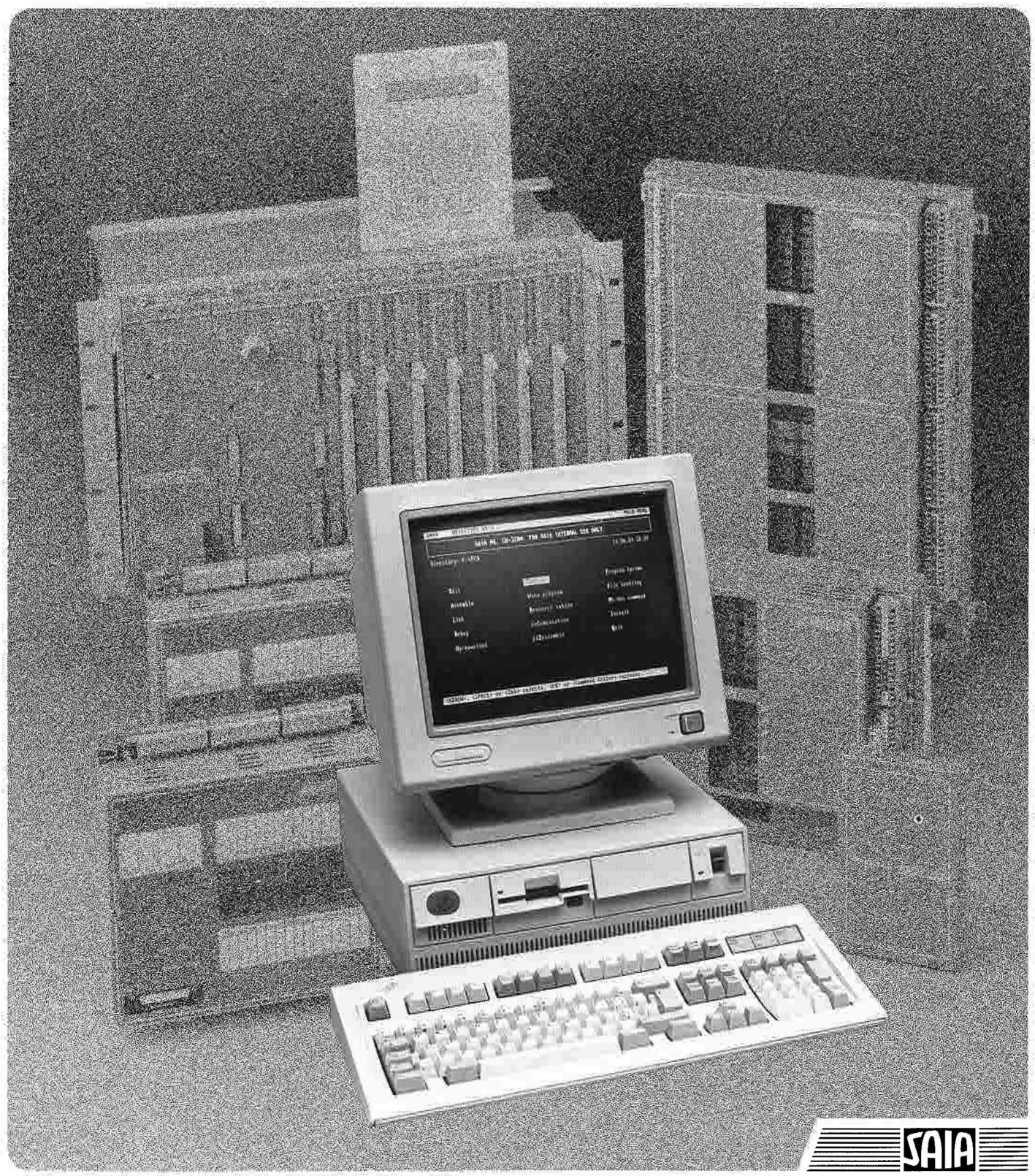
SAIA®PLC
Programmable controllers

Manual of the software level 2+3



English edition 26/722 E1

PART H AUXILIARY INSTRUCTIONS, LEVEL 2

H 1 One-line instructions NOP 1111

H 2 2-line PAS-instructions
PAS 16/17
PAS 19
PAS 23
PAS 24
PAS 50
PAS 54/56
PAS 55/57
PAS 58

H 3 10-line PAS-instructions
PAS 190 (and PAS 19)
PAS 200...212
PAS 250/251

PART H AUXILIARY INSTRUCTIONS, LEVEL 2

H 1 One-line instructions

NOP 1111 Direct interface operation

The instruction NOP 1111 may be used in a program as often as desired. It causes immediate operation of the serial interface on the receiving side.

NOP 1111 is particularly used where no PP-change and thus no interface operation is performed due to bit processor instructions in succession (e.g. some SCR-instruction in succession).

```
SCR   26Ø
 31   262
SCR   26Ø
 28   281
NOP   1111 ; NOP 1111 causes operation of the interface
SCR   26Ø
 3Ø     4
  .
  .
  .
```

NOP 1111 is independent of the ACCU state and does not influence it either.

H 2 2-line PAS-instructions

PAS 16, PAS 17 Exclusive operation without time base

To be able to supply accurate timing pulses (e.g. for step motor), or to execute a small part of a program rapidly, it is often desired to have activated only one parallel program.

PAS 16 blocks all other PPs, timers, the serial data interface and software date-time.

PAS 17 cancels the blocking instruction, i.e. the parallel programs, timers and date-time continue from where they were interrupted. The serial interface is active again.

Instruction format

PAS (29) 16 ØØ Ø	Blocking instruction 2nd line always remains Ø
PAS (29) 17 ØØ Ø	Deblocking instruction 2nd line always remains Ø

1. Example of an accurate timing pulse of 2ms*

The user program consists of several parallel programs. A circulating program consists of the following instructions:

```
        .
        .
STH      4ØØ
JIO
PAS      16
      ØØ  Ø
SCR      26Ø
      ØØ  16
SEO       4Ø
DEC      26Ø
STH      26Ø
JIO
REO       4Ø
PAS      17
      ØØ  Ø
STH       3
ANL       7
        :
```

Duration of interrupt during which all other parallel programs, timers, serial data interface and software date-time are blocked.

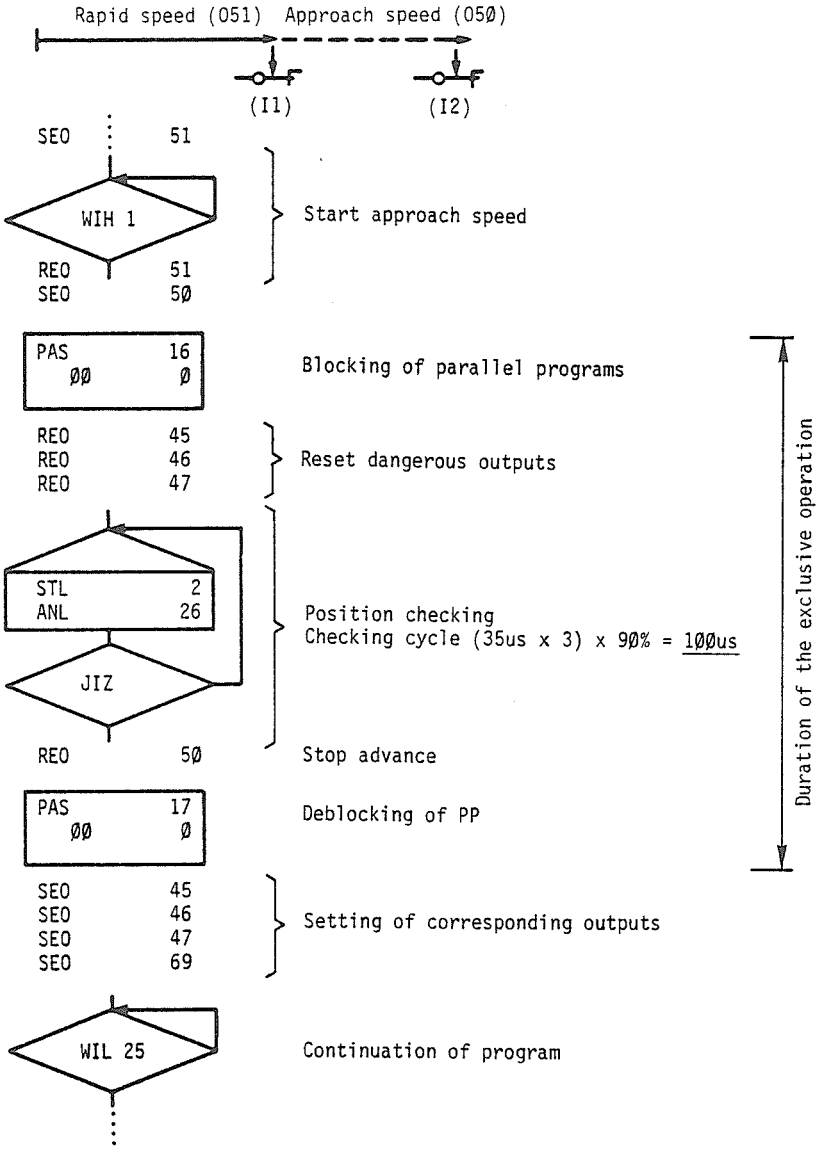
Calculation of the pulse length:
(35µs x 4 x 16) x 9Ø% = 2ms*

The reduction factor of 9Ø% results from the cessation of PP processing. It must be checked by means of measurements.

*) Times are valid for PCA232. They are twice as long for PCAØ2, PCA14 and PCA222.

2. Example of an extremely short reaction time

A carriage moving at creep speed must be positioned extremely accurately by means of a signal switch (I2).



In the above example the interrogation loop I2 consists of 3 instruction lines. Depending on the time of arrival of the stop signal (opening of I2) 4 to 6 program lines up to REO 5Ø must be executed. This corresponds to an internal reaction time of approximately 125...19Øµs.*

The total reaction time is calculated by adding the delays at the input and output interfaces to the above mentioned values. If smoothed DC-current is supplied, these delays are constant and can directly be introduced in the process.

---

*) Times are valid for PCA232. They are twice as long for PCAØ2, PCA14 and PCA222.

Remarks

- The duration of blocking is independent of the user program and can last several seconds. In the above-mentioned example, all monitoring functions are interrupted during this time. If this is to be avoided, these functions can be integrated into the interrogation loop.

- If blocking lasts longer than 90ms, the software clock and the internal timers will be slow by the amount of the corresponding time exceeded (with selected time base 1/10s).

- During blocking no transmission of information via the serial data interface is allowed, since transmission errors may occur.

PAS 19 End of interrupt management
(see PAS 19Ø)

PAS 23 Text output in the bit processor
(see chapter K 5 - text input and output)

PAS 24 Immediate telegram output

PAS 24 allows immediate output of a character or telegram via the serial interface irrespective of the time base. The response times can thus be reduced with PAS 24 via the serial interface.

Description

The user program informs the system when a character or telegram must be output by setting the element TBY "high". The system program, however, analyses the logic state of TBY only at certain, defined points of time (provided that the serial interface is not yet transmitting). These points of time are:

- clock of the time base, i.e. every 1ØØms or 1Øms respectively
- when the characters ACK, ENQ and the last character of a telegram are output or received (ETX or BCC)
- after terminating an interrupt service routine with the instruction PAS 19
- after outputting a text with the instruction PAS 23 or TXT (PCA232)
- with the instruction PAS 24

Application

```
SEO    5Ø9
PAS     24
ØØ       Ø
```

Transmit Busy (TBY) is set high

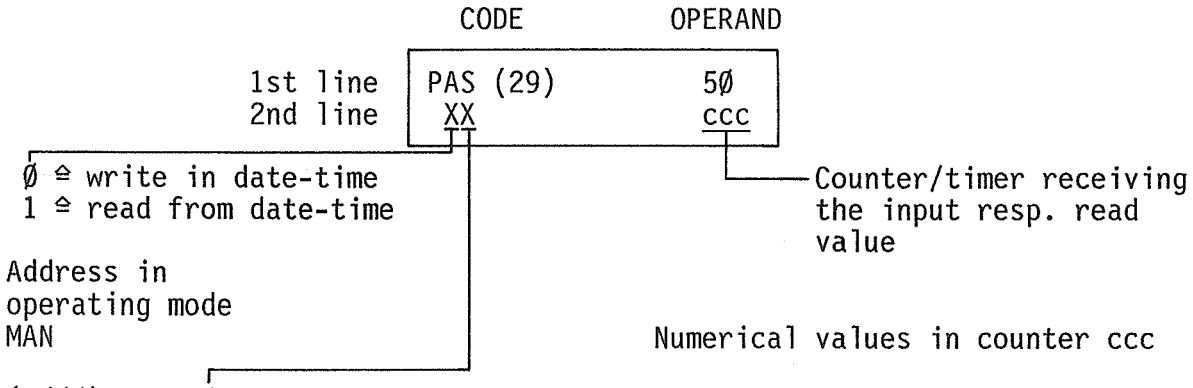
Immediate telegram output

PAS 24 is always executed independently of the ACCU and does not influence the ACCU.

PAS 5Ø Data transfer between date-time and counter

Writing and reading of the date-time in the address range 4ØØØ...4ØØ7 by means of the programming unit PØ5 was described already under operating modes MAN BIT. This section deals with date-time access via the user software. It allows setting of the date-time via user software as well as using date-time for real time switching functions.

By means of a 2-line PAS-instruction the date-time can be read out or new values can be entered.



Address in operating mode MAN		Numerical values in counter ccc
(4ØØØ)	Ø ≙ week of the year	1...53
(4ØØ1)	1 ≙ day of the week	1...7
(4ØØ2)	2 ≙ year	Ø...99
(4ØØ3)	3 ≙ month	1...12
(4ØØ4)	4 ≙ day of the month	1...28, 29*, 3Ø, 31
(4ØØ5)	5 ≙ hours	Ø...23
(4ØØ6)	6 ≙ minutes	Ø...59
(4ØØ7)	7 ≙ seconds	Ø...59

Output of date and time is also possible in several ways via the serial interface (see K 5 "Meaning of the special characters").

Example:
The minutes and seconds are reset in the 3Øth minute of every hour via input 7 according to the rising edge of a synchronisation pulse.

```
  STH          7;      Synchronisation input
  DYN        4ØØ
  JIZ
  SCR        3ØØ      } Set reference counter to zero
  ØØ           Ø      }
  PAS          5Ø  **  Set seconds to zero
  Ø7         3ØØ
  SCR        3ØØ      } Set reference counter to 3Ø
  ØØ           3Ø     }
  PAS          5Ø  **  Set minutes to 3Ø
  Ø6         3ØØ
  ------------> :
```

Part L
Example 1

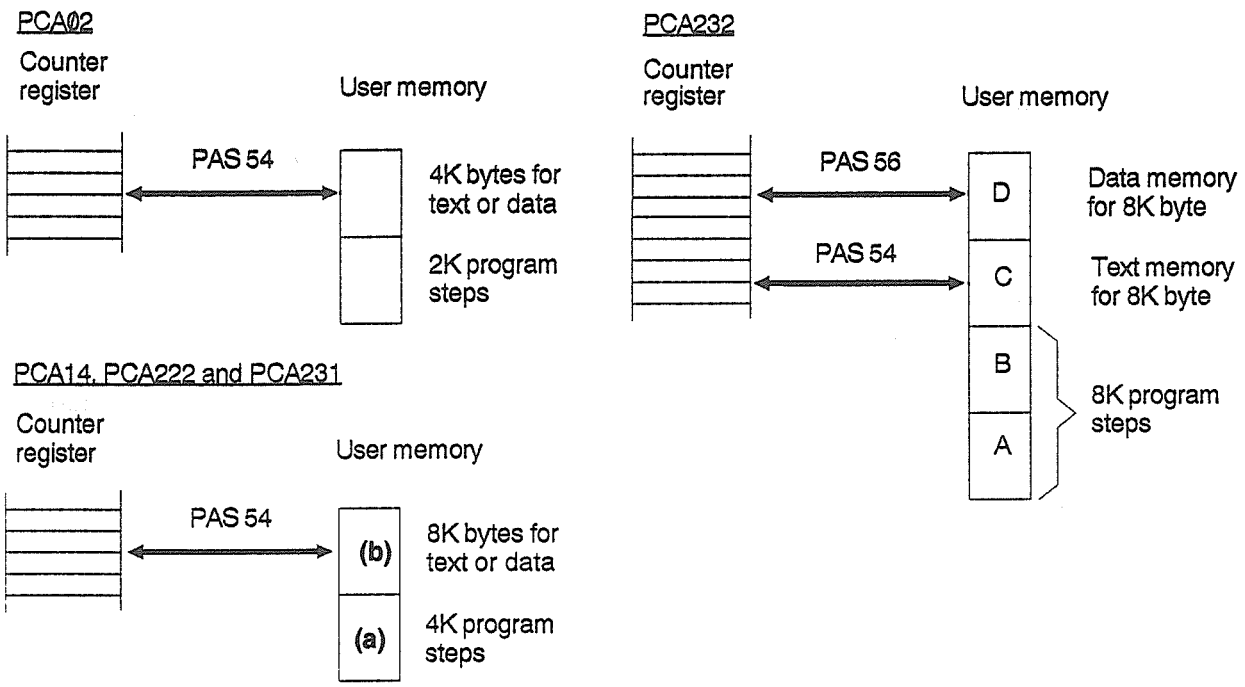
*) Only a hardware date-time takes leap years into account, the software date-time of the CPU does not.
**) The PAS 5Ø instruction is always executed independently of the ACCU-state and does not influence the ACCU.

PAS 54 Data exchange counter register <---> text memory

PAS 56 Data exchange counter register <---> data memory (only PCA232)

As evident from the following overview, both instructions can be used to store a great amount of data in the SAIA°PCA by using the text memory (or part of it). For PCA232, a special data memory is available which can be activated with PAS 56. In order to ensure upwards-compatibility with the other PAS 54 can also be used between counter register and text memory for PCA232.

Overview

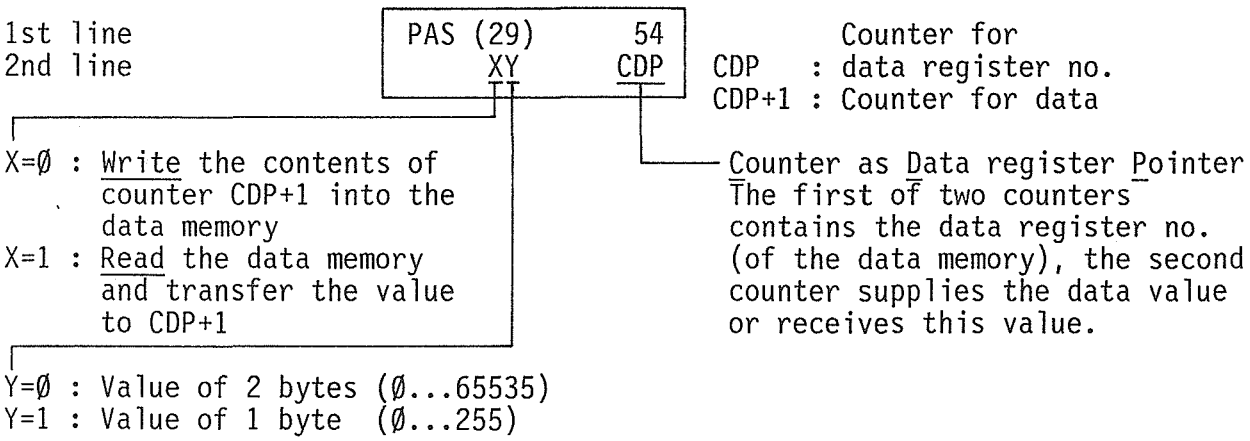


The following points must be noted:

- The memory area of the text memory or data register needs to be designed as a RAM, if data must be entered into it.
- "Manual" reading and overwriting of the data memory with the aid of the programming unit is described in chapter "Operating modes" text or MAN BCD.
- The term "data memory" will be used for both memories (text and data) in the following description.
- All following descriptions and examples refer to PAS 54. The instruction PAS 56 (for PCA232) can be used exactly in the same way.

PAS 54 Transfer between data memory and counter register

Instruction format:



PAS 54 is executed independently of the ACCU and does not change the ACCU.

The instruction PAS 54 causes the following actions:

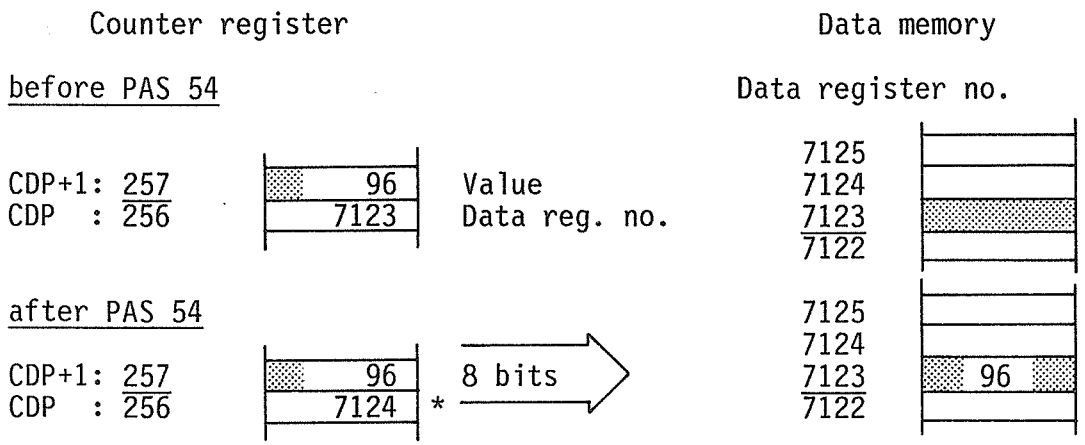
- Write (X=Ø): The value of CDP+1 (1 or 2 bytes) is written into the data memory with the data register number according to the contents of CDP.
- Read (X=1): Read the 1 or 2 data registers of the data memory with the data register number according to CDP and transfer to counter CDP+1.
- After executing one of these operations the contents of counter CDP is automatically increased by 1 or 2 (corresponding to 1 or 2 bytes). The pointer is automatically at the correct data register number for the following data transfer.

Examples

- Write a value of 1 byte from a counter into the data memory

The counter 257 contains the value 96 and must be transferred to data register 7123.

```
SCR (15)    256 }  Load counter CDP
    ØØ     7123 }  with data register number
PAS (29)     54
    Ø1      256    Write 1 byte into CDP+1 (C257) into the data memory
```

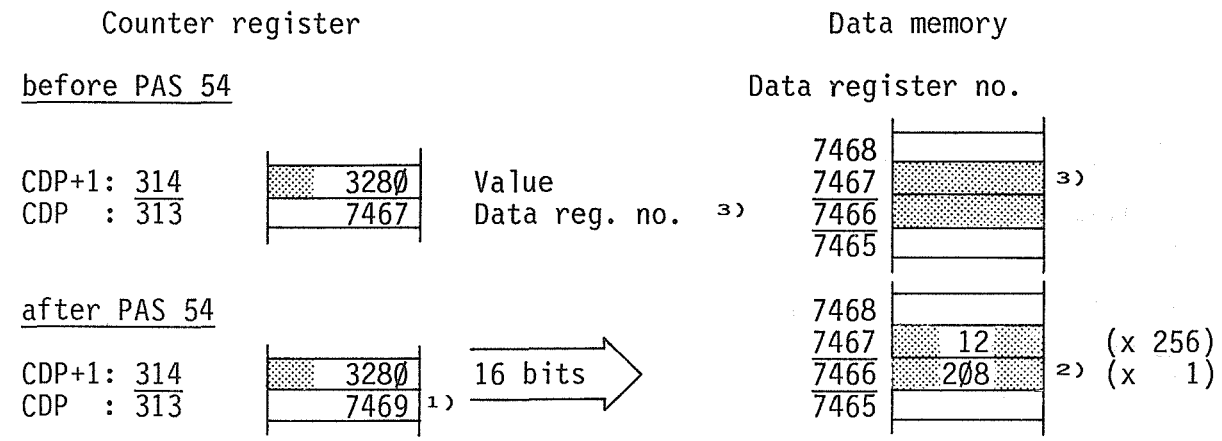


*) The value of CDP has been increased by 1.

- Write a value of 2 bytes from a counter into the data memory

Counter 314 contains value 328Ø and must be transferred to data registers 7466 and 7467.

```
SCR (15)    313 }  Load counter CDP
     ØØ    7467 }  with the higher data register number
PAS (29)     54
     ØØ     313    Write 2 bytes from CDP+1 (C314) into
                   the data memory
```



- Read a value of 2 bytes from the data memory and transfer it to a counter

The value 56477, stored in data register numbers 193Ø and 1931 must be transferred to counter 275.

