu s) \quad 200 \, \mu s = 1000 \, \mu s \]

\[ \text{time 3:} \quad 75 \, \mu s \]

The standby time of the bit central processor is:
\[ 1150 \, \mu s \]

! E 01,00 = A 02,00

Example for 07 ZE 86:

! E 00,00 = A 01,00

\[ \text{time 1:} \quad 75 \, \mu s \]

! MO1,00 + cm 00,00 = MO1,02
\[ 170 \, \mu s \ (640 \, \mu s + 170 \, \mu s) \quad 180 \, \mu s = 1160 \, \mu s \]

\[ \text{time 3:} \quad 75 \, \mu s \]

The switchover time of the central processor is:
\[ 1310 \, \mu s \]

! E 01,00 = A 02,00

11.2.4 Modules BA

Module BA are processed directly (bit central processor in standby). Parameters can be transferred.

The time for processing the module comprises the following partial times:

- call and end of module 
- 250 µs
- 50 µs
- run time (see section 11.2.3).
- replacement of formal parameters by actual parameters (per each formal parameter)
  for MW.. and MA.,
  for EA.. and AA.,
- 50 µs
- 130 µs

Example:

! E 00,00 = BA23,A004
MW01,02
MW01,05
MW03,07
NOPO

! PBA004
! RW00 + MW01,00 + RW01
(170+50) + 170 + (170+50)
= RW02
+ (200+50) =

! RW01 * MW01,02 = MW01,03
(170+50) + 310 + 200 = 730 µs

The standby time of the bit central processor is:
\[ 1990 \, \mu s \]

11.2.5 Modules BS...A...

The module is essentially processed in parallel to the bit program. Thus accessing the bus is merely necessary at module start and for data transfer (in the case of I/O variables).

It is necessary to distinguish between the processing time for the module and the standby time of the bit central processor.
<table>
<thead>
<tr>
<th>Function</th>
<th>Run time (in µs)</th>
<th>Bit processing standby (in µs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>call and end of the module per each parameter transferred</td>
<td>450</td>
<td>450</td>
</tr>
<tr>
<td>run time according to table in section 11.2.3</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td>polling EW,..., EA,..., polling E,..., A,..., and M',,..., with attribute b or c</td>
<td>-</td>
<td>130</td>
</tr>
<tr>
<td>output AA,..., AW,..., output A,..., resp. M',,..., with attribute b or c</td>
<td>-</td>
<td>130</td>
</tr>
<tr>
<td>input of the values to the actual parameters (done once only) per parameter EA,...,</td>
<td>60</td>
<td>60</td>
</tr>
<tr>
<td>MW,..., MA,..., AW,..., and AA,..., per each bit parameter with attribute b</td>
<td>20</td>
<td>-</td>
</tr>
<tr>
<td>with attribute c</td>
<td>160</td>
<td>160</td>
</tr>
<tr>
<td>output of the output parameter (done once only)</td>
<td>390</td>
<td>390</td>
</tr>
<tr>
<td>AW,..., and AA,..., MW,..., and MA,..., bit parameter with attribute b</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>with attribute c</td>
<td>20</td>
<td>-</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>writing a word flag ($\text{MH}..., = \text{CR})</td>
<td>800 µs</td>
<td></td>
</tr>
<tr>
<td>writing an analog flag ($\text{MA}..., = \text{CR})</td>
<td>450 µs</td>
<td></td>
</tr>
<tr>
<td>writing a word flag ($\text{MH}..., = +12345)</td>
<td>750 µs</td>
<td></td>
</tr>
<tr>
<td>writing an analog flag ($\text{MA}..., = +1,345)</td>
<td>450 µs</td>
<td></td>
</tr>
<tr>
<td>receiving an integer value</td>
<td>700 µs</td>
<td></td>
</tr>
<tr>
<td>receiving an analog value</td>
<td>550 µs</td>
<td></td>
</tr>
</tbody>
</table>

11.3 Serial interface
The times shown are related to the processing of one command, i.e. to the program run time after receiving the termination character up to the transmission of the first response character.

1. Monitor

2. Serial input SIP

8.2