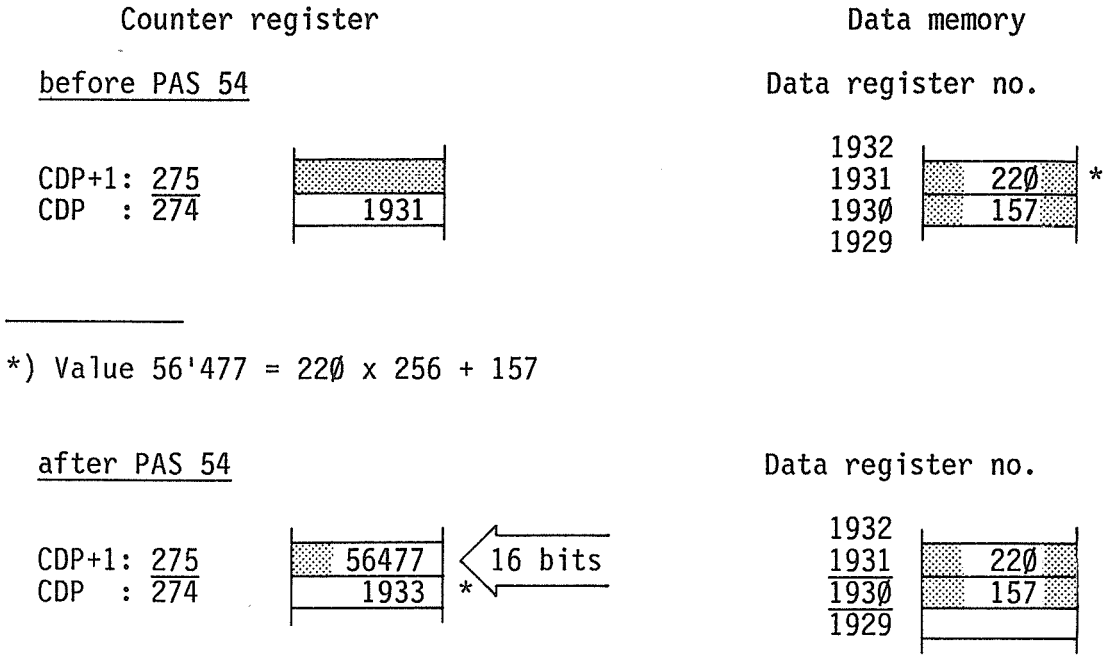
```
SCR (15)    274 }  Load counter CDP
     ØØ    1931 }  with the higher data register no.
PAS (29)     54
     1Ø     274    Write 2 bytes into CDP+1 (C275)
```

---

1) The value of CDP has been increased by 2.

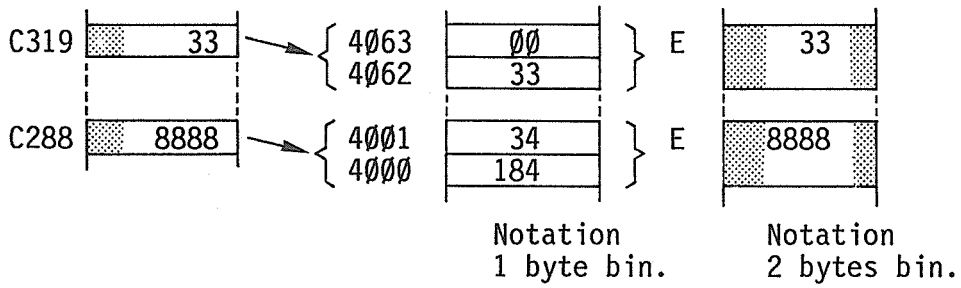
2) The value is stored in the data memory in binary notation (12 x 256 + 2Ø8 = 328Ø). The monitor of the CDP allows representing these values as a combined 2-byte-value with the aid of P1Ø/PØ5 (see "Operating modes").

3) It is recommended to choose odd data register numbers for values of 2 bytes.



- 32 counters from C288...C319 with values from 0...9999 must be transferred to the data memory starting with data register number 4001.

```
   SEI      0

   SCR    260  } Load counter CDP with the
    00   4001  } lowest data register number

→SCR    261  } Carry bit counter (CDP+1)
    31  1288  } for C288...C319

  ┌PAS     54
  │ 00    260┘  Write 2 bytes from CDP+1 (C261) into the data memory

   INI     31
  └JIO
```



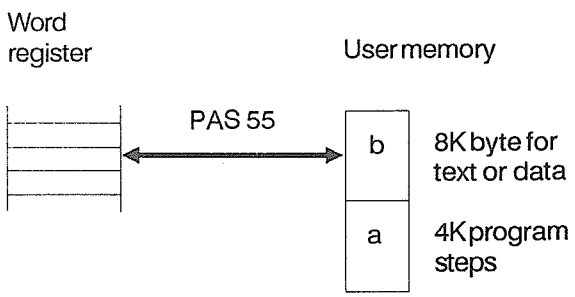
PAS 55 Data exchange word register <---> text memory (PCA231 and 232)

PAS 57 Data exchange word register <---> data memory (only PCA232)

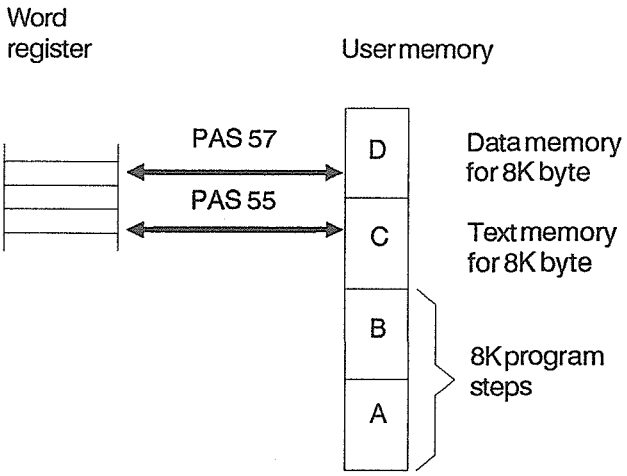
Like PAS 54/56, the instructions PAS 55/57 can be used for data transfer to the word registers.

Only the instruction PAS 57 will be described in the following. PAS 55 assumes the same function in the transfer to the text memory which is evident from the following two diagrams.

PCA231



PCA232



Instruction format:

1st line
2nd line

PAS (29) 57
X0 CDP

CDP : Counter for data register
CDP+1 : Counter for word register address

X=0 : Write the contents of the word register block (pointer CDP+1) into the data memory (pointer CDP)

X=1 : Read the contents of the data memory (pointer CDP) and transfer the value into the word register (pointer CDP+1)

Counter as Data register Pointer
The first of 2 counters (CDP) contains the register no. (of the data memory), the second counter (CDP+1) contains the address of the word register block.

The instruction PAS 57 triggers the following actions:

- Write (X=0): The contents of the word register block with address CDP+1 (5 word registers = 9 BCD-values + preceding sign on the whole) is written into the data memory as a 5 x 8 bit value starting with the data register according to CDP.

- Read (X=1): The 5 values starting with the number according to CDP are read in the data memory and transferred to the word register block acc. to CDP+1.

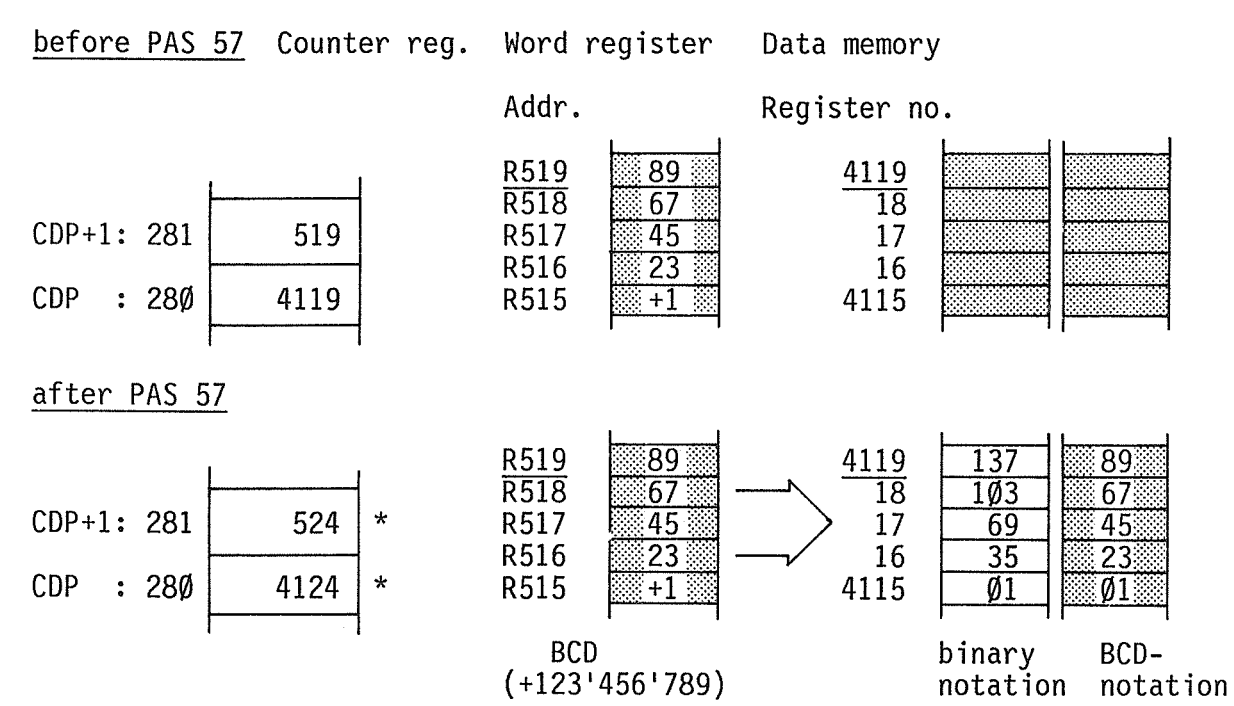
- The contents of the counter CDP and CDP+1 are then increased by 5 respectively. The pointers are automatically in the correct positions for the following data transfer.

Examples

- Write a word register block into the data memory

The register block R515...R519 must be written into the data memory at the data register numbers 4115...4119.

```
SCR (15)    28Ø  }  Load CDP with
     ØØ    4119  }  register number (MSD)

SCR (15)    281  }  Load CDP+1 with
           519   }  word register address (MSD)

PAS (29)     57     Write value of register
     ØØ     28Ø    block into the data memory
```



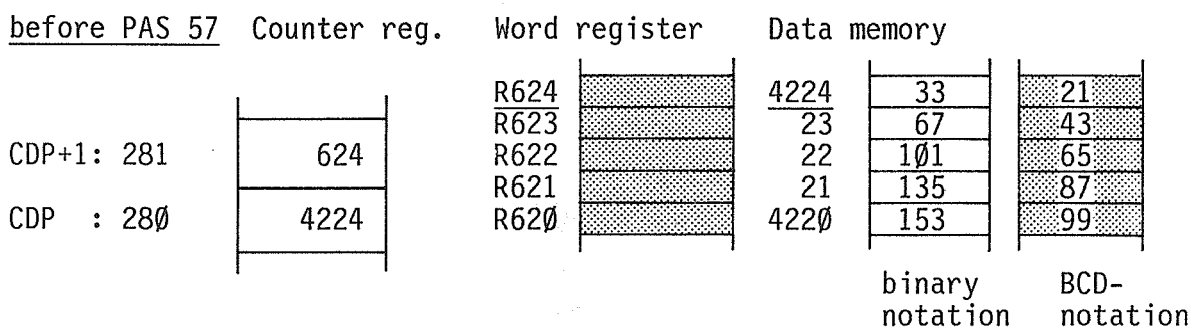
The preceding sign + is received as a Ø. The other values are stored 8 bit-wise in the correct sequence, if interrogated manually, however, they are displayed in binary notation. For manual conversion to BCD-notation refer to section "Operating modes". When they are transferred back into the word register later, these data are read in BCD-notation with the original values again.

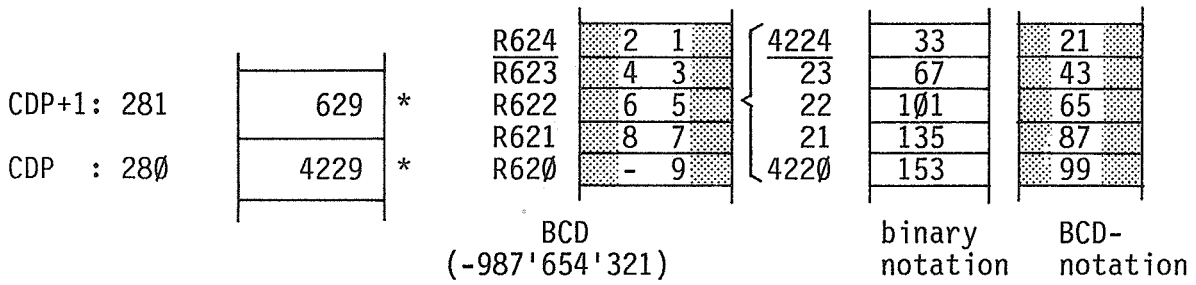
*) The counters for CDP and CDP+1 have been incremented by 5.
It is of advantage to choose addresses ending with 4 or 9 when transferring register blocks.

- Read 5 characters in the data memory and transfer into the word register

The data registers 422Ø...4224 are to be read and transferred to the word register block R62Ø...R624 **.

```
SCR (15)    28Ø  }   Load CDP with the
     ØØ    4224  }   highest data register no. (MSD)

SCR (15)    281  }   Load CDP+1 with the highest
     ØØ    624   }   word register address (MSD)

PAS (29)     57      Read value in the data memory
     1Ø     28Ø      and transfer to word register
```





*) The counters for CDP and CDP+1 have automatically been incremented by 5.

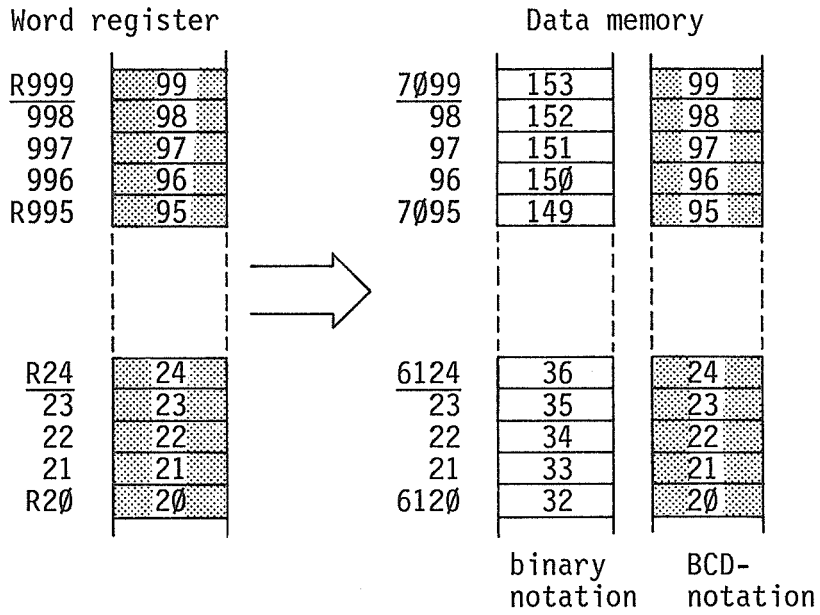
**) It is of advantage to choose addresses ending with 4 or 9 when processing register blocks.

- The entire word register R2Ø...R999 must be copied into the data memory starting with data register number 612Ø.

  R2Ø...R999 are 196 register blocks.

```
SCR     289  }  Load loop counter with
 ØØ     196  }  the number of register blocks

SCR     29Ø  }  Load CDP with the
 ØØ    6124  }  lowest data register no. + 4

SCR     291  }  Load CDP+1 with the
 ØØ      24  }  lowest word register address + 4

PAS      57
 ØØ     29Ø     Write to data memory

DEC     289
STH     289
JIO
```

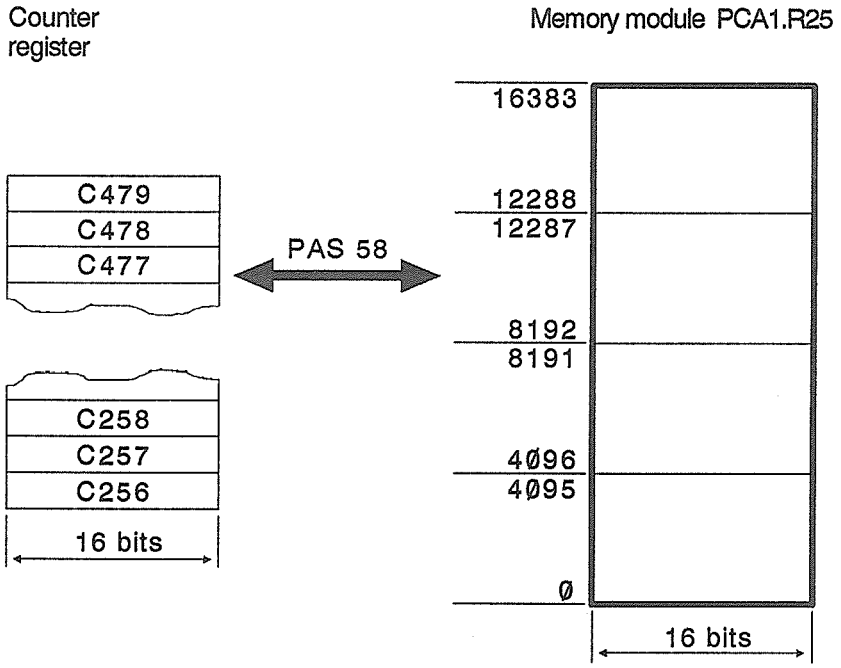
after PAS 57



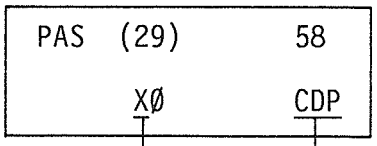
PAS 58 Data exchange between RAM memory module PCA1.R25 and counter register (only PCA14 from V6.Ø36 onwards)

The RAM memory module PCA1.R25 of the PCA14 is a large, buffered data memory of 16 K words (32 K bytes). The PAS 58 instruction allows data transfer between the data memory and the counter registers.

Overview



Instruction format:



CDP : Counter for memory address R25
CDP+1 : Counter for data

X=Ø : Write the contents of counter CDP+1 to memory module R25.

X=1 : Read the contents of memory module R25 and transfer the value to CDP+1.

Counter as Data register Pointer
The first of two counters (CDP) contains the memory address of module R25, the second counter provides or receives the data value.

PAS 58 is executed independently of the ACCU and does not modify it.

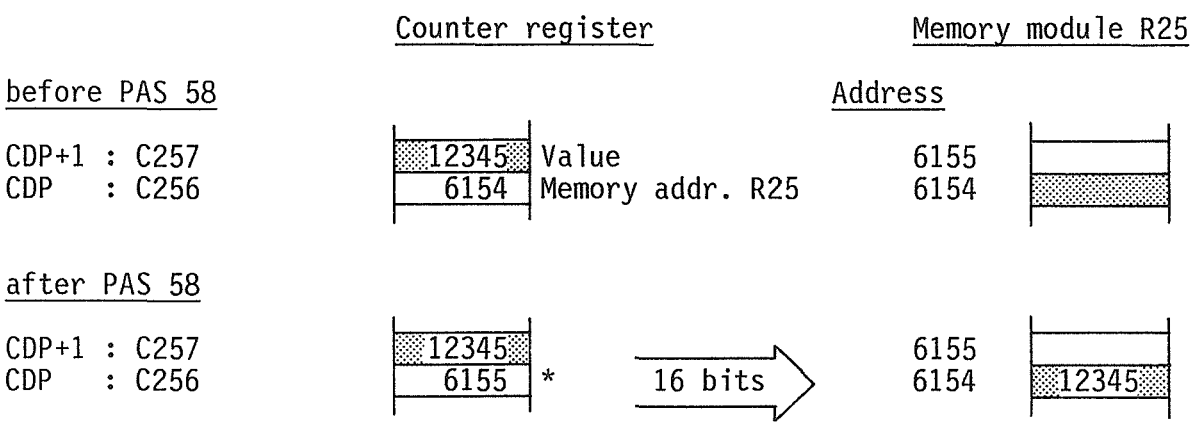
The instruction PAS 58 assumes the following functions:

- Write (X=Ø): The value of CDP+1 (1 word = 2 bytes) is written to memory module PCA1.R25 memory address in accordance with CDP.
- Read (X=1): Read the value (1 word = 2 bytes) in memory module PCA1.R25, memory address in accordance with contents of CDP, and transfer to counter CDP+1.
- On execution of one of these operations, the contents of counter CDP is automatically incremented by 1. The pointer is consequently at the correct memory address of R25 for the data transfer following next.

Examples

- Write the contents of C257 (e.g. value 12345) to the data memory of the module PCA1.R25, address 6154.

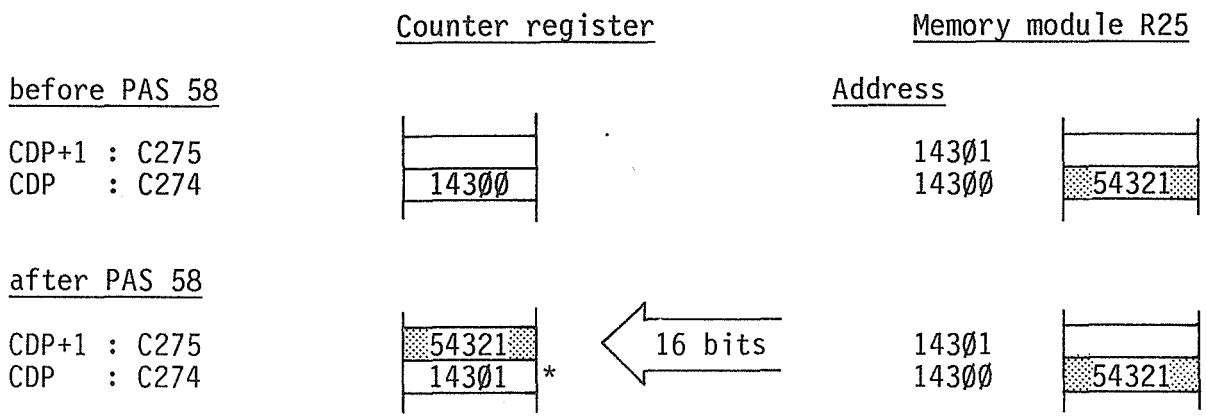
```
SCR (15)   256   ; Load counter CDP with memory address of R25
     ØØ   6154

PAS (29)    58
     ØØ    256   ; Write the value of CDP+1 (C257) to memory
                   module R25
```



*) The value of CDP has been incremented by 1.

- Read the value (e.g. 54321) of memory module R25, address 143ØØ, and transfer to counter C275.

```
SCR (15)  274    ;  Load counter CDP with memory address of R25
     Ø6  2Ø12        (6 x 2Ø48 + 2Ø12 = 143ØØ)

PAS (29)   58
     1Ø   274    ;  Read the value in R25 and transfer to CDP+1 (C275)
```



*) The value of CDP has been incremented by 1.

- The 16Ø counter contents from C32Ø to C479 must be transferred to memory module PCA1.R25 starting with address 4ØØ1.

```
        SEI     Ø
        SCR   26Ø  ;  Load counter CDP with memory address of R25
         ØØ  4ØØ1

┌─────► SCR   261  ;  Transmission counter (CDP+1) for C32Ø...C479
|        31  132Ø
|
|      ┌──────────┐
|      | PAS   58 |
|      | ØØ   26Ø |
|      └──────────┘
|
|       INI   159
└──────  JIO
```

- The 16Ø values of memory module R25, addresses 4ØØ1...416Ø, must be transferred to counters C32Ø...C479.

```
        SEI      Ø
        SCR    26Ø  ;  Load counter CDP with memory address of R25
         ØØ   4ØØ1

┌─────► ┌──────────┐
|       | PAS   58 |
|       | 1Ø    26Ø|
|       └──────────┘
|
|       SCR   132Ø  ;  Transfer value of transmission counter (CDP+1)
|        31    261     to C32Ø...C479
|
|       INI    159
└──────  JIO
```

H 3 1Ø-line PAS-instructions

PAS 19Ø and PAS 19 Interrupt management (IM)

Functional description

An interrupt service routine can be exclusively started via an element, normally an input. This interrupt service routine (ISR) is assigned instead of parallel program PP15 via PAS 19Ø.

The element INTRQ (Interrupt Request) can either be defined to be active "H" or active "L".

If the element INTRQ is active "H" and the element INTA (Interrupt Acknowledgement, normally an output) is "L", the interrupt service routine is started exclusively when the next change of a parallel program occurs. Execution of the remaining parallel programs is interrupted simultaneously.

In the interrupt service program normal program execution can be continued after the necessary measures (PAS 19) have been taken.

Assignment via PAS 19Ø is effected instead of the 15th parallel program via a 1Ø-line instruction (usually at the beginning of a program in the assignment section):

```
 1   PAS   (29)      19Ø     ;    Interrupt assignment
 2         ØØ        ssss  *  ;    Start of ISR (Interrupt-Service-Routine)
 3         ØØ          En    ;    Interrupt Request (input INTRQ)
 4         ØØ        ØØØx    ;    Level INTRQ ("H" = 1, "L" = Ø)
 5         ØØ          En    ;    Interrupt Acknowledge (output INTA)
 6         ØØ           Ø    )
 7          .           .    |
 8          .           .    }    NUL
 9          .           .    |
1Ø         ØØ           Ø    )
```

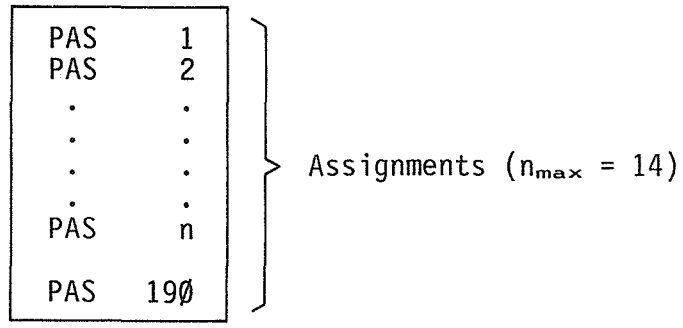
Exclusive operation is stopped at the end of the interrupt program via the two-line instruction PAS 19:

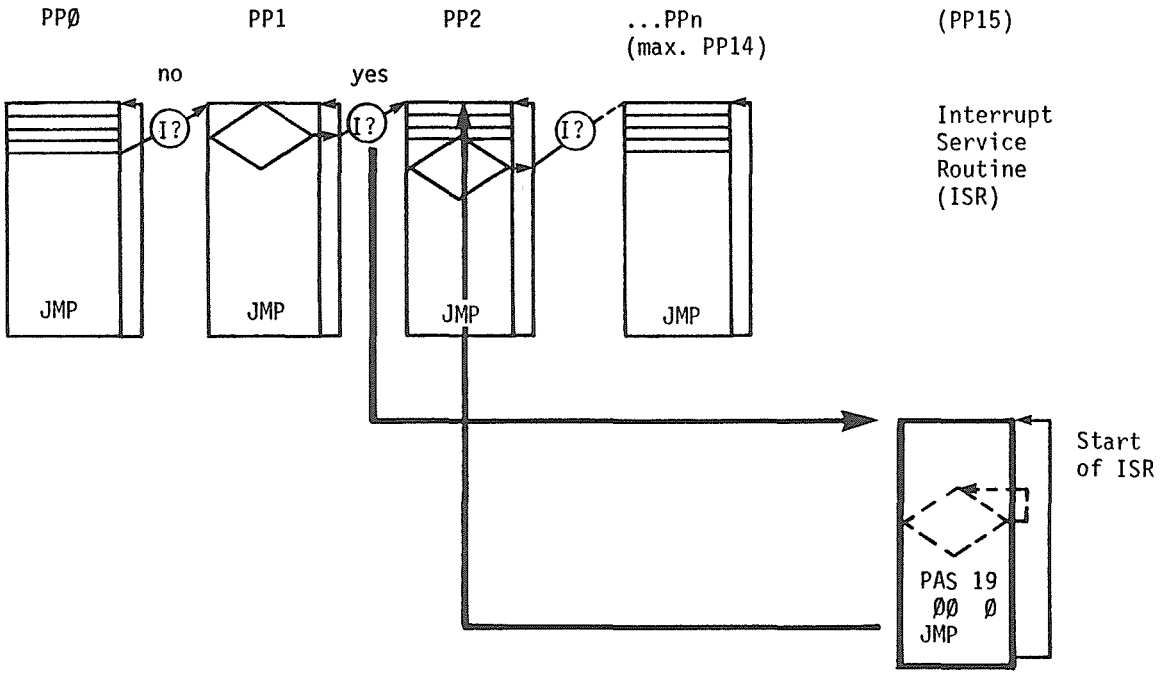
```
 1   PAS   (29)       19     ;    End of the exclusive run of the inter-
                                  rupt service routine (End ISR)
 2         ØØ          Ø     ;    NUL
```

*) When IAD/KAD (INSERT and KILL) are used by the "CI" program of the PCA-ASSEMBLER, this step address is not corrected automatically.

Details

a) Program organisation





The interrupt conditions are interrogated on each change from one PP to the next whereby the interrupt service routine is exclusively executed, if the result is positive.

If the conditions for returning to the normal program are given, this is initiated via PAS 19 in the ISR:

- The PPs are continued from where they were interrupted.

- A defined restart can be effected by reassigning the PPs.

b) Reaction time

Since the interrupt conditions are interrogated on each change from one PP to another, the reaction time between closing of the "interrupt contact" and start of the ISR-program virtually depends only on the input delay of the "interrupt input". The standard input delay of 8ms is increased by max. 1ms (PCA232) resp. 2ms (PCAØ2/14/222) resulting from the internal execution even in the case of an unfavourable program structure.

c) Timer execution and serial data interface

During exclusive execution of the ISR the time base is active, i.e. both timers and serial data interface can be used unrestrictedly. Consequently, the ISR may contain the output of texts or messages to a superior system in addition to time functions.

d) Interdependence between INTRQ, INTA and the interrupt service routine (ISR)

The following criteria must be fulfilled before starting the ISR:

- INTRQ (normally an input) "H" resp. "L", depending on the definition of the 4th line in the interrupt assignment.

- INTA (normally an output) "L".

INTA is automatically set to "H" by the system program. After approximately 1ms the exclusive run of the ISR is started. Execution of the ISR should take less time than two changes of INTRQ to "H" or to "L" respectively.

Regarding REO INTA, two cases having completely different effects are distinguished:

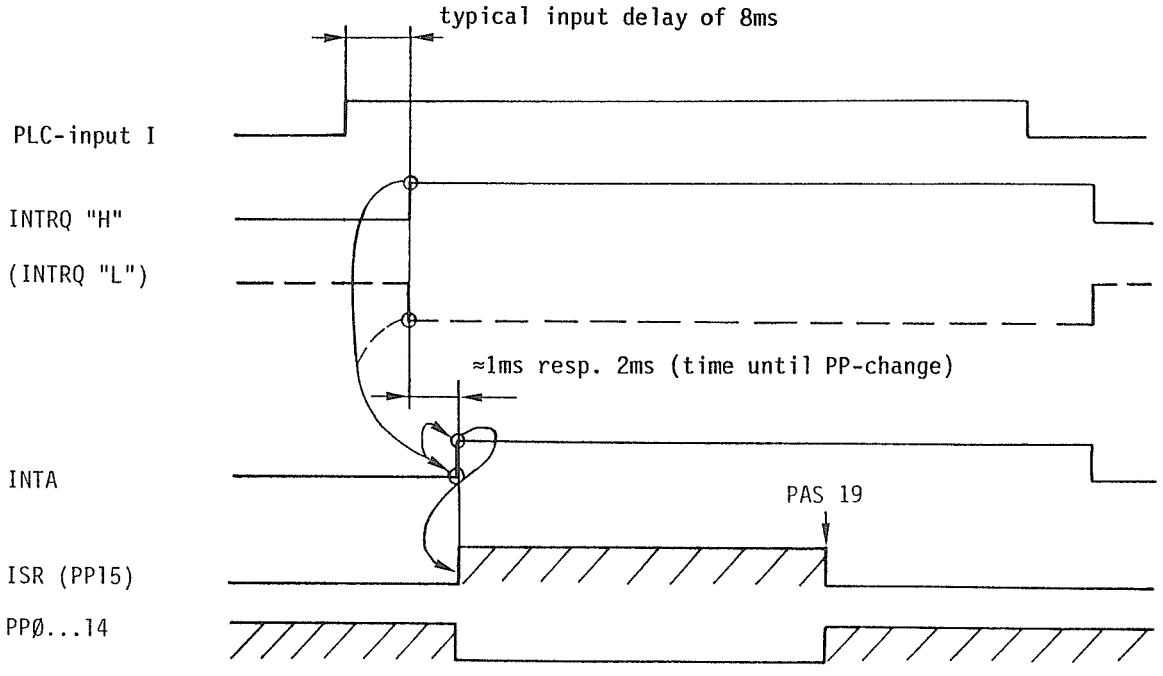
REO INTA is
- 1) outside the ISR
- 2) within the ISR

1) REO INTA outside the ISR (general case) in a circulating program

a) INTRQ active "H" : STL INTRQ
REO INTA

b) INTRQ active "L" : STH INTRQ
REO INTA

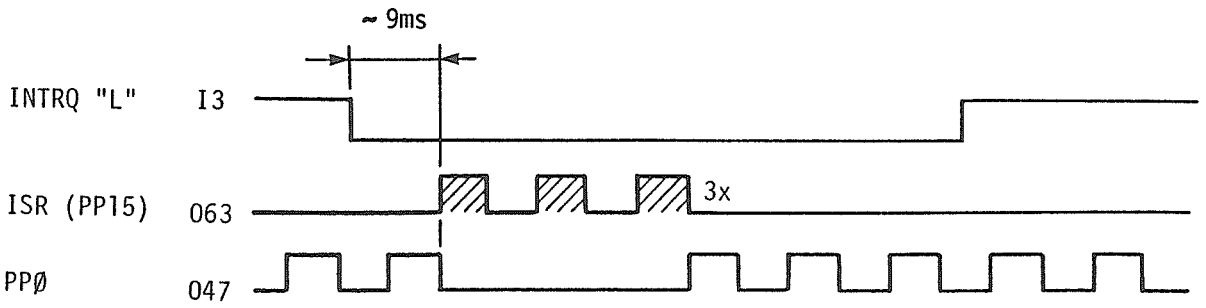
In this case the ISR is executed only once, when INTRQ changes to "H" or to "L" respectively.



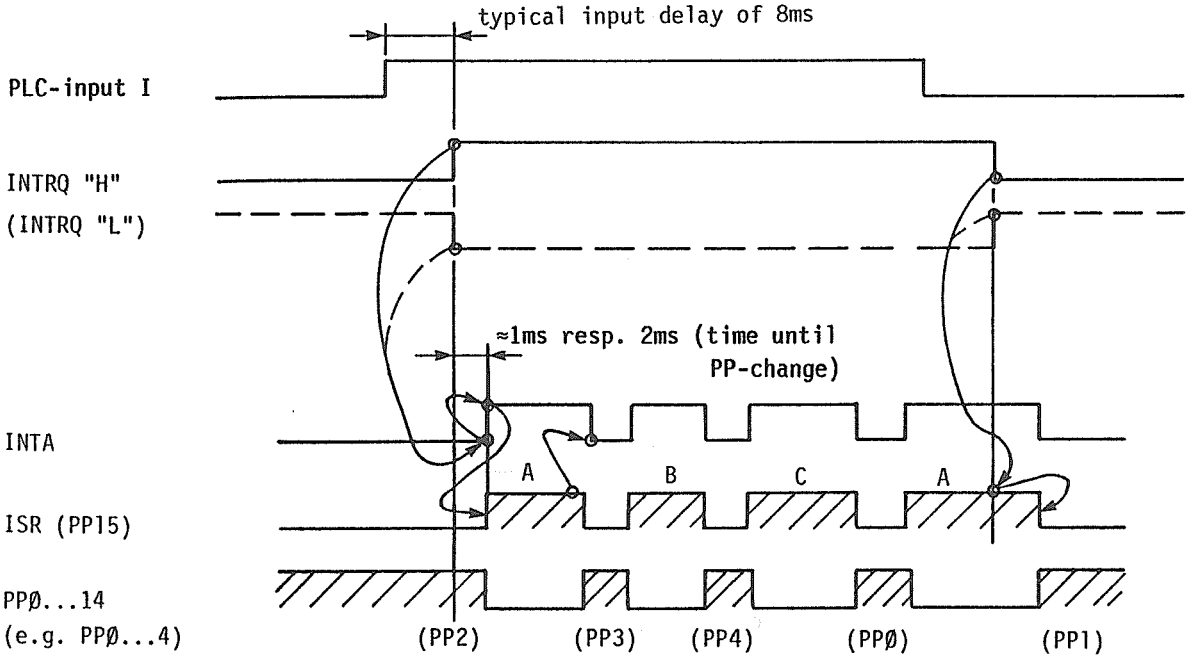
Example of a once-through exclusive ISR execution (REO INTA outside ISR, INTRQ active "L")

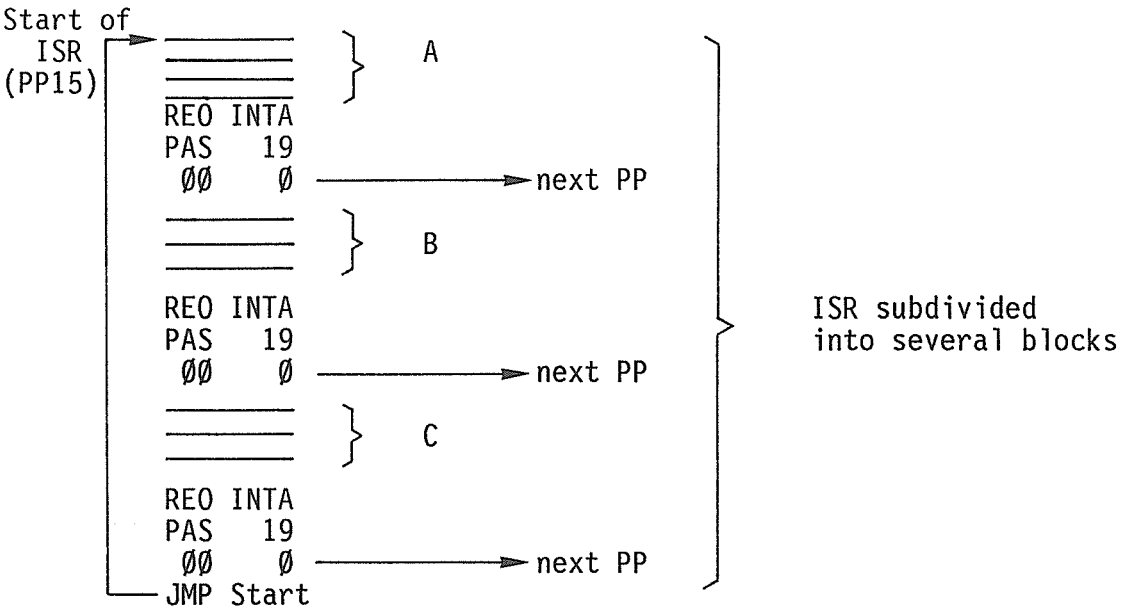
Didactic example:

```
851   PAS   19Ø      Start of ISR
  2    ØØ   873      INTRQ
  3    ØØ     3      LEVEL INTRQ active "L"
  4    ØØ     Ø      INTA
  5    ØØ    55                                  } Assignment
  6    ØØ     Ø
  7    ØØ     Ø
  8    ØØ     Ø
  9    ØØ     Ø
86Ø    ØØ     Ø
-------------------------------------------------
863   STR   256
  4    ØØ     5
  5   WIH   256      (PP-change --->)
  6   COO    47      ┌─┐ ┌─┐ ┌─┐ ┌─┐
  7   STH     3      INTRQ                       } PPØ
  8   REO    55      INTA
869   JMP   863      (PP-change --->)
-------------------------------------------------
873   SCR   26Ø      Start of ISR
  4    ØØ     6
  5   STR   257
  6    ØØ     2
  7   WIH   257
  8   COO    63      ┌─┐ ┌─┐ ┌─┐
  9   DEC   26Ø                                  } ISR (PP15)
88Ø   STH   26Ø
  1   JIO   875
  2   PAS    19      End of ISR
  3    ØØ     Ø
884   JMP   873
-------------------------------------------------
```



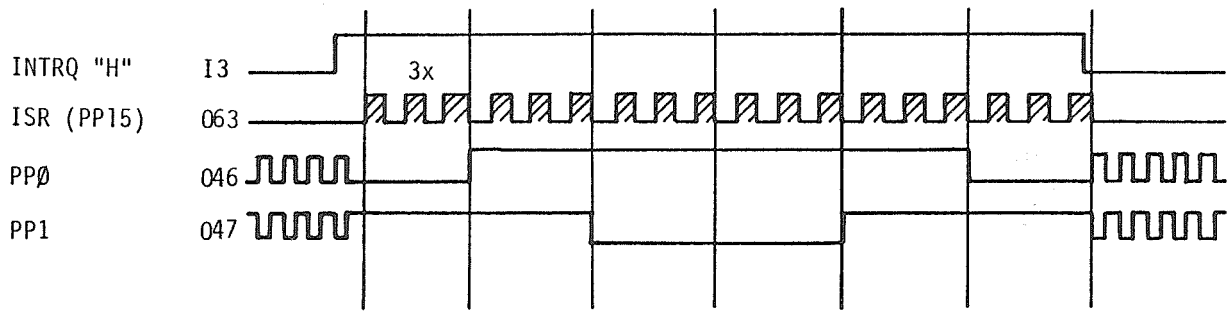
2) If REO INTA is effected within the ISR (PP15), the ISR is executed as long as INTRQ is "H" resp. "L". The instruction PAS 19 indicates the end of an exclusive run of the ISR and enables the next PP to execute a logic operation. The ISR is subsequently processed again. The ISR may also consist of several blocks, which are processed with considerably higher priority (e.g. time-critical actions) than the other PPs.





Didactic example of multiple ISR execution with high priority
(REO INTA within the ISR, INTRQ active "H")

```
899    PAS    1                                  } Assignment PP1
900    00     920
-------------------------------------------------
901    PAS    190                                } Assignment ISR (PP15)
  2    00     930    Start of ISR
  3    00       3    INTRQ
  4    00       1    LEVEL INTRQ active "H"
  5    00      55    INTA
  6    00       0
  7    00       0
  8    00       0
  9    00       0
910    00       0
-------------------------------------------------
913    STL    256                                } PP0
  4    STR    256    Indicator
  5    00       1    046
  6    COO     46
  7    JMP    918    PP-change*
918    JMP    913    PP-change
-------------------------------------------------
920    STL    257                                } PP1
  1    STR    257    Indicator
  2    00       1    047
  3    COO     47
924    JMP    920    PP-change
-------------------------------------------------
930    SCR    260    Start of ISR               } ISR (PP15)
  1    00       6
932    STR    258
  3    00       5
  4    WIH    258    Indicator
  5    COO     63    063
  6    DEC    260    3x
  7    STH    260
938    JIO    932
  9    REO     55
940    PAS     19    INTA
  1    00       0    End of ISR
942    JMP    930
```

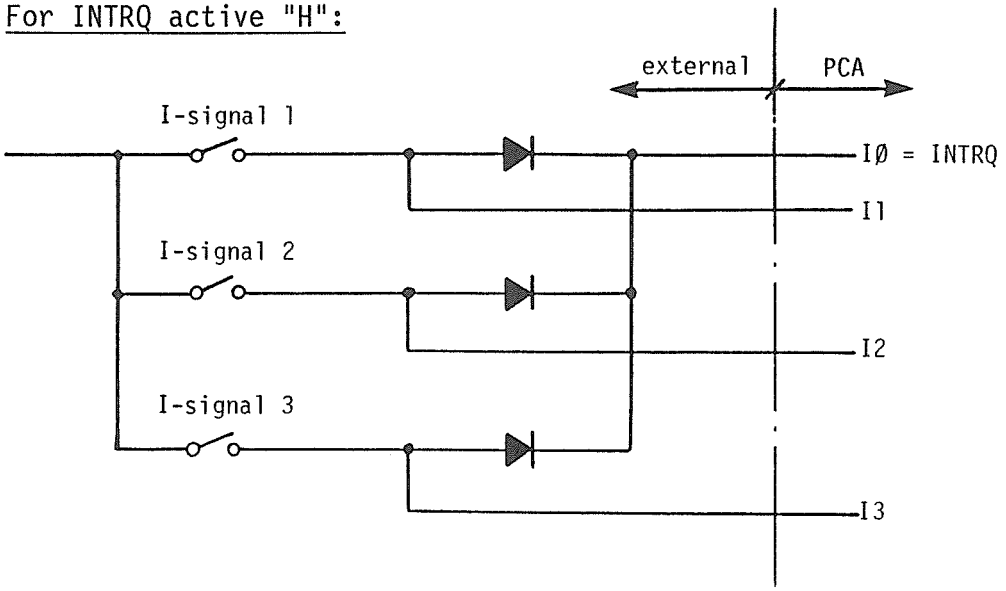


*) As a result of the artificially added JMP an additional PP-change is caused.

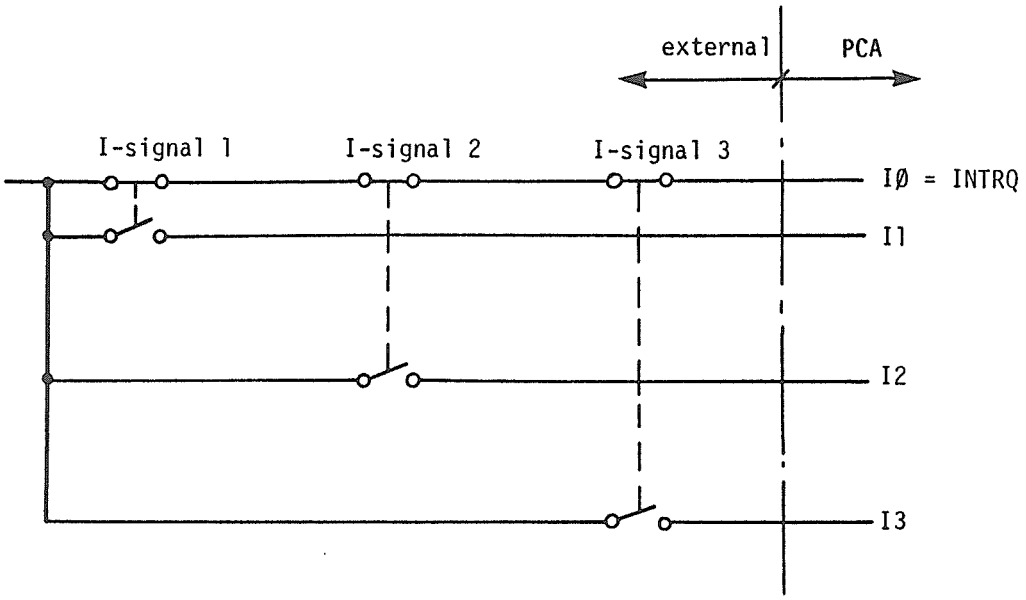
e) Various interrupt priorities

Various priority levels or differentiated ISRs are taken into account as described in the following:

- For INTRQ active "H":



- For INTRQ active "L":



The various interrupt signals are transmitted to one input each.

If INTRQ is "H", a common interrupt input is formed via a diode matrix (e.g. IØ).

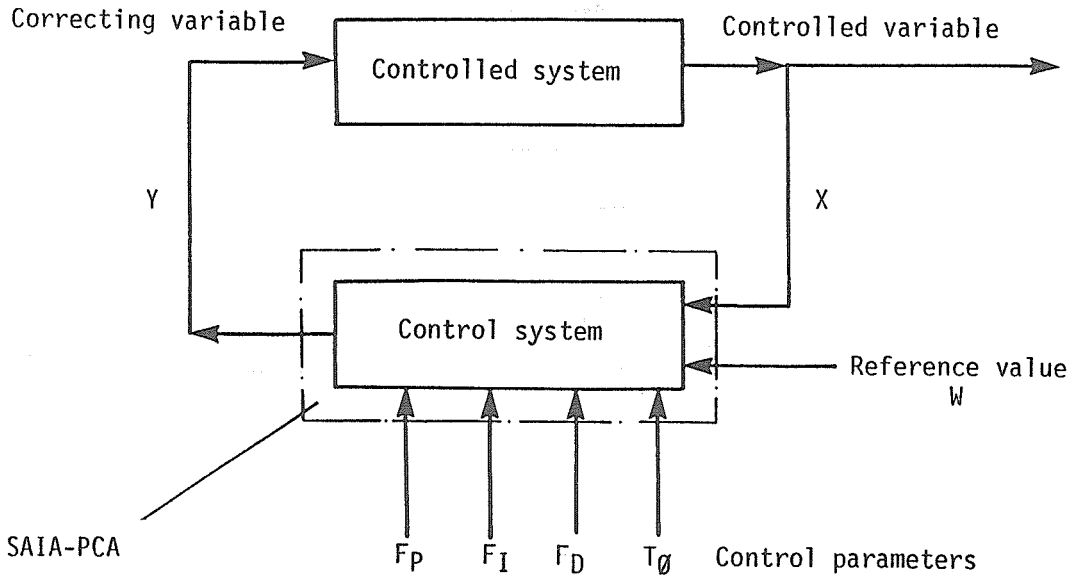
If INTRQ is "L", 2 contacts are required for each I-signal. The joint interrupt input is formed by a serial connection of the 2nd contact. The latter must be assigned as INTRQ via PAS 19Ø.

Within the ISR the respective priority with the corresponding program section can then be assigned to the individual I-signals.

PAS 2ØØ to 212 PID-control

General

The practical use of PID-control systems requires some knowledge of control engineering. The following sections can merely explain the basic principles of a control loop and processing via the instructions PAS 2ØØ...212.



The above figure shows a closed control loop. The controlled variable X appears at the output of the controlled system. This value X is supplied to the controller and compared to the reference value W. The result of this comparison, the correcting variable Y, influenced by the control parameters F_P, F_I and F_D, is in turn supplied to the controlled system according to the sampling period $T_Ø$.

The digital control system, represented by a SAIA-PCA, comprises the following parts:

- Analog input for the controlled variable X (with A/D-converter)
- Processor (CPU)
- Analog output (with D/A-converter)

Every CPU of the PCA series provides registers (so-called data blocks) for a maximum of 32 PID-control loops. This means, preassuming sufficient in- and outputs, that a maximum of 32 independent control loops can be processed with just one SAIA-PCA employing the so-called DDC-technology (Direct Digital Control).

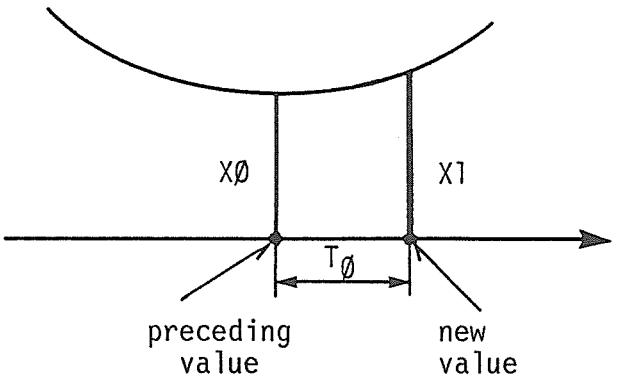
The processor (CPU of the PCA) uses the following formula to calculate the correcting variable Y:

$$Y_1 = F_P [(W-X_1) + F_I S_1 + F_D (X_0 - X_1)]$$

whereby $S_1 = S_\emptyset + (W - X_1)$

Meaning of the letters:

W = Reference value
X = Controlled variable (controlled value)
Y = Correcting variable
F_P = Proportional factor = 1 / X_P
X_P = Proportional band
F_I = Integral factor = $T_\emptyset$ / T_I
$T_\emptyset$ = Sampling period
T_I = Integral time
= T_N= Integral action time
F_D = Differential factor = T_D / $T_\emptyset$
T_D = Differential time
= T_V = Rate time



Indices Ø and 1 mean:

Ø = preceding value
1 = new value

For PCA232 (from firmware version V7.12Ø onwards), the following extended formula may be used alternatively, which also allows taking reference value jumps into account rapidly.

$$Y_{1W} = F_P \{ (W_1 - X_1) + F_I S_1 + F_D [(X_0 - X_1) - (W_0 - W_1)] \}$$

Program organisation

By calling the parameter instructions PAS 2ØØ...212 the system program of the PCA calculates the new correcting variable Y_1 based on the preceding values $X_Ø$, $Y_Ø$, $S_Ø$ and on the control parameter contained in the PAS-instruction.

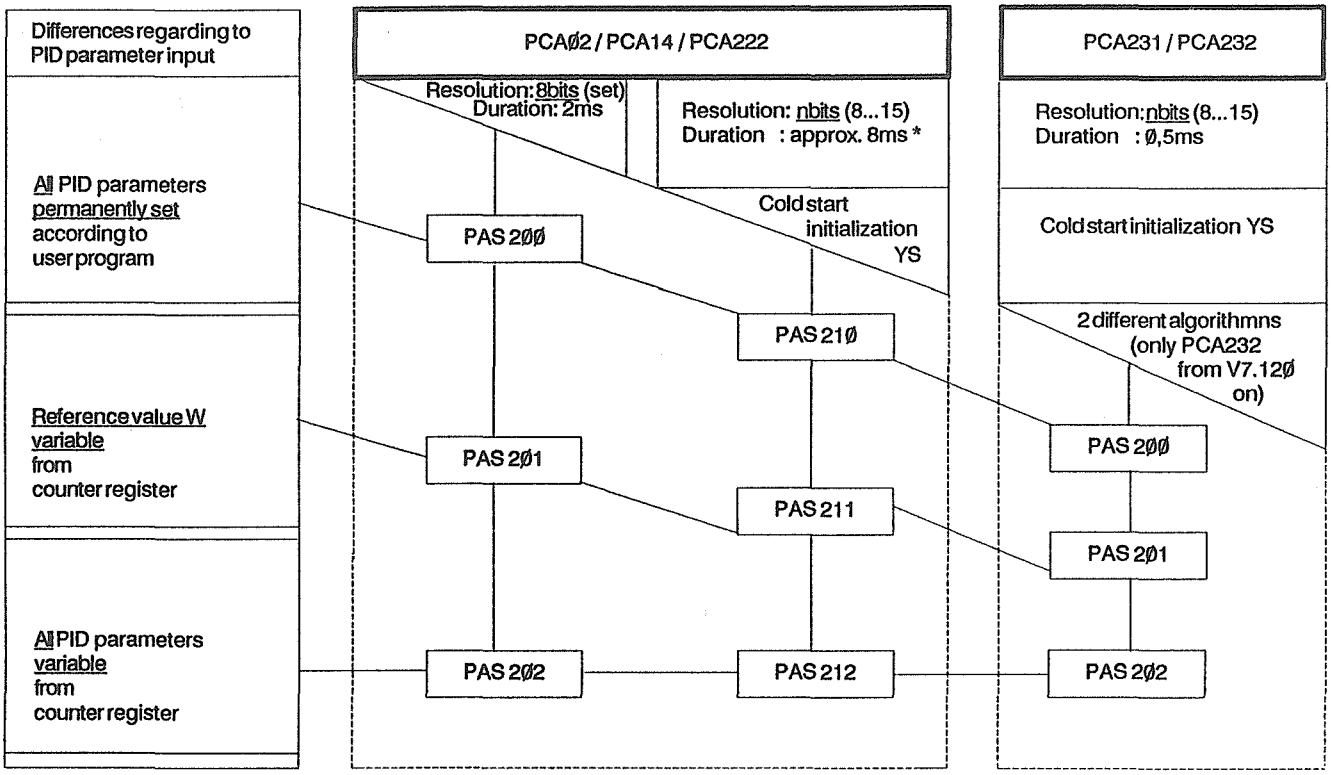
The PAS 2ØØ...212 instruction is therefore no assignment instruction, it must rather be incorporated in a circulating program for periodical execution according to the following example.

Example:

```
        .
        .
        .
    STH  256        } Check, if timer for TØ (sampling period) has elapsed.
  ┌─JIO             }
  │ STR  256        } If yes, then:
  │  ØØ    4        } - Restart sampling period e.g. Ø.4 sec.
  │
  │ SEO    3        }
  │ SEO    8        } - Read in controlled variable X from analog input
  │ SCR  26Ø        }   channel 3 i.e. transfer 12 bit value into auxiliary
  │  25   11        }   counter 26Ø (applies to module PCA2.W14).
  │
  │ PAS  2ØØ...212  }
  │ 1Ø-line         } - Calculate correcting variable Y
  │ instruction     }
  │
  │ SCR  26Ø        } - Output correcting variable Y via analog output
  │  22   11        }   channel 15 thus supplying it to the controlled system
  │ SEO   15        }   (applies to module PCA2.W14)
  │
  │       .
  │       .
  └──►    .
```

Description of parameter instructions PAS 2ØØ...212

The meaning of the instructions PAS 2ØØ...212 slightly differs between CPUs with 8-bit µP (PCAØ2, PCA14 and PCA222) and those with 16-bit µP (PCA231 and PCA232), which is evident from the following table. It also shows that the PID-parameters for every instruction group may be "specified in user program" or "variable from counter register". This enables the user to protect the control parameters from being altered or to select them via BCD-switches, for example.



PAS 2ØØ, 2Ø1, 2Ø2 — For series PCAØ2 / PCA14 / PCA222
(fast PID-algorithm for 8-bit analog modules)

Detailed description of parameter instructions 2ØØ and 2Ø1

Overview				Comment		Numerical range
1	PAS(29)	2ØØ	;	PID-control instruction		2ØØ or 2Ø1
2	ØØ	DBLK	;	Data Block	No.	31...Ø
3	ØØ	W	;	Reference value	Value	Ø...255*
4	ØØ	X, Y	;	Counter addr. for controlled variable or correcting variable	ADDR	256...319
5	XX	F_I	;	Integration factor ($T_Ø/T_I$)	Value	Ø...32767
6	XX	F_D	;	Differential factor ($T_D/T_Ø$)	Value	Ø...32767
7	XX	F_P	;	Proportional factor ($1/X_P$)	Value	Ø...32767
8	ØØ	DR	;	Dead range	Value	Ø...255
9	ØØ	Ø	}	NUL		
1Ø	ØØ	Ø	}			

*) for PAS 2Ø1: Counter addr. 256...319

- 1st line

PAS 2ØØ: The reference value W from line 3 is to be programmed directly
PAS 2Ø1: The reference value W from line 3 is taken from the respective counter.

- 2nd line Data Block

A maximum of 32 independent PID-loops can be defined. In order to be able to store values $X_Ø$, $Y_Ø$ and $S_Ø$ in separate registers, which are necessary for calculating value Y, numbers 31...Ø of the data block must be specified for each control loop. In order not to affect the counter registers (up to C479), it is recommended to start with Data Block 31 (see "Register organisation" in the hardware manual).

- 3rd line Reference value W

. For PAS 2ØØ put in the 8-bit value Ø...255 directly (immediate).
. For PAS 2Ø1 put in the respective counter address (256...319), from which the value (Ø...255) must be taken. PAS 2Ø1 is therefore suitable for the external input of a reference value.

- 4th line Auxiliary counter for X or Y respectively

To read in the actual controlled variable X1 and to read out the actual correcting variable Y1 an auxiliary counter (addr. 256...319) is required. See C26Ø in the example of section "Program organisation". This must correspond with the address of line 4.

- Lines 5...7 F_I, F_D, F_P

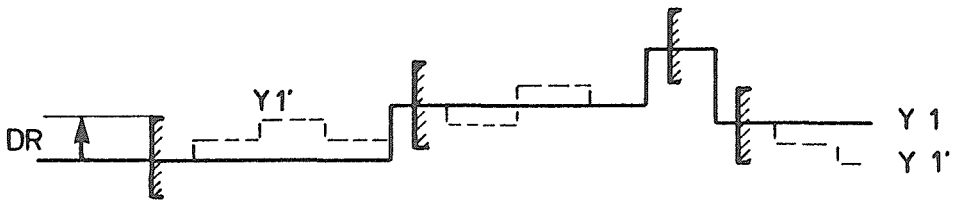
  In practice these values amount to between Ø.1 and 7.99. To obtain sufficient resolution, the value must be multiplied by 256.

Practical value	Input value in the operand	
Ø.1	25	
1.Ø	256	
7.99	2Ø45	
24.25	62Ø8	[C] ---> Ø 3 / ØØ64

  Even operands exceeding 2Ø47 can be entered directly. The PLC stores the modulo in the operand and the multiple of 2Ø48 in the code. Key [C] (convert) is used to represent the value in one way or the other.

- Line 8 Dead range DR

  Little fluctuations of the correcting variable Y can be compensated for by determining a dead range, thus avoiding oscillations.



If the calculated value Y1', remains within the dead range DR, the output correcting variable Y1, is not changed.

DR is entered as a value between Ø...255 in the same unit as the reference value W. DR is active for fluctuations both downwards and upwards.

Important:

PAS 2ØØ, 2Ø1 and 2Ø2 are always executed independently of the ACCU state. Execution of the 1Ø-line parameter instruction takes approximately 2 ms.

Detailed description of the parameter instruction PAS 2Ø2

For testing and commissioning a control loop it can be of advantage to enter all parameters externally and to change them as desired during operation. The PAS 2Ø2 instruction meets this requirement.

Overview:

```
 1  PAS(29)    2Ø2   ;  PID-control instruction
 2      ØØ   DBLK   ;  Data Block
 3      ØØ    FIC   ;  First address of 6 counters
                       for values W, X/Y, FI, FD, FP, DR
 4      ØØ      Ø   ]
 5      ØØ      Ø   ]
 6      ØØ      Ø   ]
 7      ØØ      Ø   } NUL
 8      ØØ      Ø   ]
 9      ØØ      Ø   ]
1Ø      ØØ      Ø   ]
```

- Line 3 FIC (first counter)

  If e.g. address 26Ø stands for FIC, the following assignment ensues automatically:

		Numerical values for C	
C 26Ø	Reference value W	Ø...255	
C 261	Auxiliary counter for X1 or Y1	Ø...255	
C 262	F_I Integration factor	Ø...32767	]
C 263	F_D Differential factor	Ø...32767	} *
C 264	F_P Proportional factor	Ø...32767	]
C 265	DR Dead range	Ø...255	

  The numerical values which must be entered in the respective counter registers first have the same meaning as those in PAS 2ØØ and 2Ø1.

---

*) In order to obtain the numerical values, which must be entered in the counters, multiply the practical values for F_I, F_D and F_P by 256.

Part L
Example 6

PAS 21Ø, 211, 212 For series PCAØ2 / PCA14 / PCA222

PAS 2ØØ, 2Ø1, 2Ø2 For series PCA231 / PCA232

Resolution selectable from 8...15 bits

For historical reasons, the various series do not use the same instruction for the same function. The processor module PCA2.M31 or M32 developed after the PCA14 has featured a selectable resolution for calculation of the correcting variable Y right from the start after the introduction of analog modules with a 12-bit resolution of the PCA1 series, an appropriate PAS-instruction also had to be created for PCA14. In order to ensure compatibility with existing programs, the new instruction needed to be given a name different from the 8-bit instruction, namely PAS 21Ø etc.

In the following description, only the instructions PAS 21Ø, 211, 212 for series PCAØ2 / PCA14 / PCA222 will be mentioned. However, all versions are also valid for the instructions PAS 2ØØ, 2Ø1, 2Ø2 for series PCA231 / 232 (see also table on page 31H).

There are only two differences:

- Execution time for PAS 2ØØ...212 < PCAØ2 / 14 / 222 : approx. 8 ms
  PCA231 / 232 : Ø,5 ms

  Despite the execution time of 8ms, the serial interface is operated every 1ms, so that communication up to 96ØØ bauds is not impaired concerning this instruction.

- From firmware version V7.12Ø, the PCA232 provides an extended algorithm, which also allows taking reference value jumps into account rapidly (see line 9 of the PAS instruction).

Detailed description of the parameter instructions PAS 21Ø and 211 for PCAØ2 / PCA14 / PCA222 (resp. 2ØØ and 2Ø1 for PCA231 / PCA232)

Overview				Comment	Numerical range	
1	PAS(29)	21Ø	;	PID-control instruction	21Ø or 211	
2	n	DBLK	;	Resolution n bit and data block no.	31... Ø	
3	ØØ	W	[1];	Reference value	Value Ø... 4Ø95	[2][3]
4	ØØ	X, Y	;	Counter addr. for controlled or correcting variable	ADDR 256... 319	[2]
5	XX	F_I	;	Integration factor ($T_Ø/T_I$)	Value Ø...32767	
6	XX	F_D	;	Differential factor ($T_D/T_Ø$)	Value Ø...32767	
7	XX	F_P	;	Proportional factor ($1/X_P$)	Value Ø...32767	
8	ØØ	DR	;	Dead range	Value Ø... 4Ø95	[2]
9	ØX	YS	;	Counter addr. for cold start value YS	ADDR 256... 319	[2]
1Ø	ØØ	Ø	;	Ø		

- Line 1:
  PAS 21Ø: The reference value W of line 3 must be programmed directly.
  PAS 211: The reference value W of line 3 is taken from the respective counter.

- Line 2: Resolution n-bit and data block
  In the code, n indicates the number of bit to be processed for the values W, X, Y, YS and DR:

  n = Ø8 [4] ≙ 8 bits ---> Ø... 255
  12 [5] ≙ 12 bits ---> Ø... 4Ø95
  15 ≙ 15 bits ---> Ø...32767

A maximum of 32 independent PID-loops can be defined. In order to be able to store values $X_Ø$, $Y_Ø$ and $S_Ø$ in separate registers, which are necessary for calculating value Y, the number of data block 31...Ø must be specified for each control loop in the operand. In order not to affect the counter registers for PCA14 (up to C479), it is recommended to start with data block 31 (see "Register organisation" in the hardware manual).

- Line 3: Reference value W
  . For PAS 21Ø enter the 12-bit value Ø...4Ø95[3] directly (immediate).
  . For PAS 211 enter the respective counter address (256...319), from which the value (Ø...4Ø95)[3] must be taken. PAS 211 is therefore suitable for external input of the reference value.

- Line 4: Auxiliary counter for X or Y
  For reading in the actual controlled variable X_1 and for outputting the actual correcting variable Y_1 an auxiliary counter (address 256...319) is required. See C26Ø in the example of section "Program organisation". This must correspond to the address in line 4.

---

1) For PAS 211: Counter addr. 256...319
2) The values in the operand or in the counter depend on n in line 2.
3) Applies to n = 12 of the 2nd line.
4) With PCA14 it is also possible to enter ØØ instead of Ø8 for 8 bits.
5) With PCA232 it is also possible to enter ØØ instead of 12 for 12 bits.

- Line 5...7: F_I, F_D, F_P

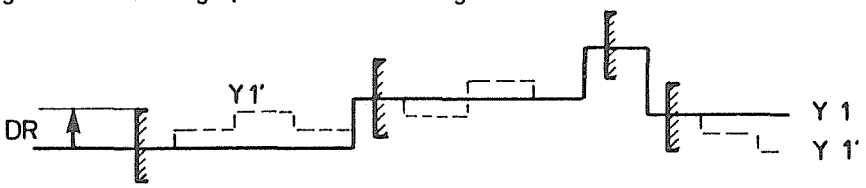
In practice, these values are usually in the range from Ø.1 to 7.99 approximately. To obtain sufficient resolution, the value to be entered is multiplied by 256.

Practical value	Input value in the operand
Ø.1	25
1.Ø	256
7.99	2Ø45
24.25	62Ø8 [C] ---> Ø 3 / ØØ64

Even operands greater than 2Ø47 can be entered directly. The PLC stores the modulo of 2Ø48 in the operand and the multiple of 2Ø48 in the code. In order to represent the value in one way or the other, use key [C] (convert) of the P1Ø/PØ5 programming unit.

- Line 8: Dead range DR

Minor fluctuations of the correcting variable Y are compensated for by determining a dead range, thus avoiding oscillations.



If the calculated value Y_1' remains within the dead range DR, the output correcting variable Y_1 is not changed.

DR is entered as a value between Ø...4Ø95 having the same unit as the reference value W. DR is applied for fluctuations both downwards and upwards.

- Line 9: Cold start value YS (initial correcting variable)

Since the data block registers 31...Ø are non-volatile, the respective values must be set to a certain starting value when being used for the first time (or for another controlled system).

If the value YS is loaded into the counter CYS of the operand addressed in line 9, the following values are used for the cold start (and stored in the data block register):

Data block: $X_Ø = X$; $Y_Ø = YS$; $SØ = \dfrac{YS/F_P - (W - X)}{F_I}$

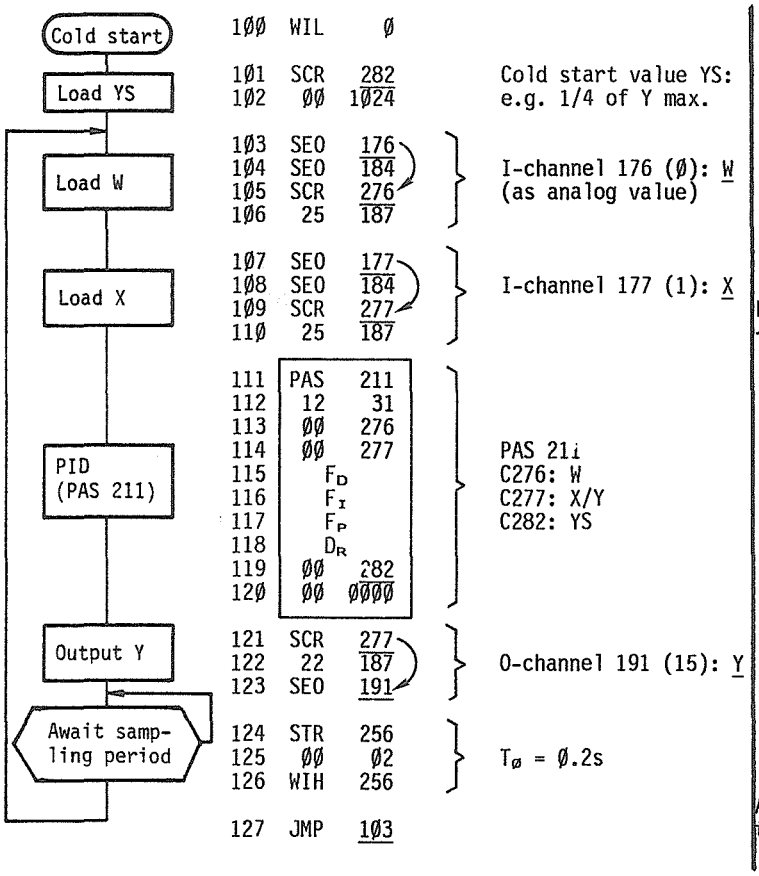
Correcting value: $Y = YS$

After the first execution of the PAS-instruction the counter CYS is automatically reset (= Ø) and will not be taken into account during the next run. In addition to the cold start, smooth transitions from manual to automatic operation or parameter changes can also be achieved with the CYS counter (see programming example 7).

In the Code of line 9, ØØ is used to activate the regular algorithm for calculating correcting variable Y. If Ø1 is used, the PCA232 (from V7.12Ø onwards) applies the extended formula with the reference value jumper (see page 29H).

As operation of the analog module of the PCA2 series is easier than operation of the corresponding module of the PCA1 series, CPU PCA2.M22 and analog module PCA2.W14 at basic address 176 is used here.

A cold start control program is thus organised as follows:

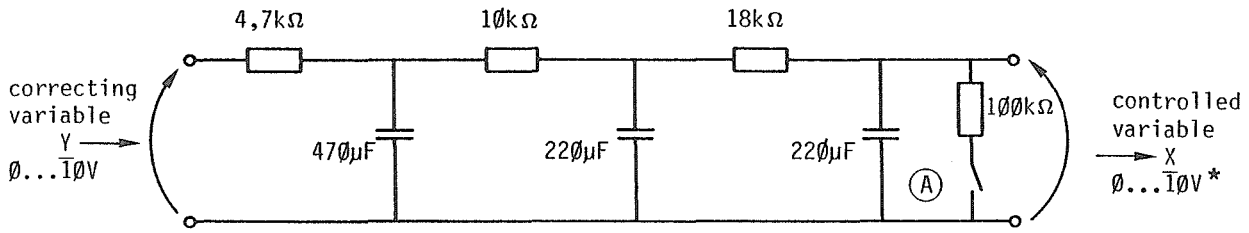


Basic address	Ø	176	
<-- 0	15	191	Ⓨ<-- C277
	.	.	
	.	.	
	.	.	
0	12	188	
I	7	183	
	.	.	
	.	.	
	.	.	
	.	.	
	.		
--> I	1	177	Ⓧ --> C277
--> I	Ø	176	Ⓦ --> C276
			YS --> C282

Addresses of the analog module type PCA2.W14

Control example: By means of this program, the following RC circuit (simulating a controlled system) must be PID controlled.

Simulated controlled system:



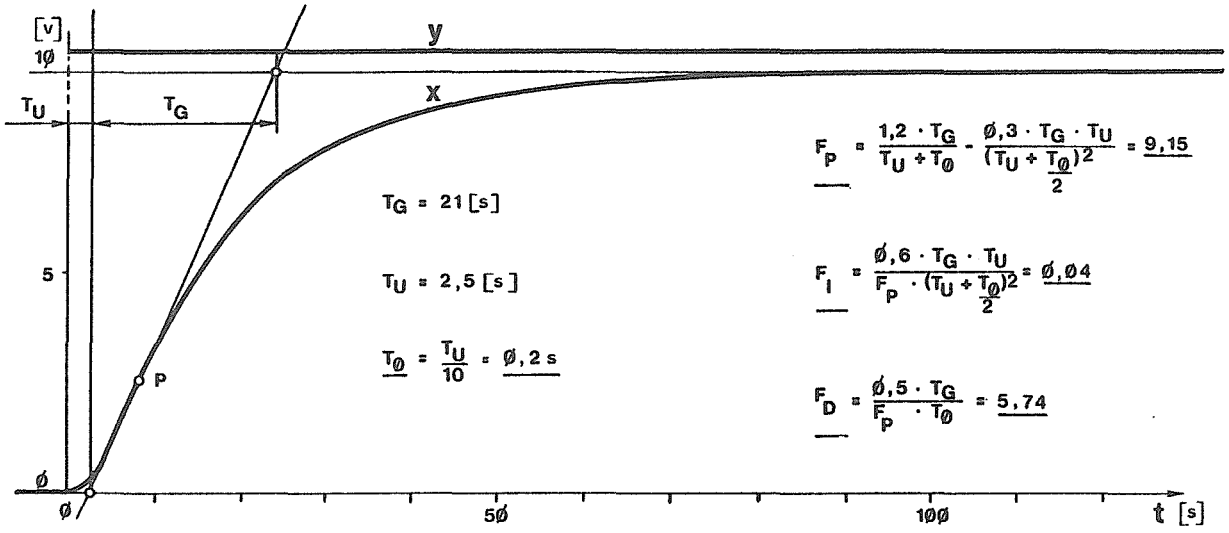
Switch (A) produces a disturbance of about 20% in the controlled system.

A record of the step function response (Ø ---> 1ØV) allows calculating the parameters F_P , F_I , F_D and $T_ø$ according to Takahashi's algorithm.

---

*) The measuring instrument for X must have an input resistance of at least 1MΩ.

Evaluation of the step function response of a controlled system



The above calculated values must be multiplied by 256 for the input using PAS 211:

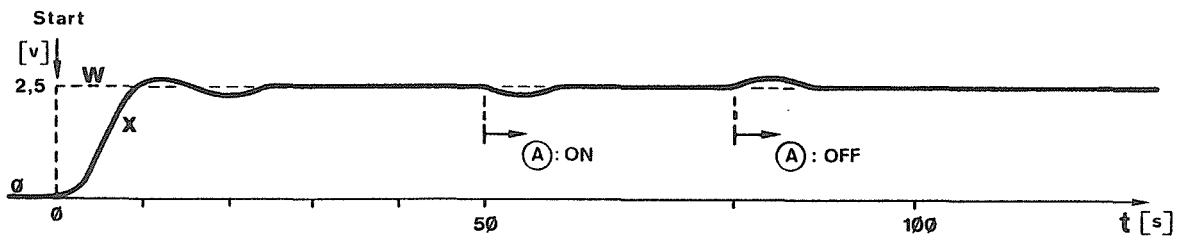
$F_I = F_I$ x 256 = 1Ø

$F_D = F_D$ x 256 = 1469

$F_P = F_P$ x 256 = 2341

```
PAS    211
12      31   ;  12 bits, data block 31
ØØ     276   ;  Counter for W
ØØ     277   ;  Counter for X/Y
ØØ      1Ø   ;  FI
ØØ    1469   ;  FD
Ø1     293   ;  FP ( C = E E 2341)
ØØ      ØØ   ;  DR
ØØ     282   ;  Counter for YS
ØØ      ØØ   ;  --
```

Behaviour of the controlled system with the above parameters for a reference value W = 1Ø24 (2.5V):



Ⓐ : Simulated disturbance (≈2Ø%).

Detailed description of the parameter instruction PAS 212

For testing and starting a control loop it can be of advantage to enter all parameters externally and to change them as desired during operation. The PAS 212 instruction (PAS 2Ø2 for PCA231/232) meets this requirement.

Overview:

```
 1  PAS(29)    212   ;  PID-control instruction
 2       n    DBLK   ;  Resolution n bit and data block
 3      ØØ     FIC   ;  First address of 7 counters
                        for values W, X/Y, FI, FD, FP, DR, YS
 4      ØØ       Ø   ⎫
 5      ØØ       Ø   ⎪
 6      ØØ       Ø   ⎪
 7      ØØ       Ø   ⎬  NUL
 8      ØØ       Ø   ⎪
 9      ØØ       Ø   ⎪
1Ø      ØØ       Ø   ⎭
```

- Line 2: see PAS 21Ø

- Line 3: FIC (first counter)

  If e.g. address 276 stands for FIC, the following assignment ensues automatically:

		Numerical values for C	
C 276	Reference value W	Ø... 4Ø95	2)
C 277	Auxiliary counter X1 or Y1	Ø... 4Ø95	2)
C 278	F_I Integration factor	Ø...32767	1)
C 279	F_D Differential factor	Ø...32767	1)
C 28Ø	F_P Proportional factor	Ø...32767	1)
C 281	DR Dead range	Ø... 4Ø95	2)
C 282	YS Cold start value	Ø... 4Ø95	2)

  The numerical values, which must be entered into the respective counter registers first, have the same meaning as for the PAS 21Ø and 211 instructions.

  Important:

  All PAS 2ØØ...212 are always executed independently of the ACCU state.

---

1) In order to obtain the numerical values which must be entered into the counters, multiply the practical values for F_I, F_D and F_P by 256.

2) Values for analog module with 12-bit resolution (n).

PAS 25Ø/251 Rotation register, shift register and FIFO-stack register

PAS 25Ø Rotation and shift register

The 1Ø-line instruction PAS 25Ø can be used to define and process a rotation or shift register in the bit processor which is preferably assigned to the flag range - a register can of course be defined and processed using readable outputs, too. The PAS instruction is organised as follows:

```
1    PAS   (29)    25Ø   ;  Activation of the parameter function
2          ØØ      SADD  ;  Source address of register SADD        1)
3          ØØ      DADD  ;  Destination address of register DADD   1)
4          ØØ      NNN   ;  Width of the information (N bit)       2)
5          ØØ       Ø    )
6          ØØ       Ø    |
7          ØØ       Ø    }  NUL
8          ØØ       Ø    |
9          ØØ       Ø    |
1Ø         ØØ       Ø    )
```

Depending on whether SADD or DADD is at the lower address the rotation or shift register is shifted upwards or downwards. The parameter function PAS 25Ø is executed independently of the ACCU state and does not influence the ACCU.

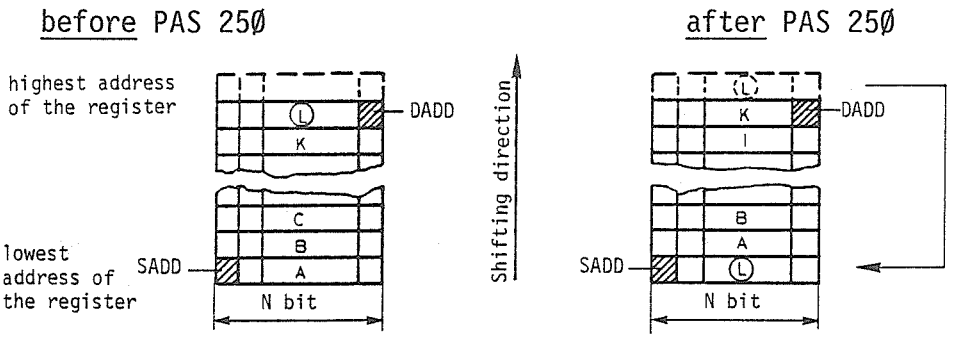
---

1) Both SADD (source address) and DADD (destination address) are indexable.

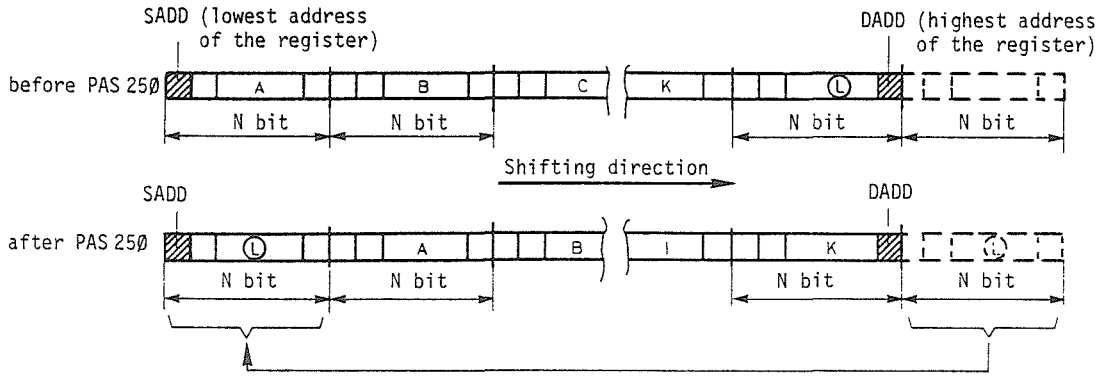
2) The width of the information can be 1...255 bits.
   NNN = 1...255: Fixed value
   NNN = 256...511: The information N bit is contained in the respective counter.

Functional description

a) SADD < DADD ≙ shift upwards

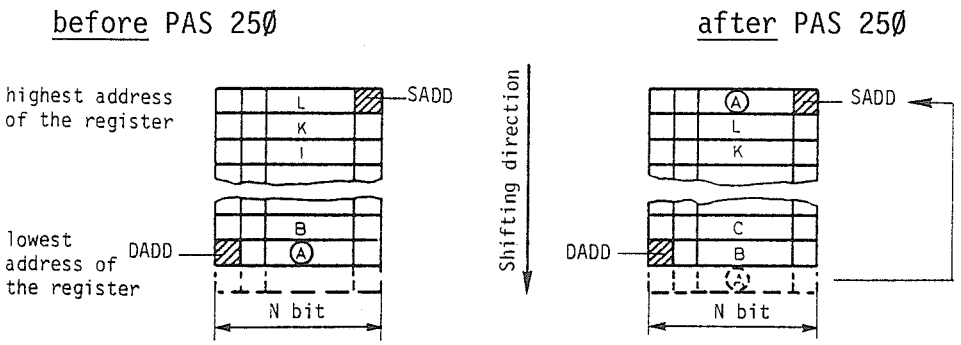


The same register shown differently:



The figures show that starting with address DADD the following addresses of N bit must be reserved, since an additional register level is used as a circulating buffer.

b) SADD > DADD ≙ shift downwards



The figures show that before address DADD the addresses of N bit immediately located in front must be reserved.

As is evident from the figures, the instruction PAS 25Ø processes a rotation register. If this must be turned into a linear shift register, make sure in case a) (shift upwards) that the information is taken from level DADD. After shifting, the new information to be introduced must be transferred to level SADD.

Part L
Examples 8 and 9

PAS 251 Input into the FIFO register (First in - First out)

The 1Ø-line instruction PAS 251 is used to write information of N bit into the FIFO register and to shift it to the last free space.

The size of the FIFO is defined by the source address SADD and the destination address DADD.The PAS-instruction is organised as follows:

```
 1   PAS    (29)    251   ;  Activation of the parameter function
 2           ØØ     SADD  ;  Source address of the register SADD      1)
 3           ØØ     DADD  ;  Destination address of the register DADD 1)
 4           ØØ     NNN   ;  Width of the register (N bit)           2)
 5           ØØ      Ø    )
 6           ØØ      Ø    |
 7           ØØ      Ø    }  NUL
 8           ØØ      Ø    |
 9           ØØ      Ø    |
1Ø           ØØ      Ø    )
```

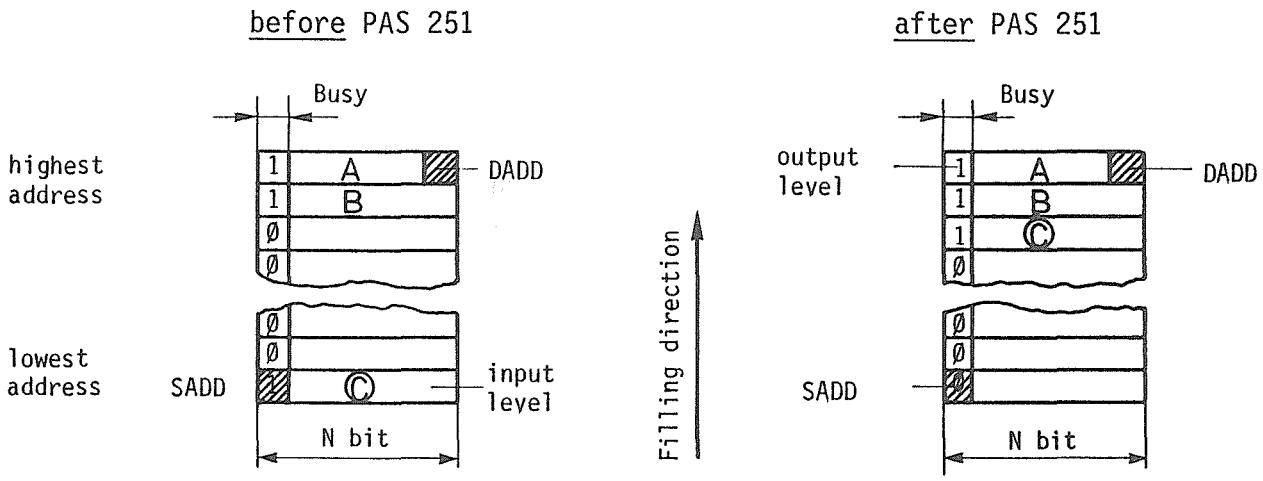
SADD must always be on a lower address than DADD. The parameter function is executed independently of the ACCU state. PAS 251, however, does influence the ACCU (see two pages later).

---

1) Both SADD (source address) and DADD (destination address) are indexable.

2) The width of the register can be 1...255 bits.

Functional description

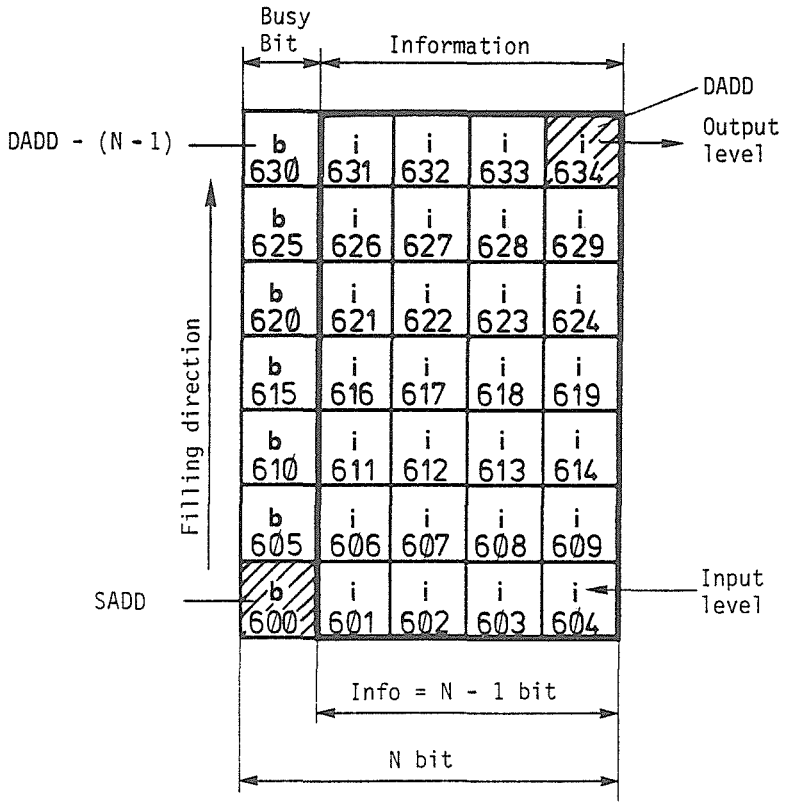


The above figures show that in the FIFO-register a new value of N bit is stored on the last free space in the direction SADD ---> DADD.

In order to prevent information from being lost in the FIFO-register, make sure every time a new value is input that the FIFO is not yet full. If this is not done, the value last introduced will be replaced by the new value. In order to be able to carry out this test a "busy" bit is reserved in front of each register level. This bit will be "H", if the respective level is full.

It is sufficient to test the "busy" bit of the input level before every new input via PAS 251. Then make the right decision depending on the result.

The example of a FIFO-register is thus organised as follows:



SADD = 6ØØ
DADD = 634
N = 5 bits
Information = 4 bits

Proceed:

a) If the value last introduced does not have to be erased, check in advance whether SADD (busy bit) is "H".

   SADD = H : FIFO full
   SADD = L : free spaces in the FIFO

b) The user now enters information of (N-1) bit at the input level and sets the bit "SADD" to "H".

c) Execution of PAS 251.
   After PAS 251, SADD is automatically set to "L", provided that the FIFO is not yet full. It remains "H", if all register levels are full.

d) Processing of the result after PAS 251:

   ACCU = 1: No errors detected.

   ACCU = Ø: Faulty definition of the FIFO, PAS 251 was not executed.

   SADD = H: FIFO full. The input of another value replaces the value last introduced.

   SADD = L: FIFO not full, i.e. free spaces for further information.

   DADD - (N-1) = L: FIFO empty.

Notes:

PART I INSTRUCTIONS OF WORD PROCESSOR, LEVEL 3 (only PCA232 or PCA231)

PART I INSTRUCTIONS OF WORD PROCESSOR, LEVEL 3 (only PCA232 or PCA231)

I 1 Changing from word processor and instruction set of the word processor

NOP 1248 Changing from bit to word processor

The 5 bits available to the instruction code can be used to define only 32 instructions. With the key word NOP 1248 32 additional instructions of the word processor can be used.

```
Example:      :
              STH      3
              ANH      4
              ORH     1Ø
              JIZ
              NOP  W 1248
              LAC  W  256
              ADD  W  13Ø
              EXG  W   Ø
              SAC  W  256
              EWP  W   Ø
              SCR     256
               22     311
                :
```

Program part in the bit processor

Instruction for switching over to word processor

Program part in the word processor

End of word processor

Program part in the bit processor

NOP 1248 is always executed independently of the ACCU state.

NOP 1248 sets the ACCU = 1

Certain word processor instructions set the ACCU = Ø, if e.g. capacity limits are exceeded or in the case of division by Ø.
By checking the ACCU state after EWP it can be determined whether the admissible limits were exceeded.

```
NOP  W 1248    - Instruction for switching over to word processor, ACCU = 1

LAC  W  256      C256 in AØ
EXG  W   Ø       C256 in A1
CLA  W   Ø       AØ = Ø
DIV  W   Ø       A1 : Ø ---> ACCU = Ø

EWP  W   Ø     - End of word processor

JIZ              Jump into error message routine
```

Program part in the word processor

Remark: From NOP 1248 to JIZ no change from one parallel program to another occurs. This is important in the case of time-critical PPs, since the word processor programs are divided into smaller parts.

Instruction set of the word processor

Num. code	Mnemo code	Operand	Instruction	Description	Data format	
ØØ	NOP	1248	---	Change from bit to word processor ACCU set = 1		

Data transfer instructions

Ø1	RRG	Rn	Read Register	Read word Rn and store it in R4	8 bits	i
Ø2	WRG	Rn	Write into Register	Write the word of R4 into Rn	8 bits	i
Ø3	RRE	Rn	Read Register and write in Elements	Read word Rn and store it in the 8 elements En...En-7, addressed via A1	8 bits	i
Ø4	WRE	Rn	Write Elements into Register	Write Register Rn with elements En...En-7, addressed via A1	8 bits	i
Ø5	LAR	Rn	Load AØ with Registers	Load register block Rn into AØ	5x8 bits	i
Ø6	SAR	Rn	Store AØ in Registers	Store AØ into register block Rn	5x8 bits	i
Ø7	LAC	Cn	Load AØ with Counter	Load AØ with counter Cn	BCD	i
Ø8	SAC	Cn	Store AØ in Counter	Store AØ in counter Cn	BCD	i
Ø9	WEL	En	Write into elements { Lower / Upper } digit	Write { R4 ($1Ø^Ø$) / R4 ($1Ø^1$) } digit into elements En...En-3	4 bits	i
1Ø	WEU	En				

Counting and jump instructions

11	INR	Rn	Increment Register	Incr. } BCD-value by 1 and set "carry" if result { > 99		i
12	DER	Rn	Decrement Register	Decr. } BCD-value by 1 and set "carry" if result { < Ø		i
13	SNC	Ø En	Skip if no "carry" Skip if En = Ø	Skip the next instruction, if { "carry" = Ø / En = Ø		i
24	SEW	Ø En	Skip to EWP if no "carry" Skip to EWP if En = Ø	Skip to EWP or NOP 1248 if { "carry" = Ø / En = Ø		i

(i) = indexable

Processing of arithmetic registers

14	CLA	Ø,1,2	Clear A	Clear register AØ or A1 or A2		
15	LAI	K	Load A immediately	If K ≤ 99 ---> Load data into R4 If K = 1ØØ...2Ø47 ---> Load data into AØ		
16	DBN	Ø	Decimal to Binary	Convert AØ decimal into AØ binary		
17	BND	Ø	Binary to Decimal	Convert AØ binary into AØ decimal		
18	ROR	Ø, 1	Rotate Register	Rotate R4...RØ, resp. RØ...R4		
19	ROA	Ø, 1	Rotate A	Rotate AØ...A2, resp. A2...AØ		
2Ø	EXG	Ø	Exchange A1 with AØ	Exchange A1 with AØ		
		Rn	Exchange A1 with Rn...Rn-4	Exchange A1 with Rn...Rn-4		i

Arithmetic instructions

25	CMP	En	Compare AØ with A1	Compare AØ with A1 A1 > AØ ---> En = 1 A1 = AØ ---> En-1 = 1 A1 < AØ ---> En-2 = 1	BCD	i
26	SQR	Ø	Square Root from A1	A1 ---> A1, only integers	BCD	
27	ADD	Ø	Add AØ to A1	A1 + AØ ---> A1, "carry"	BCD	
		K	Add K to A1	A1 + K ---> A1, "carry" (K=1...2Ø47)	BCD	
28	SUB	Ø	Subtract AØ from A1	A1 - AØ ---> A1, "carry"	BCD	
		K	Subtract K from A1	A1 - K ---> A1, "carry" (K=1...2Ø47)	BCD	
29	MUL	Ø	Multiply A1 by AØ	A1 . AØ ---> A1, "carry"	BCD	
		K	Multiply A1 by K	A1 . K ---> A1, "carry" (K=1...2Ø47)	BCD	
3Ø	DIV	Ø	Divide A1 by AØ	A1 : AØ ---> A1, Rest in AØ, "carry"	BCD	
		K	Divide A1 by K	A1 : K ---> A1, Rest in AØ, "carry" (K=1...2Ø47)	BCD	

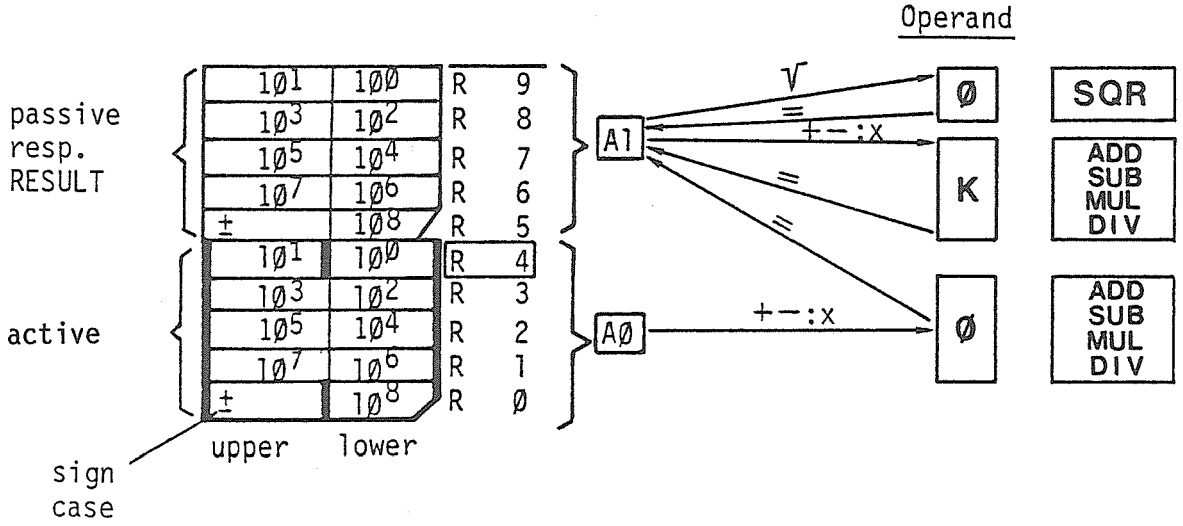
Various instructions

ØØ	NOP	Ø	No operation	No operation		
21	CLK	En	Clock source	Assignment of a timing pulse source		
22	SHI	Rn	Shift registers	Shift words as from R2Ø to Rn by one upwards		i
23	TXT	Txn	Text	Start of text output		i
31	EWP	Ø	End word processor	End of the operation in the processor		

(i) = indexable

I 2 Arithmetic instructions

The arithmetic operations are executed exclusively between the A-registers A1 and AØ or a constant K in the operand. A1 contains the passive operator. In A1 the result of the operation is stored, while AØ remains unchanged (with the exception of division).



For the arithmetic operations the value must be available in BCD-representation. Positive and negative BCD-values consisting of integers are processed up to a capacity of ±1Ø⁹-1 (±999'999'999). The preceding sign is placed before decimal 1Ø⁸ and has the following meaning:

Ø = positive +
9 = negative -

When a register block is read, all bit patterns deviating from Ø are fitted with a negative preceding sign.

For the data transfer, any bit patterns of 1Ø x 4 bits can be used.

ADD Addition

ADD: Add AØ (or K) to A1

Mnemo code	Numerical code	Operand	Operation
ADD	27	Ø	A1 + AØ ---> A1
		K = 1...2Ø47	A1 + K ---> A1

Capacity: ±999'999'999 or ±(1Ø⁹-1)
In case of "overflow", carry will be = 1 and ACCU = Ø. A1 contains only the remainder.

SUB Subtraction

SUB: Subtract AØ (or K) from A1

Mnemo code	Numerical code	Operand	Operation
SUB	28	Ø	A1 - AØ ---> A1
		K = 1...2Ø47	A1 - K ---> A1

Capacity: ±999'999'999 or ±(1Ø⁹-1)

In case of "overflow", "carry" will be = 1 and ACCU = Ø. A1 contains only the remainder.

MUL Multiplication

MUL: Multiply A1 by AØ (or by K)

Mnemo code	Numerical code	Operand	Operation
MUL	29	Ø	A1 x AØ ---> A1
		K = 1...2Ø47	A1 x K ---> A1

Capacity of the result: ±999'999'999 or ±(1Ø⁹-1)

In case of "overflow", "carry" will be = 1 and ACCU = Ø. A1 contains the maximum value of ±999'999'999.

DIV Division

DIV: Divide A1 by AØ (or by K)

Mnemo code	Numerical code	Operand	Operation
DIV	3Ø	Ø	A1 : AØ ---> A1
		K = 1...2Ø47	A1 : K ---> A1

Any remainder is stored in AØ.
The "carry" is set to = 1 the ACCU = Ø in the case of division by Ø. In this case, the operation is not executed and all data remains unchanged.
Capacity limit for the result: ±999'999'999.

SQR Square root

SQR: Square Root from A1

Mnemo code	Numerical code	Operand	Operation
SQR	26	Ø	$\sqrt{A1}$ ---> A1

Capacity limit $\sqrt{+999'999'999}$.
The result is output only in integers (e.g. $\sqrt{3}$ = 1). The remainder is lost.

Higher accuracy is obtained by multiplying the radical by a multiple of 1ØØ (e.g. $\sqrt{3'ØØ'ØØ}$ = 173).

If the square root of a negative number is extracted, "carry" is set = 1 and ACCU = Ø. A1 contains the unchanged negative radical.

CMP Compare the numerical values A1 and AØ

CMP: Compare A1 with AØ (A1 - AØ)

Mnemo code	Numerical code	Operand	Operation
CMP	25	Element address En = 2...999(i)	A1 > AØ ---> En =1, carry=Ø A1 = AØ ---> En-1=1, carry=Ø A1 < AØ ---> En-2=1, carry=1

(i) = indexable

The result of the comparison operation is output directly onto 3 element addresses (flags or outputs), thus supplying the respective information to the process. The operation does not change the registers AØ and A1.

The comparison takes into account the preceding sign up to ±999'999'999 or ±($1Ø^{9}$-1) respectively. E.g. +3 is greater than -5ØØ.

I 3 Instructions for processing A-registers

CLA Clear register AØ, A1 or A2

CLA: Clear A

Mnemo code	Numerical code	Operand	Description
CLA	14	Ø or 1 or 2	Clear AØ, A1 or A2

LAI Load register AØ with constant K

LAI: Load AØ immediately

Mnemo code	Numerical code	Operand	Description
LAI	15	K (Ø...2Ø47)	K = Ø...99 ---> only R4 is loaded K = 1ØØ...2Ø47 ---> entire AØ is loaded

If the constant is e.g. 57, only R4 will be loaded, RØ to R3 remain unchanged.

If the constant, however, is e.g. 225, the entire AØ will be loaded, i.e. the 7 decimals before the number 225 are filled with zeroes.

DBN, BND Conversion from decimal ---> binary or binary ---> decimal

DBN, BND: Decimal to binary, binary to decimal

Mnemo code	Numerical code	Operand	Description
DBN	16	Ø	Convert AØ decimal ---> AØ binary (capacity limit ±999'999'999)
BND	17	Ø	Convert AØ binary ---> AØ decimal (capacity R4 to R1)

All arithmetic operations are executed correctly with the numbers in BCD-representation. If, e.g. purely binary values are loaded from analog inputs, they must be converted to BCD-representation before continuing processing.
The contrary is the case with the output of data to analog outputs.

ROR Rotate registers

ROR: Rotate Registers

Mnemo code	Numerical code	Operand	Description
ROR	18	Ø	Rotate R4->R3->R2->R1->RØ->R4
		1	Rotate RØ->R1->R2->R3->R4->RØ

The function of these instructions is evident from the overview at the beginning of the following chapter.

ROA Rotate A

ROA: Rotate A

Mnemo code	Numerical code	Operand	Description
ROA	19	Ø	Rotate AØ -> A1 -> A2 -> AØ
		1	Rotate A2 -> A1 -> AØ -> A2

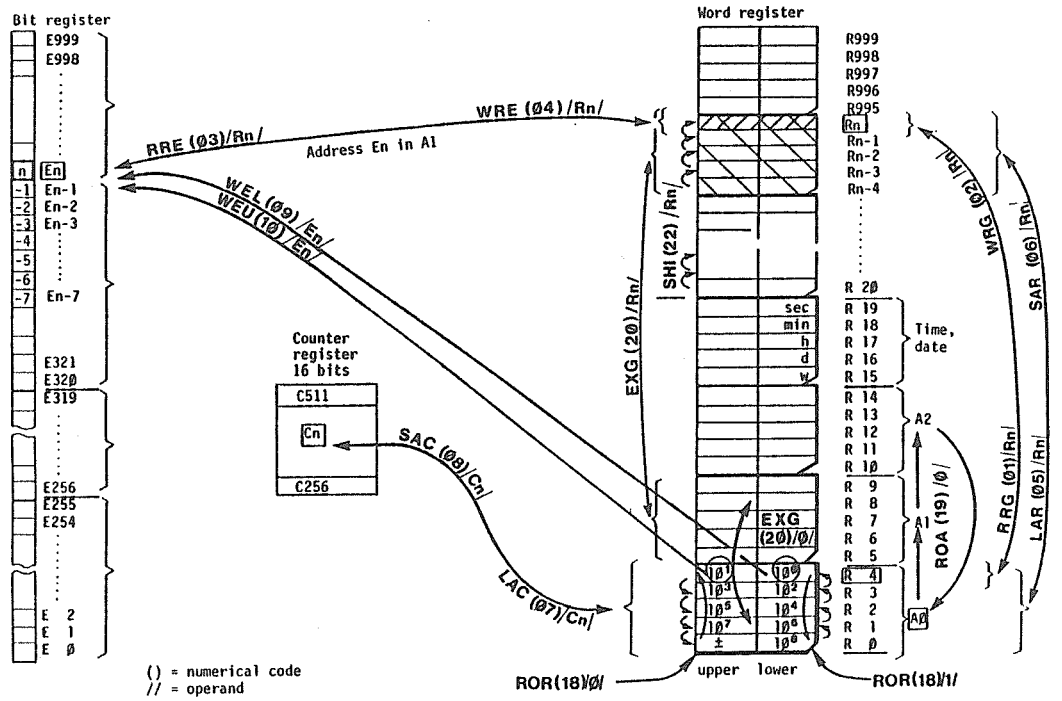
EXG Exchange A1 with AØ or with register block Rn-4...Rn

EXG: Exchange A

Mnemo code	Numerical code	Operand	Description
EXG	2Ø	Ø	Exchange A1 with AØ
		Rn from 14...999 (i)	Exchange A1 with Rn-4...Rn R9 <---> Rn R8 <---> Rn-1 R7 <---> Rn-2 R6 <---> Rn-3 R5 <---> Rn-4

(i) = indexable

I 4 Overview of the transfer instructions



Num. code	Mnemo code	Operand	Instruction	Description	Data format	

Data transfer instructions

01	RRG	Rn	Read Register	Read word Rn and store it in R4	8 bits	i
02	WRG	Rn	Write into Register	Write the word of R4 into Rn	8 bits	i
03	RRE	Rn	Read Register and write in Elements	Read word Rn and store it in the 8 elements En...En-7, addressed via A1	8 bits	i
04	WRE	Rn	Write Elements into Register	Write register Rn with elements En...En-7 addressed via A1	8 bits	i
05	LAR	Rn	Load A0 with Registers	Load register block Rn into A0	5x8 bits	i
06	SAR	Rn	Store A0 in Registers	Store A0 into register block Rn	5x8 bits	i
07	LAC	Cn	Load A0 with Counter	Load A0 with counter Cn	BCD	i
08	SAC	Cn	Store A0 in Counter	Store A0 in counter Cn	BCD	i
09	WEL	En	Write into { Lower } digit	Write { R4 (10^0) } into elements		
10	WEU	En	elements { Upper }	digit { R4 (10^1) } En...En-3	4 bits	i

Processing of the arithmetic registers

18	ROR	0, 1	Rotate Register	Rotate R4...R0, bzw. R0...R4		
19	ROA	0, 1	Rotate A	Rotate A0...A2, bzw. A2...A0		
20	EXG	0	Exchange A1 with A0	Exchange A1 with A0		
		Rn	Exchange A1 with Rn...Rn-4	Exchange A1 with Rn...Rn-4		i

(i) = indexable

The previous overview clearly explains the function of all instructions. The transfer of data is not only possible within the word register, data can also be transferred to the counter and bit register.
Various instructions with word lengths of 4, 8 or 4Ø bits are available. In general, the data will have BCD-format. Purely binary data, however, can be exchanged among all other registers, with the exception of the counter register. For arithmetic processing, however, the data to be processed must have BCD-format (see DBN or BND).

RRG, WRG Data transfer between Rn and R4

RRG: Read Register (Rn)

WRG: Write into Register (Rn)

Mnemo code	Numerical code	Operand	Description
RRG	Ø1	Rn from	Read word Rn (8 bits) and store it in R4
WRG	Ø2	Ø...999(i)	Write the word of R4 (8 bits) into Rn

The data format can be BCD or binary.

LAR, SAR Data transfer between register block Rn...Rn-4 and AØ

LAR: Load AØ with Registers

SAR: Store AØ into Registers

Mnemo code	Numerical code	Operand	Description
LAR	Ø5	Rn from	Load AØ with register block Rn...Rn-4 (5 x 8 bits)
SAR	Ø6	9...999(i)	Transfer AØ into register block Rn...Rn-4 (5 x 8 bits)

Contrary to the instructions RRG and WRG complete register blocks of 5 x 8 bits or 9 BCD-numbers including preceding sign are transferred here. The data can also have purely binary format.

As mentioned before, the name of the operand Rn implies that the whole register block Rn...Rn-4 is contained. For the sake of clearness it is recommended to divide the word register into block of 5 and to use only end addresses ending with 4 or 9 for these instructions.

EXG: Exchange A1 with register block Rn...Rn-4: see I 3

(i) = indexable

LAC, SAC Data transfer between counter register Cn and AØ

LAC: Load AØ with Counter (Cn)

SAC: Store AØ into Counter (Cn)

Mnemo code	Numerical code	Operand	Description
LAC	Ø7	Cn from	Load AØ with counter Cn
SAC	Ø8	256...511(i)	Store AØ into counter Cn

The data transferred from AØ to Cn must have BCD-format. The data stored in AØ transferred from Cn has BCD-format, too.
Observe the capacity limit of Cn = 65'535. Therefore, the preceding sign of AØ cannot be taken into account.

The counter register is basically used by the counters and timers which can be activated in the bit processor. The instruction LAC is used to read out and transfer counter and current timer values. If, however, a timer value of AØ is to be adopted, the program must be organised as follows:

```
SAC          256     Transfer AØ ---> C256 (to counter register!)
 .            .
 .            .
EWP           Ø
 .            .
 .            .
STR (14)     256 |   Transmission of the value and simultaneous
     31       256 |   activation as timer
```

WEL, WEU Transfer of 1 BCD-digit to elements (En)

WEL: Write Elements with lower Digit

WEU: Write Elements with upper Digit

Mnemo code	Numerical code	Operand	Description	
WEL	Ø9	En from	Write into elements	R4(1Ø⁰)
WEU	1Ø	3...999(i)	En...En-3 the digit of	R4(1Ø¹)

This instruction allows the output of numbers digit by digit (4 bits) directly to flags or outputs, which are required e.g. for multiplex output to external displays.

The 4 digits can have both BCD-format and any other bit pattern.

(i) = indexable

RRE, WRE Transfer of register words Rn to elements En

RRE: Read Register and write in Elements

WRE: Write Register with Elements

Mnemo code	Numerical code	Operand	Description
RRE	Ø3	Rn from Ø...999(i)	Read word Rn (8 bits) and store it in the elements En...En-7. En is addressed via A1.
WRE	Ø4	Rn from Ø...999(i)	Write the contents of the elements En...En-7 (8 bits) in register word Rn. En is addressed via A1.

Before using this instruction the element address En must be in A1. Only the numbers of the registers R9 and R8 are relevant for this address.

I 5 Counting and skip instructions

INR, DER Decrementing or incrementing the register value by one

INR: Increment Register by 1

DER: Decrement Register by 1

Mnemo code	Numerical code	Operand	Description
INR	11	Rn from Ø...999(i)	Increment BCD-value Rn by 1 and set "carry", if result > 99
DER	12		Decrement BCD-value Rn by 1 and set "carry" if result < Ø

Together with SNC these instructions serve to form BCD-counters of any size within the register Rn.

(i) = indexable

SNC Conditional skip of the next instruction

SNC: Skip if no carry

Mnemo code	Numerical code	Operand	Description
SNC	13	Ø	Skip the next instruction if "carry" = Ø
		En (i)	Skip the next instruction if En = L

By using this instruction together with INR and DER, counting cascades can be constructed.

SEW Conditional skip of several instructions until EWP or NOP 1248

SEW: Skip to EWP or to NOP 1248

Mnemo code	Numerical code	Operand	Description
SEW	24	Ø	Skip to EWP or NOP 1248 if "carry" = Ø
		En (i)	Skip to EWP or NOP 1248 if En = L

When "carry" = 1 or En = "H" the instruction SEW has no effect on program execution. However, if "carry" = Ø or En = "L", the instructions following SEW ignored. EWP causes switching over to the bit processor while working in the word processor is continued due to NOP 1248.

(i) = indexable

I 6 Various instructions

CLK Assignment of a timing pulse source

CLK: Clock Source

Mnemo code	Numerical code	Operand	Description
CLK	21	En	Assignment of a timing pulse source to the clock register
		Ø...999	Acts on sec-register
		1ØØØ...1999*	Acts on min-register
		2ØØØ	Reassignment to internal clock

*) If 1ØØØ is added to the element address (do not confuse with indexing), the register of the minutes is activated.

The instruction CLK is used to supply the clock by an external pulse source (e.g. main clock) via a PLC-input. Depending on the operand the pulses act on the sec- or min-register.
The pulse itself must have a pulse or pause length of 125ms (max. 4Hz).
If the assignment via CLK is missing, stepped addressing of the clock from the internal "clock" or from module R27 is effected automatically. The operand 2ØØØ is used to re-establish this state.

SHI Shift register

SHI: Shift Register

Mnemo code	Numerical code	Operand	Description
SHI	22	Rn 21...999(i)	Shift register words (8 bits) by 1 address upwards to Rn starting with R2Ø

The overview at the beginning of this chapter contains the function of the instruction. It is used to form one-bit or multibit shift registers.

After shifting, the information on R2Ø is not changed and rests on R2Ø, while the information on Rn is lost.

(i) = indexable

TXT Start of text output

TXT: Text

Mnemo code	Numerical code	Operand	Description
TXT	23	TXn from Ø...818 (i)	Start of text output

The text with the text number TXn is output via the serial data interface until the character NUL (in the text) is found.
Details see K 5.

NOP No operation

NOP: No Operation

Mnemo code	Numerical code	Operand	Description
NOP	ØØ	Ø	No operation

Like in the bit processor instruction set NOP Ø results in empty lines in the program part of the word processor. These are executed, but they have no effect on the program.

EWP End of operation in the word processor

EWP: End Word Processor

Mnemo code	Numerical code	Operand	Description
EWP	31	Ø	End of operation in the word processor

Each program part in the word processor starts with the bit instruction NOP 1248 and ends with EWP Ø. At these points shifting to the relevant instruction set is effected.

(i) = indexable

Notes:

PART K TEXT OUTPUT AND COMMUNICATION

PART K TEXT OUTPUT AND COMMUNICATION

K 1 The serial data interface

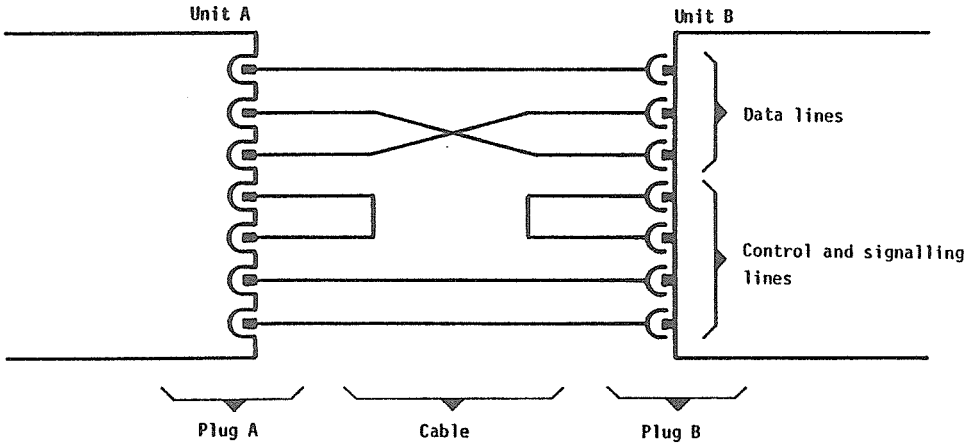
The serial data interface of the PCA can be used for the following functions:

- Output of texts from the text memory (TEXT mode)
- Input of texts by means of a peripheral unit into the text memory (EDITOR mode)
- Exchange of single ASCII characters with another system (mode C)
- Exchange of numerical data according to DIN 66Ø19 (mode N)
- Direct access to PLC data such as elements, counters, registers, program lines according to DIN 66Ø19 (mode P)

K 1.1 What is a serial data interface?

If you are familiar with the expressions ASCII code, baud rate, parity bit, RS232c and 2ØmA current loop, you may proceed to section K 2. If you are not, the following information should be read with special care.

Nowadays, most peripheral units and computers have interfaces corresponding to well-known standards such as EIA RS232c, CCITT V.24, DIN 66Ø2Ø or 66259; problems may nevertheless arise during the connection of these units. The reason for this is that all these "standards" leave considerable lattitude for individually required specifications.



As the figure shows, the following questions arise just as regards hardware:

- What type of connector (number of pins, male/female) is used?
- Connector pin assignment?
- Are control and signalling lines required in addition to data lines?
- Cable dimensioning?

Among others, the following questions regarding the characteristics of the devices have to be answered:

- Baud rate?
- No. of stop bits?
- Parity?
- Full or half-duplex?
- Which is the active unit (i.e. supplies the current of 2ØmA)?

etc.

K 1.1.1 What is the ASCII code?

In computer science informatics only "Ø" and "1" are used as electric signals. If numbers or other characters are to be presented electrically, it is necessary to use a combination of several bits. With 7 bits 2^7 = 128 characters or numbers can be represented. The ASCII code arranges these 128 characters in a table and subdivides them into so-called "cases". In the "control case" there are 32 control commands such as "CR" = Carriage Return, "LF" = Line Feed, etc.

The "symbols case" contains 32 graphic characters and the decimal numbers. The "upper case" contains the capital letters and the "lower case" the small letters.

The table shows the characters decimally numbered from Ø to 127. These decimal numbers are used for text input via the PØ5 programming unit. They can also be represented as hexadecimal numbers, from Ø to 7F.

The English meaning of the "control case" is listed as follows:

Controls:

(Ø)	NUL : Null (blank)	(8)	BS : Back Space
(1)	SOH : Start of Heading	(9)	HT : Horizontal Tabulation
(2)	STX : Start of Text	(1Ø)	LF : Line Feed
(3)	ETX : End of Text	(11)	VT : Vertical Tabulation
(4)	EOT : End of Transmission	(12)	FF : Form Feed
(5)	ENQ : Enquiry	(13)	CR : Carriage Return
(6)	ACK : Acknowledgement	(14)	SO : Shift-Out
(7)	BEL : Bell	(15)	SI : Shift-In
(16)	DLE : Data Link Escape	(24)	CAN : Cancel
(17)	DC1 : Device Control 1	(25)	EM : End of Medium
(18)	DC2 : Device Control 2	(26)	SUB : Substitute
(19)	DC3 : Device Control 3	(27)	ESC : Escape
(2Ø)	DC4 : Device Control 4	(28)	FS : File Separator
(21)	NAK : Negative Acknowledgement	(29)	GS : Group Separator
(22)	SYN : Synchronous Idle	(3Ø)	RS : Record Separator
(23)	ETB : End of Transmission Block	(31)	US : Unit Separator

ASCII CODE CHART

B7 B6 B5 / BITS / B4 B3 B2 B1	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
	CONTROL		SYMBOLS		UPPERCASE		LOWERCASE	
0 0 0 0	NUL 0	DLE 16	SP 32	0 48	@ 64	P 80	\ 96	p 112
0 0 0 1	SOH 1	DC1 17	! 33	1 49	A 65	Q 81	a 97	q 113
0 0 1 0	STX 2	DC2 18	" 34	2 50	B 66	R 82	b 98	r 114
0 0 1 1	ETX 3	DC3 19	# 35	3 51	C 67	S 83	c 99	s 115
0 1 0 0	EOT 4	DC4 20	$ 36	4 52	D 68	T 84	d 100	t 116
0 1 0 1	ENQ 5	NAK 21	% 37	5 53	E 69	U 85	e 101	u 117
0 1 1 0	ACK 6	SYN 22	& 38	6 54	F 70	V 86	f 102	v 118
0 1 1 1	BEL 7	ETB 23	' 39	7 55	G 71	W 87	g 103	w 119
1 0 0 0	BS 8	CAN 24	(40	8 56	H 72	X 88	h 104	x 120
1 0 0 1	HT 9	EM 25	) 41	9 57	I 73	Y 89	i 105	y 121
1 0 1 0	LF 10	SUB 26	* 42	: 58	J 74	Z 90	j 106	z 122
1 0 1 1	VT 11	ESC 27	+ 43	; 59	K 75	[91	k 107	{ 123
1 1 0 0	FF 12	FS 28	, 44	< 60	L 76	\ 92	l 108	¦ 124
1 1 0 1	CR 13	GS 29	- 45	= 61	M 77	] 93	m 109	} 125
1 1 1 0	SO 14	RS 30	. 46	> 62	N 78	^ 94	n 110	~ 126
1 1 1 1	SI 15	US 31	/ 47	? 63	O 79	_ 95	o 111	RUBOUT (DEL) 127

The two characters of the "character case":

SP (no. 32) = Space
DEL (no. 127) = Delete or rubout (erase)

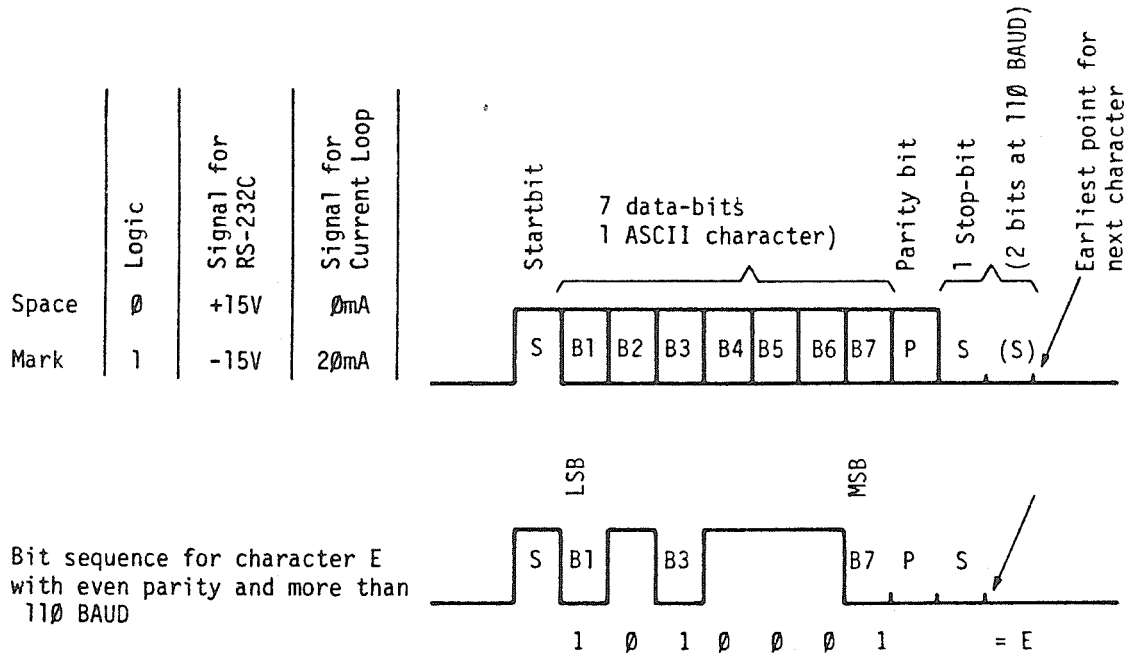
---

*) In most peripheral units the "control case" characters are obtained by simultaneously pressing the CTRL -key and the corresponding "upper case" character. For example, "FF" = Form Feed (no. 12) is obtained by simultaneously pressing CTRL and L ---> CTRL/L.

K 1.1.2 Serial data transmission, baud rate, parity

As is shown in the ASCII table, the binary value 1ØØØ1Ø1 (decimal value 69D) corresponds to the character E. The binary value can be transmitted as a serial signal via an electric line. The format and electric level for transmission are defined in RS232c. In order to determine the beginning and the end of a character, a start bit (as space) is transmitted at the beginning and a stop bit (as a mark) at the end. Two stop bits are only required for 11Ø bauds.

The "standard" resp. use of the 2ØmA current loop corresponds to that of RS232c.



The transmission speed resp. the number of bits transmitted per second is designated as baud rate. The following baud rates are standard:

11Ø, 15Ø, 3ØØ, 6ØØ, 12ØØ, 24ØØ*, 48ØØ*, 96ØØ* bauds.

If 11 bits are used for the entire bit format, max. 1Ø ASCII-characters are transmitted per second at a transmission speed of 11Ø bauds, at 96ØØ bauds about 1ØØØ characters per second.

To check the transmitted characters a parity bit can be added to the 7 bits. This parity bit is usually selected in such a way that the number of ones within the 8-bit sequence is even in the case of a synchronous transmission (see figure). If the receiver detects an odd number, a transmission error has occured.

---

*) In order to obtain higher baud rates the user program needs to be organised correspondingly (checking with the PCA ASSEMBLER program "RTA" is recommended).

K 2 The 2ØmA current loop
(also called TTY, line current or current loop interface)

The electric signal for representation of the logic "Ø" or "1" may be a voltage or a current. According to the standard RS232c or V.24, a ± voltage of 15V is necessary. Although this interface is the most common one, it has major disadvantages in industrial applications:

- No opto-isolation is possible between transmitter and receiver.
- Because of susceptibility to interference, connections of more than 15m are not allowed.

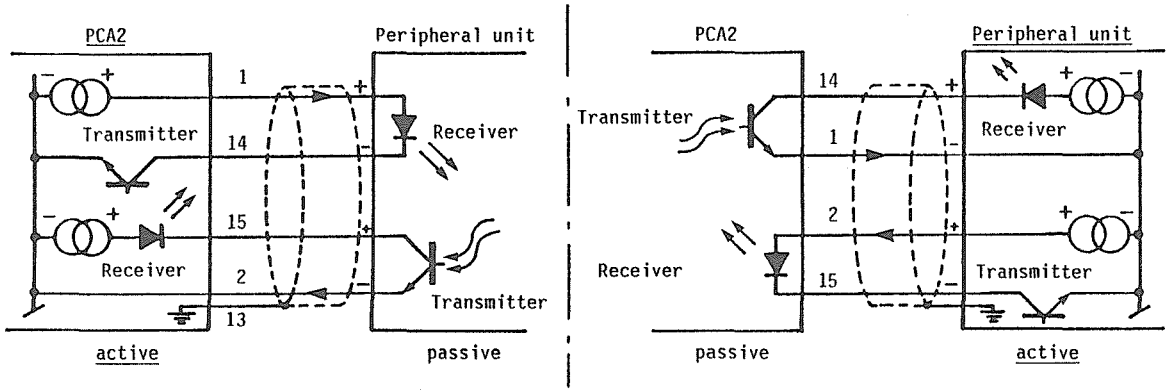
The current loop allows on the other hand:

- The possibility of transmitting 2ØmA directly to optocouplers, thus providing opto-isolation between transmitter and receiver.
- Interference-safe transmission for line lengths up to 1ØØØm.
- Transmission speed is limited to max. 96ØØ bauds (this is of minor importance in the case of PLC).

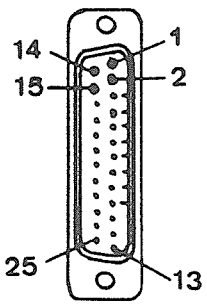
For a current loop interface it must always be determined which side provides the current. This so-called active side is galvanically connected to the line, with the passive side is separated via optocouplers.

K 2.1 PCA2.M22 and M32

The interface can be made "active" or "passive" by reinserting the multiple connector (see PCA2 Hardware), thus adapting it to the peripheral unit.
The connecting cable PCA2-peripheral equipment must be designed according to the following figure.



The interface connector:



Designation:
DATA LINES, male connector
25-pole D-Sub connector with the possibility of screw locking.

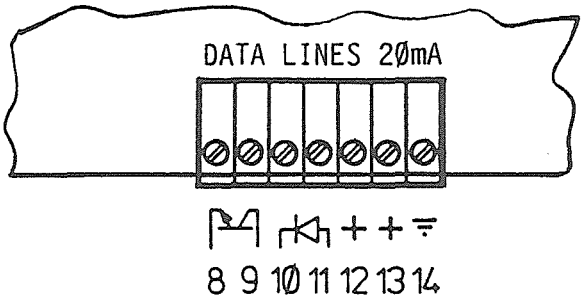
K 2.2 PCA14 and PCAØ2

These two types allow connecting the serial interface (2ØmA current loop) via pluggable screw terminals. The PCA can be made active or passive by corresponding wiring. It is also possible to make only the transmitter active, the receiver, however, passive.

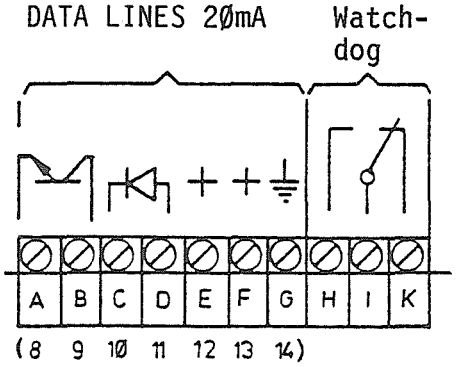
Except for the different terminal designation, PCA14 and PCAØ2 are completely compatible concerning hardware as well as software.

Terminal assignment

PCA14

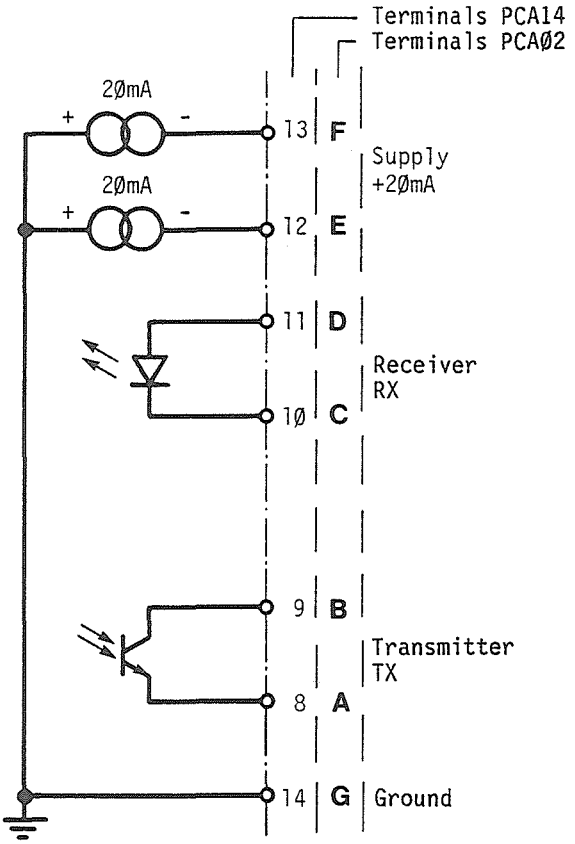


PCAØ2



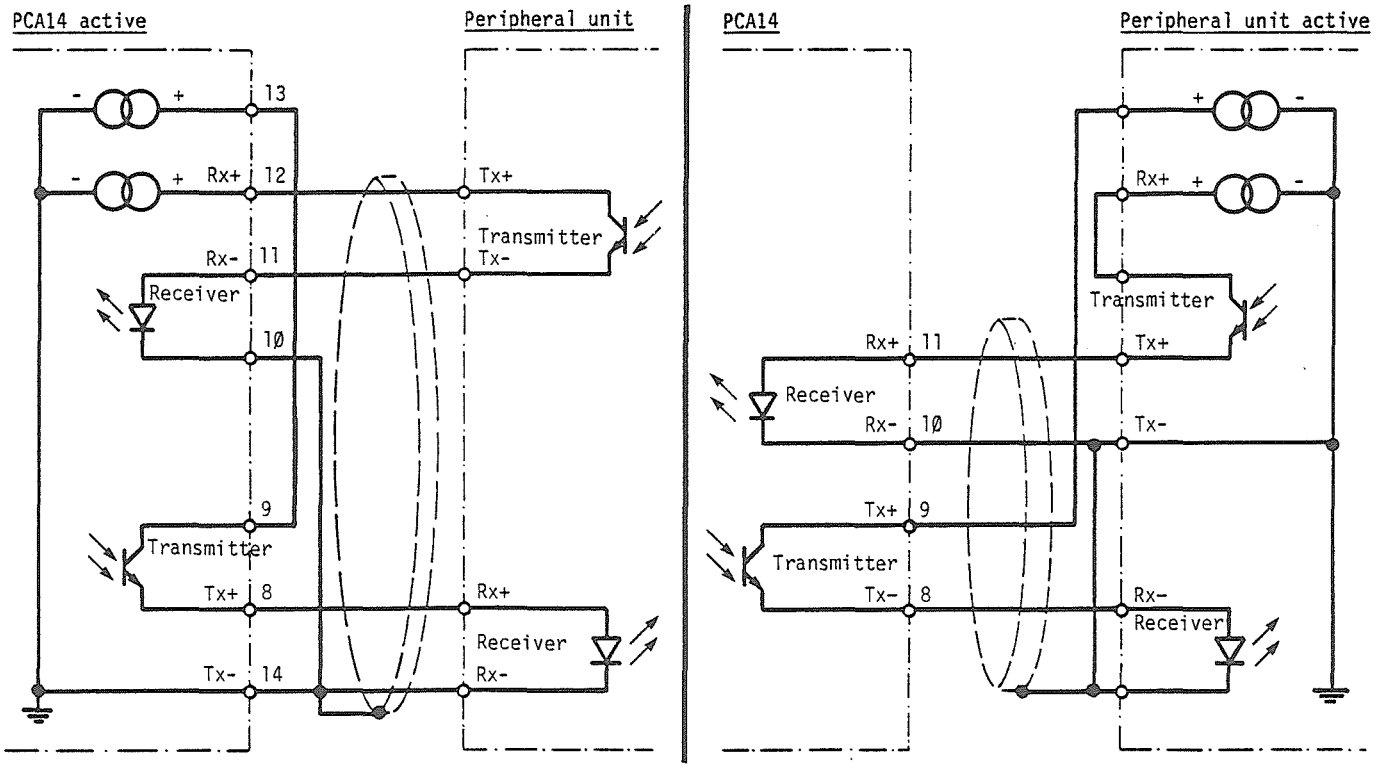
corresponding terminals on PCA14

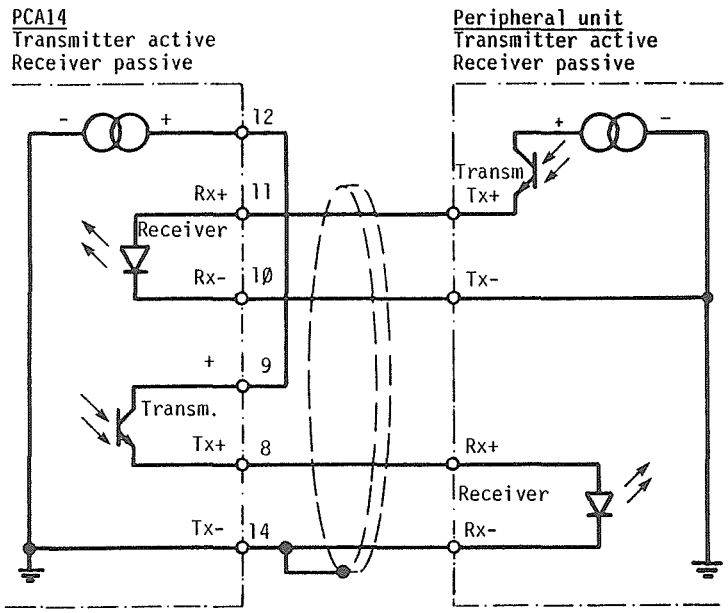
Connection diagram



Connection diagrams (PCA14 and PCAØ2)

The following diagrams only refer to the PCA14. Due to the PCAØ2 terminal configuration on the previous page, it becomes evident how the PCAØ2 can also be connected in all three ways.





K 3 Function of the "error"-lamp of the operating panel

The "error" lamp will light up, if:

a) there is a wrong character, i.e. parity wrong, framing or overrun

b) the telegram is wrong, i.e.
- STX is not the first character
- the telegram contains either forbidden or too many characters
- BCC is wrong

c) the circuit is interrupted.

The "error" lamp goes out on each cycle of seconds (or every 1/10ms, if the time base 1/100s has been selected).

K 4 Assignment of the interface on the PCA CPU

It is evident from the instruction that not only the correct connector cable and connectors must be selected for a serial exchange of data, also the various transmission parameters of the two communicating units must be determined and match. In the case of the PCA the transmission parameters are determined by the PAS 1ØØ instruction.

PAS 1ØØ heads a 1Ø-line instruction, the function of the various lines being as follows:

```
Line 1:  PAS   (29)      1ØØ   ;  Activation of the interface
     2:        ØØ        xxxx  ;  Baud rate, parity and stop bits
     3:        Øx        xxx   ;  Text busy flag (TXB)
     4:        n         xxx   ;  Receive Buffer busy Flag (RBY)
     5:        n         xxx   ;  Transmit Buffer busy Flag (TBY)
     6:        Øx          d   ;  Block Check Character
     7:        ØØ          Ø   }
     8:        ØØ          Ø   }  NUL
     9:        ØØ          Ø   }
    1Ø:        ØØ          Ø   }
```

PAS 1ØØ is normally assigned at the beginning of a program and is consequently processed only once.
If an assigned mode (C, N, P) must be changed, corresponding reassignment can be effected in the user program.

Program line 1: PAS 1ØØ

Line 2: Baud rate, parity and stop bits

The operand of the 2nd line contains the sum of the following parameter values. The code is always ØØ.

Parameter		Constant to be added
Baud rate:	11Ø bauds	Ø
	15Ø bauds	1
	3ØØ bauds	2
	6ØØ bauds	3
	12ØØ bauds	4
	24ØØ bauds*	5
	48ØØ bauds*	6
	96ØØ bauds*	7
No. of data bits per character:	7	Ø
	8	32
Parity generation and check:	passive	Ø
	active	128
If parity active:	odd	Ø
	even	256
No. of stop bits:	1 bit	512
	1 1/2 bits	1Ø24
	2 bits	1536

*) In order to obtain high baud rates a program structure with short periods between two PPs is required. Checking with the PCA ASSEMBLER "RTA" is recommended.

Example:	12ØØ bauds	4	
	7 data bits	Ø	
	Parity, active	128	
	Parity, even	256	
	1 stop bit	512	
	Total	9ØØ	≙ operand of 2nd line

Line 3: Text Busy Flag (TXB)

ØØ in the code applies to the general application of the data interface.
Ø1 in the code activates the editor for the text input via the serial interface (see K 5.3).

The address of a flag or output, which is "H" as long as text is output, must be in the operand. On termination of the text output, this element (busy signal) is "L" again.

Thus, it is possible to finish the output of a text before starting another one.

If a text is to take priority (e.g. an alarm message), the current text can be interrupted at any time, without interrogation of the "busy" element to provide space for the priority text.

Line 4 and 5: Receive/transmit buffer

The size of this buffer is determined in the code, the address of the busy flag is in the operand. For further details refer to chapter K 6.

Line 6: Block check character (BCC)

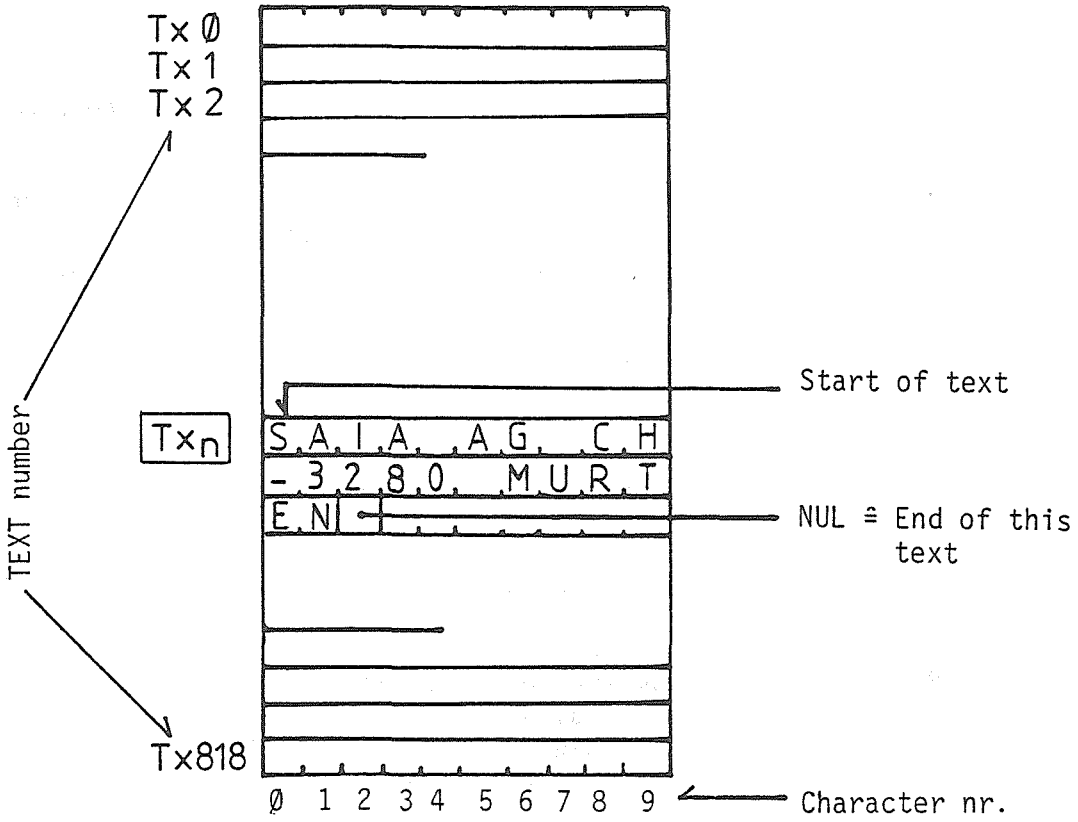
Transmission reliability for transmissions according to DIN 66Ø19 may be increased by determining a block check character. For details refer to chapter K 6.3.

K 5 Text input/output

K 5.1 Organisation of the text memory

According to the hardware description, the text memory is part of the user memory. 4K user memory correspond to 818 texts of 1Ø characters, i.e. 8K charakter.

Each text beginning can be addressed between TxØ...Tx818. The first character has the character no. Ø. The end of the text must always be the character Ø (ASCII symbol NUL). Longer texts may consist of as many text numbers as desired.



K 5.2 Output of a text

All level-2 processors use the PAS 23 instruction. On level 3 (PCA231 and 232) the instruction TXT is available in the word processor.
The effect is exactly the same. Both instructions are always executed independently of the ACCU-state and do not influence the ACCU.

Both the bit processor instruction

PAS	(29)	23	
	ØØ	375	(i)

and the word processor instruction (only PCA231/232)

TXT	(23)	375	(i)

output the text starting with text no. 375 to the "NUL" symbol. The output is effected via the serial interface (2ØmA current loop). To ensure reliable transmission, the appropriate parameters must be determined and programmed before:

- The active/passive current loop is determined by the hardware via the selectable connector on the CPU PC-board (see Hardware manual).
- The transmission parameters such as BAUD rate, parity, etc. must be programmed in the 2nd line of the PAS 1ØØ instruction in the user program (see previous section).
- By means of the PAS 1ØØ instruction an element (flag or output) is determined in the 3rd line; this remains "H" as long as text is being output; the "busy" element TXB is "L" again on termination of the text output.
- In the 4th and 5th line of the PAS 1ØØ instruction, elements must be addressed which are usually always "L".

Example:

Suppose text "SAIA AG CH-328Ø MURTEN" is at text no. 375. It must be output to a terminal by closing contact I7.

The corresponding program is the following:

1	PAS	(29)	1ØØ	Interface assignment
2		ØØ	9Ø2	Parameter for terminal*
3		ØØ	32	Busy display (TXB)
4		ØØ	254	} the higher address of 2 elements which
5		ØØ	254	} permanently remains "L" (254 and 253)
6		ØØ	Ø	}
7		ØØ	Ø	}
8		ØØ	Ø	} NUL
9		ØØ	Ø	}
1Ø		ØØ	Ø	}
11	STH	(Ø1)	7	Wait for contact 7
12	DYN	(Ø9)	4ØØ	
13	JIZ	(22)	11	
14	WIH	(25)	32	Wait for text output
15	PAS	(29)	23	
16		ØØ	375	Text output as from no. 375
17	JMP	(2Ø)	11	

Text output in the the bit processor

15	NOP	(ØØ)	W	1248	
16	TXT	(23)	W	375	
17	EWP	(31)	W	Ø	
18	JMP	2Ø		11	

Text output in the word processor

*) 48ØØ bauds, 7 bits, parity active and even, 1 stop bit
(i) = indexable

K 5.3 Input of texts

Texts are introduced into the RAM text memory in the user program which is located on the lower plug-in location.

On input and checking of the texts, the RAM contents can be copied in the same way user programs are copied.

If a certain text section is to be copied, the text numbers must be converted into step addresses as follows:

Starting text no. x5 ≙ starting step address
Destination text no. x5 (+4) ≙ destination step address

Example:

Starting text no. 15Ø ≙ starting step address 75Ø (15Ø x 5)
Destination text no. 2ØØ ≙ destination step address 1ØØ4 (2ØØ x 5 + 4)

Special characters

In addition to normal texts, the text memories can also contain special characters, by which registers may be read out and, in combination with clear text, enhanced to provide communication logs or graphic representations.

Except for the character "NUL" (Ø)*, which marks the end of the text, all symbols of the ASCII chart from 1...127 (1...7F hex) can be used. They are generally transmitted without being modified except for the special character $, followed by 1 to 4 characters.

If 8 bits per character are used, the ASCII-characters 128...255 may be output, the effect of which depends on the connected terminal.

---

*) The ASCII symbol "NUL" can be generated on most terminals by CTRL/@.

Meaning of the special characters

Character sequence in text memory		Character sequence in the form it is output
Alphanum. input	Input with PØ5 (dec. no.)	
$$	36, 36	$
$C Cn *	36, 67,...	Contents of counter register Cn, 5 decimal numbers without suppression of preceding zeroes.
$c Cn *	36, 99,...	Contents of counter register Cn, 5 decimal numbers with suppression of preceding zeroes (spaces).
$A Rn *	36, 65,...	Contents of register block Rn, as 1Ø decimal numbers without suppression of preceding zeroes **
$a Rn *	36, 97,... ***)	Contents of register block Rn, 9 decimal numbers (with preceding sign, if negative) with suppression of preceding zeroes (spaces) **
$R Rn *	36, 82,...	Contents of the register word Rn, 2 decimal numbers **
$r Rny	36, 114,...	Contents of the LOWER resp. UPPER part of the register word, 1 decimal number, y = Ø ---> LOWER, y = 1 ---> UPPER **
$E En *	36, 69,...	States of the 8 elements En...En-7 8 characters 1 or Ø
$e En *	36, 1Ø1,...	Output of ASCII characters which is formed by the 7 or 8 elements En...En-7
$S En*	36, 83,...	If element En = "H", character "-" is output If element En = "L", character "+" (PCA14) or "space" (PCA232) is output
		Contents of date-time:
$T	36, 84	: 22 : Ø6 / 13 : 3Ø : 42 Week Day Hour Min. Sec. (22.) (SAT)
$H	36, 72	83 - Ø6 - Ø4 / 13 : 3Ø : 42 Year Month Day Hour Min. Sec.
$D	36, 68	83 - Ø6 - Ø4 Year Month Day

*) see next page
**) see next page
***) only possible with PCA231/232

$ xxx [1]	36, _,...	Repeated output of the character immediately following $, as many times as determined in xxx: - xxx ≤ 127 Repetition according to xxx - xxx = 256...319 or 511 Repetition corresponding to the content of the respective counter (max. 127) → Character to be repeated. All ASCII symbols (incl. NUL) may be used - except for the following which initiate functions: $, C, c, D, E, e, H, L, T, U, A, a, R, r
$L xxx [2]	36, 76,...	Jump to text no. xxx and storage of the jump address (text subroutine analog JMS)
$U	36, 85	Jump back to starting text (analog to RET, whereas only 1 level is allowed). [3]

Part L
Examples
1,2,3,4 and 11

[1] Three digits must always be entered for Cn, Rn, En or xxx, eg. Ø27.
[2] If words of 4 bits in BCD-representation reach values between 1Ø and 15, the following characters are output:
1Ø ---> : 11 ---> ; 12 ---> < 13 ---> = 14 ---> > 15 ---> ?
[3] 3 subroutine levels are admissible from certain firmware versions onward: PCAØ2: V6.132; PCA14: V6.Ø34; PCA222: V6.23Ø.

Text input:

There are two ways of entering texts into the text memory via the hardware:

a) by using the programming units P1Ø/PØ5 or P21 connected via the 25-pin connector "PGU".

b) by using a peripheral unit with current loop interface which is connected to the connector "DATA LINES".

a) Text input via programming unit PCA2.P1Ø/PØ5

The programming unit is connected by means of the PGU-connector. Set operating mode selector switch to TEXT. The input is effected similiarly to PROG.

e.g. The text "SAIA AG CH-328Ø MURTEN" must be entered using text start number 375.

	Text number / Char. no.		Number of the ASCII-code in decimal form		Text
[A]	375Ø *	[E]	83	[E]	S
	(3751)**		65	[E]	A
	(3752)		73	[E]	I
	(3753)		65	[E]	A
	(3754)		32	[E]	
	(3755)		65	[E]	A
	(3756)		71	[E]	G
	(3757)		32	[E]	
	.		.	.	.
	.		.	.	.
	.		.	.	.
	(3776)		13	[E]	"CR"
	(3777)		1Ø	[E]	"LF"
	(3778)		Ø	[+]	"NUL" = End of text

*) Please note that the character number is entered, i.e. for text start 375 ---> 375Ø.

**) The following character numbers do not have to be input. They are displayed automatically.

b) Text input via a peripheral unit with current-loop interface

Once cable dimensioning has been completed and the transmission parameters have been determined, the peripheral is connected to the DATA LINES connector. The programming unit PØ5 must be connected, the PCA must neither be on RUN nor on BREAK mode.

Software conditions:

The user must make sure that the CPU has executed the 1Ø-line PAS 1ØØ instruction, the 3rd line of which contains the operation code Ø1. The CPU then has all received characters processed by its text editor (see the following application example).

Meaning of special characters of the text editor

The text editor may be in two different operating modes:

- text display mode
- text input mode

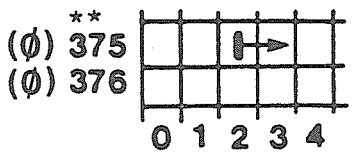
- Text display mode

With the aid of some special instructions it is possible, in this mode, to display the entered text on a screen or to jump to various positions in a text by means of the "pointer".

Txn [CR] **

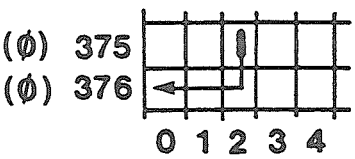
Input of a text number Txn followed by "CR" (only the last 3 digits are significant). The "pointer" is thus set to the beginning of text no. Txn and the corresponding text number with contents (1Ø characters) is displayed.

[>]



The pointer is shifted by one position to the right in the character number. By keeping the key depressed, this process is repeated several times*. The stored text is then continuously output without any lines in between.

[+]



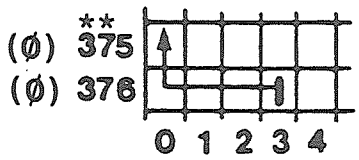
The pointer is skipped to the beginning of the next higher text address. Keeping the key depressed causes repeat function*. The stored text is then continuously output without any lines in between.

---

*) The repeat function depends on the peripheral unit.

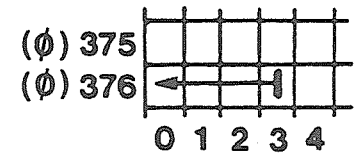
**) The PCA232 requires a 4-digit input, which means that for text 375 --> Ø375 must be entered.

[-]



The pointer is skipped upwards to the beginning of the next lower address. Keeping the key depressed causes repeat function*. The text is then output in reverse direction and displayed with 1 line in between each.

[L]



The pointer is shifted to the beginning of the current text number with its contents being output.

[CTRL/T] Change to text input mode.

- Text input mode

The text input mode is activated via CTRL/T. After a text start number has been selected it is possible to enter the current text in this mode by means of the typewriter keyboard.

[^] If this character is entered before "CR", "CR" is adopted by the text as carriage return instruction, no change to text display mode.

[BS] The PCA232 allows correcting incorrect inputs with "BS" (Back Space), i.e. the character in the cursor position is deleted.

[CR] or [CTRL/M] The characters "CR" and CTRL/M are identical and cause changing to the text display mode.

Please note: On termination of the editor function, you absolutely have to exit this mode by depressing "CR" before any other operation is started.

Example: Input of a text via peripheral unit starting with text no. 3Ø.

The assignments of the parallel programs and PAS 1ØØ with the operational assignment of the data interface will normally be at the beginning of the program.

A second block with PAS 1ØØ which is used for the text input, should be programmed at the end of the program memory, e.g. as from step 4ØØ1 followed by the 2-line instruction PAS (29) 23
ØØ 3Ø.

Then in operating mode STEP:

- jump to address 4ØØ1 to activate the text editor.
- display previously entered text for checking.

---

*) The repeat function depends on the peripheral unit.
**) The PCA232 requires a 4-digit input, which means that for text 375 ---> Ø375 must be entered.

Activation of the data interface for text input via editor:

```
4ØØ1  PAS  (29)    1ØØ  ;  Parameter for 48ØØ bauds, 7 bits, parity even,
4ØØ2       ØØ      9Ø2     1 stop bit
4ØØ3       (Ø1)     63  ;  (Ø1) for text editor; 63 for Text Busy (TXB)
4ØØ4       ØØ      254     the higher address of 2 elements which
4ØØ5       ØØ      254     permanently remains "L" (254 and 253)
4ØØ6       ØØ        Ø
 .          .        .
 .          .        .     NUL
 .          .        .
4Ø1Ø       ØØ        Ø
4Ø11  PAS  (29)     23
4Ø12       ØØ       3Ø     Output text no. 3Ø
4Ø13  JMP  (2Ø)      Ø
4Ø14       ØØ    4Ø11
```

The following text is to be entered as text no. 3Ø:

```
SAIA-PLC
CROWNED WITH PCA232
```

Procedure:

1) Connect PØ5 programming unit to PROG UNIT and set mode selector switch to "STEP".

2) Jump to the start of the assignment by depressing [A] 4ØØ1 [+] on the PØ5 programming unit.

3) Depress [+] for execution of the assignment to step address 4Ø11.

4) Enter Ø3Ø "CR" via the peripheral unit.

Display on the screen (key inputs are underlined)	Description
Ø3Ø "CR" **	With a non-cleared text memory the displayed text is randomized. A point in front of a character means that this character originates from the visible part of the ASCII table, (32...127). A control instruction of the "control case" preceded by an arrow or a ^.
Ø3Ø:.A.B. . . . ^@^@^@^@	
Ø3Ø:	

**) The PCA232 requires a 4-digit input for the text number, i.e. the last 4 digits entered are significant.

Display on the screen
(key inputs are underlined)

CTRL/T ← 5) To get into the text input mode, depress keys [CTRL] and [T] (CTRL/T) simultaneously.

6) Then enter 6 spaces using the special character [$] , followed by the SAIA°PLC logo.

7) Note that as soon as 1Ø characters have been entered the display automatically proceeds to the following text number.

```
030:      .$._.0.0.6.S.A.I.A.-
031:      .P.L.C^M^J.C.R.O.W.N
032:      .E.D._.W.I.T.H._.P.C
033:      .A.2.3.2^M^J^@^@^@
```

8) To enter "CR" without changing the text mode, [^] must be depressed first. The CPU answers by ^ M (CTRL/M) in both cases.

9) To enter "LF" = Line Feed, [CTRL/J] is depressed which corresponds to "LF". ^ J appears.

1Ø) The PCA232 allows correcting incorrect inputs in text input mode with key "BS" (Back Space). I.e. the character in the cursor position is deleted by "BS".

11) NUL is entered to end text input. This is possible via [CTRL/@] (may differ from one peripheral unit to another).

12) To exit the editor mode enter ["CR"] .

13) If the entered characters are to be displayed as clear text now, depress the [+] key on the PØ5 unit several times. The program loop 4Ø11...4Ø14 is then executed, text output is activated via instruction PAS 23.

SAIA°PLC
CROWNED WITH PCA232

Text editor in short

a) Programming the data interface as editor

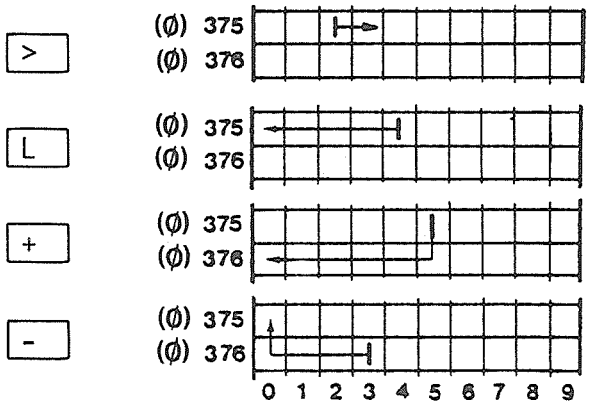
```
4001  PAS  (29)    100
4002       00      902     Transmission parameter
4003       (01)     63     TXB          (01) ---> activate editor
4004       00      254   } the higher address of 2 elements which
4005       00      254   } permanently remains "L" (254 and 253)
4006       00        0   }
  .        .         .   } NUL
  .        .         .   }
4010       00        0   }
4011  PAS  (29)     23     Text output
4012       00      Txn     in STEP mode
4013  JMP  (20)      0
4014       00     4011
```

b) Activation of the editor

In STEP mode, jump to step address 4001 via programming unit P05 and execute to step address 4011.

c) Display mode (display of the text memory)

Txn [CR] Text line with number Txn is displayed



d) Text input mode

[CTRL/T] Text input is effected as from character number Txn, which was preselected in the display mode with the "pointer".

[BS] Deletion of an entered character (only PCA232)

[^] [CR] Is accepted into the text memory as "CR".

[CR] Change to display mode.

e) Text output

In the STEP mode, depress the [+] -key of the programming unit P05 twice. The text is output every time for checking purposes.

K 5.4 Text input/output using 8 bits

The number of data bits can be 7 (according to the ASCII-code Ø...127) or 8 bits (Ø...255), i.e. operation is possible either with 7 or 8 data bits per character. This value is preselected in the 2nd line of the instruction PAS 1ØØ. The meaning of the additional characters 128...255 depends on the peripherals connected (computer, graphic terminal) and varies from device to device.

Text input

The text input can be effected via the programming unit PØ5/P1Ø/P21 or via a peripheral unit with current loop interface:

a) Text input by means of the programming unit PØ5/P1Ø via PGU-connector

The input is effected as usual in decimal format. Now, however, the values 128...255 are possible in addition to the ASCII-characters 1...127.

b) Text input by means of the peripheral unit via the text editor of the PCA

For this kind of text input a peripheral unit with current loop interface transmitting 8 data bits must be available. Via the PCA either 7 or 8 data bits per character are determined in the 2nd line of the PAS 1ØØ instruction.

Text output

a) Text display of the text memory via the text editor of the PCA

Irrespective of whether a peripheral unit functions with 7 or 8 data bits per character, the 256 characters are output via the text editor as follows:

Character no.	Editor output	Text output via user program
Ø...31	^ ⌊⌋	ASCII code (as usual)
32...127	• ⌊⌋	
128...159	! ⌊⌋ 1)	Not defined, different for each peripheral unit
16Ø...255	: ⌊⌋ 2)	

⌊⌋ Character

1) An ASCII symbol 64...95 appears as character
2) An ASCII symbol 32...127 appears as character

b) Text output via user program

Characters of 7 or 8 data bits per character can be output, depending on the selected communication mode. Moreover, the additional 128 characters allow generating for example instructions for graphic displays.

K 6 Data exchange via serial interface

K 6.1 Introduction, communication modes

The previous chapter was dealing with text input/output. While text input is a pure "off-line" function (like programming the user memory), the output can proceed "on-line". The texts are started via user program e.g. by an input signal and output via the serial interface. I.e. this is one-way information from the PLC to a peripheral unit (typical application: logging). A dialog between the peripheral unit and the PLC is not yet possible with the text output alone.

This chapter is dealing with operating modes of the serial interface which allow a real dialog (two-way exchange of information) between the following units:

- PCA14 *) with "non-intelligent" peripheral such as TTY or video terminal
- PCA14 *) with another PLC, e.g. PCAØ, PCA14, PCA23
- PCA14 *) with another "intelligent" system (e.g. computer).

This exchange of information is performed in the following modes:

Mode C Exchange of individual ASCII characters

Mode C is particularly suitable for dialogs between the PCA and a manually operated peripheral unit (TTY or video terminal). 256 different characters (128 out of which are ASCII-characters) can be exchanged via the serial interface and assigned different user functions.

Mode N Exchange of numerical data in telegram format acc. to DIN 66Ø19

In mode N, the exchange of information is accomplished in accordance with DIN 66Ø19. Transmission reliability is extremely high due to current-loop technology and check sum (BCC). The range of application is increased considerably as a max. of 31 numerical characters in one telegram. Coded information can be received, stored and evaluated for example by a reading station. It is also possible to communicate with another "intelligent" system.

Another version which also allows the exchange of numerical data with a manually operated "non-intelligent" peripheral unit was produced for simple installations. This version is also very useful during programming and commissioning. Three variants are available in mode N: N1, N2 and N3. Detailed explanations will be given in the following description.

---

*) or other PCA with a serial interface such as PCAØ2, PCA222, PCA232.

Mode P Exchange of PCA-specific data in telegram format acc. to DIN 66Ø19

The operating modes C and N already allow indirect access to inputs, outputs, flags, timers and counters. In operating mode P this access is direct and moreover extended to step addresses and operands. Thus, access to the PCA via a superior intelligent system is possible. The PCA is always passive. Two variants are distinguished: P1 and P2.

Combination of various operating modes

Interesting possibilities result from the combination of various operating modes with each other and just with text output.

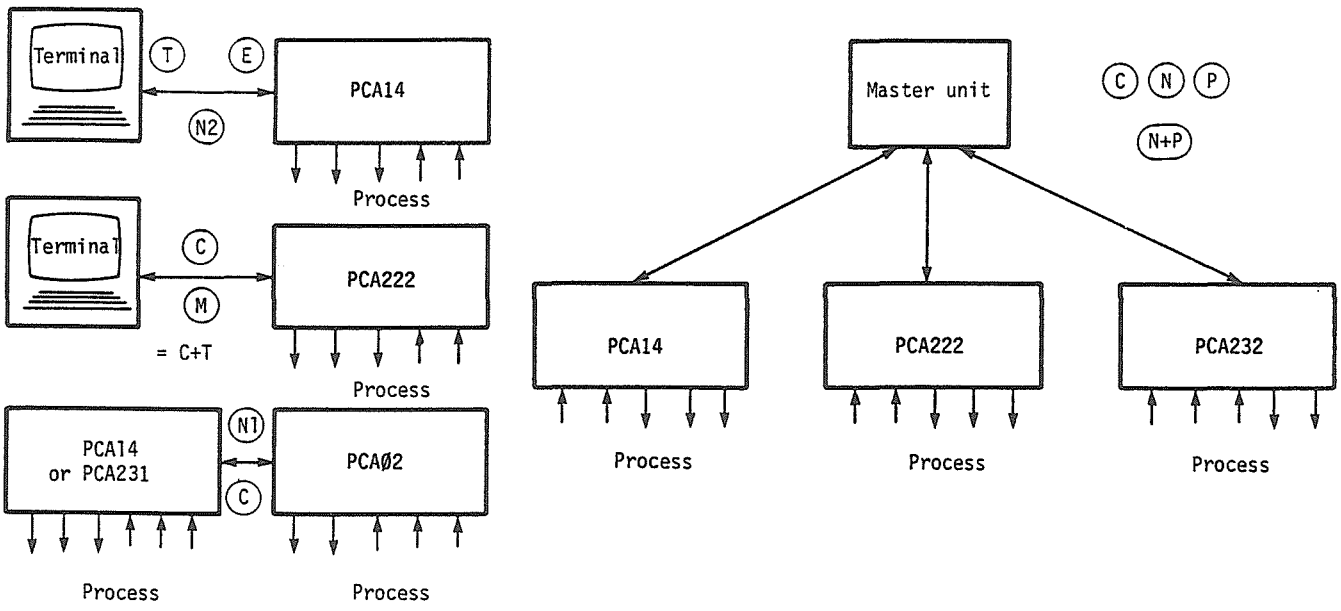
Menu mode - combination of text output with mode C (single characters)

With this combination an on-line dialog with output of clear text between an operated peripheral unit and the PCA is possible (manually operated). A menu is output by means of text. Single characters are read in and the corresponding functions triggered in the PCA by key actuation.

Computer mode - combination of mode N with mode P

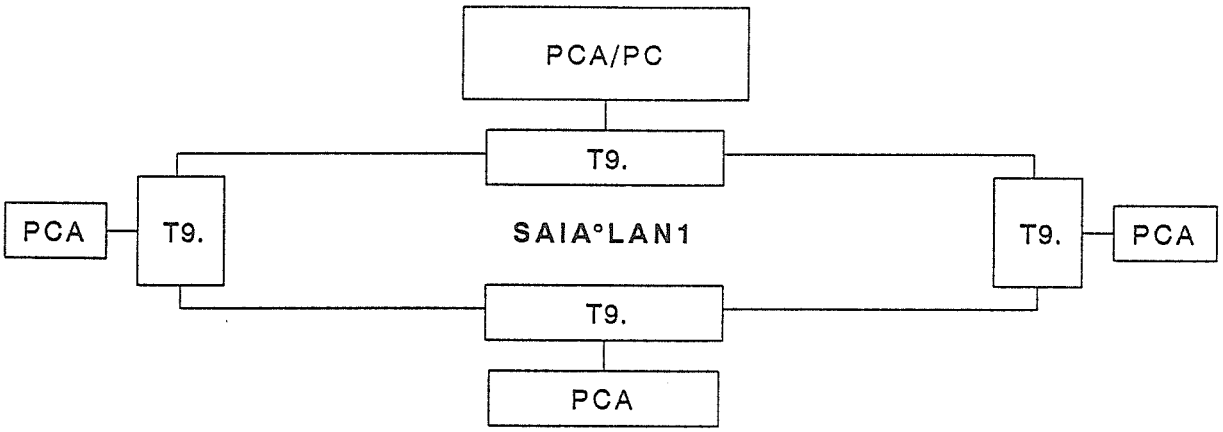
As a result of combined exchange of numerical and PCA-specific data, there are many ways of influencing the PCA by a superior system and of sending acknowledgements to the hierarchically higher system. Thus, the control structure of "distributed intelligence" can easily be realized.

Combination of units for point-to-point connections



SAIA°LAN 1

All communication modes including text output may also be used in the local network SAIA°LAN 1. After calling a destination station from a source station, communication may be accomplished completely transparently like in direct point-to-point connections. Please ask for the special SAIA°LAN 1 documentation.



Summary of operating modes for the serial interface

Text input/output to/from text memory

Mode EDITOR	(E)	Text input off-line via editor	→	Text memory
Mode TEXT	(T)	Text output acc. to user program	←	

Data exchange in the flag buffer

Mode C	(C)	Input/output of single characters	→ ←	Flag buffer
Mode N (1/2/3)	(N)	Input/output of numerical data in telegram format acc. to DIN 66Ø19	→ ←	

C and N are dependent on user program (flag buffer)

Direct communication with PCA

Mode P (1/2)	(P)	Direct input/output of PCA data in telegram format acc. to DIN 66Ø19	→ ←	PCA PCA

P is independent of user program (master-slave operation)

User guidance

Menu mode (T+C)	(M)	Input of single characters	→ (C)	Flag buffer
		Output of single characters and/or of texts	← (C)	Text memory
			← (T)	

Communication with superior system

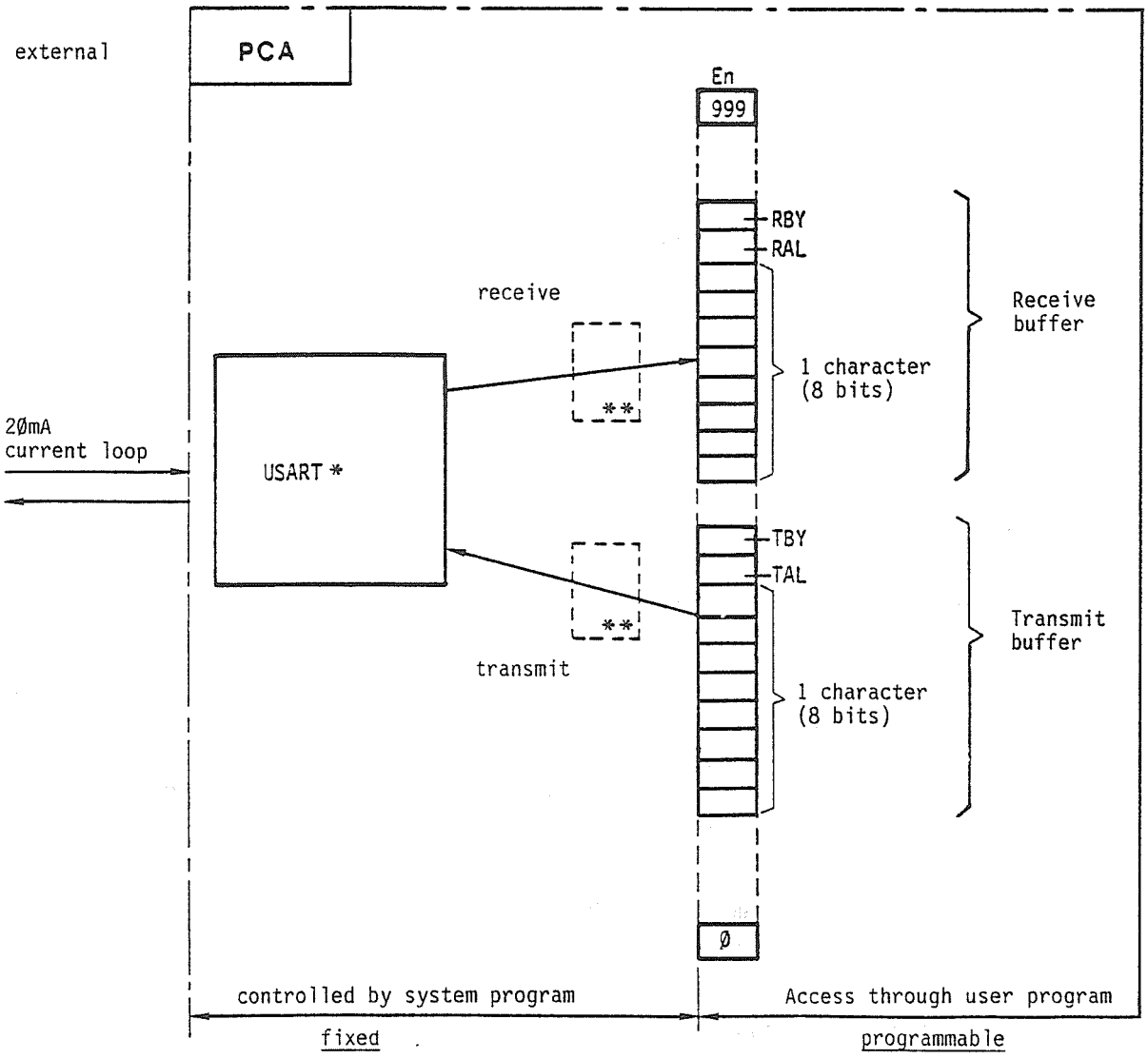
Computer mode (N+P)		Input/output of data with direct access	→ ← (P)	PCA PCA
		Input/output of numerical data via flag buffer	→ ← (N)	Flag buffer

K 6.2 Definition of mode C

In mode C, individual ASCII-characters are input and output. In order to activate the serial interface of the PCA for this kind of transmission, appropriate program packages are contained in the system program. Which assume the function of interpreting the incoming character and formatting the character to be transmitted.

As shown in the following figure, the user must define addresses for the receive buffer (character input location) and for the transmit buffer (character output location) in the flag buffer.

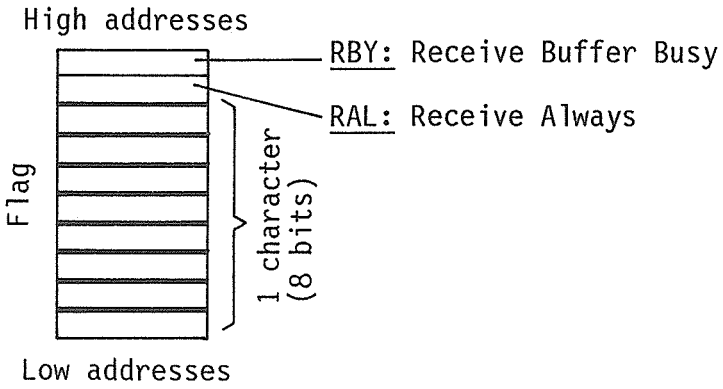
The following diagram shows these relations:



*) USART = Universal Synchronous/Asynchronous Receiver Transmitter.

**) Strictly speaking, the telegram is first of all sent to an intermediate storage. There it is checked for correctness and in case of a positive result transmitted to the appropriate flag buffers by the system program.

Receive buffer for mode C



If no transmission errors (parity, framing etc.) were detected during reception, storage of the character is effected in the receive buffer dependent on the logic state of RBY and RAL:

	RAL	RBY	1st sequence	2nd sequence
Receive always	H	L	Intermediate storage in buffer	→ RBY becomes H
	H	H		
One telegram only	L	(L)	Intermediate storage in buffer	→ RBY becomes H
	L	H	No effect	--

Receive always:
If the user program sets RAL = H and reception is not faulty, the characters are stored in the receive buffer at any time.

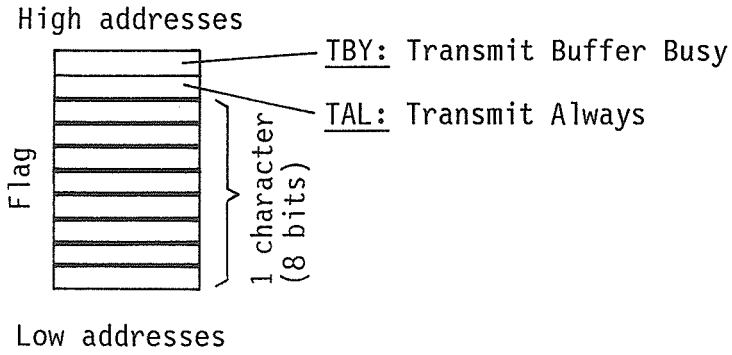
One telegram only:
However, if RAL = L is set and reception is not faulty, the character is stored in the buffer, if RBY also = L. Then RBY = H is automatically set, preventing a new character from overwriting the receive buffer. This means that the user program processes the buffer bits, before the following character is accepted by REO RBY.

Please note: The system program itself does not generate any echo. If an echo is required, this acknowledgement must be generated via the user program.

Program structure for individual reception: (RAL = L)

```
WIL   RBY     ; Wait, if a new character has not yet been received.

Reading       ; The received character is read by the user program
the receive     (possibly transferred to counter).
buffer

REO   RBY     ; Receive buffer is ready for receiving a new character.
```

Transmit buffer for mode C



Each 1/1Øs resp. 1/1ØØs (depending on the time base selected) the USART can be ordered to output character. This is done depending on the states of the TBY and TAL flags:

	TAL	TBY	1st sequence	2nd sequence
Transmit always	H	L	Character output from buffer	
	H	H		→ TBY becomes L
One telegram only	L	L	No effect	--
	L	(H)	Character output	→ TBY becomes L

Transmit always:

If TAL = H (set by the user program), the character formed by the buffer content is transmitted every 1/1Øs resp. 1/1ØØs.

One telegram only:

However, if TAL = L, the character waiting in the transmit buffer is only transmitted if the user program sets TBY = H. The system program then automatically sets TBY = L to show end of transmission to the user program (so-called handshaking).

The system program does not expect an acknowledgement after the output of the character.

Program structure for single transmission (TAL = L)

```
        WIH   TBY     ; Wait, if previous transmission has not been finished yet.
  ┌──────────────┐
  │ Filling the  │
  │ transmit     │  ;  A character (8 bits) is transferred to the transmit
  │ buffer       │     buffer by the user program.
  └──────────────┘
        SEO   TBY     ; Transmission of the character from the transmit buffer.
```

K 6.2.1 Mode C assignment

```
 1:  PAS (29)    1ØØ ; Activation of the interface
 2:       ØØ    xxxx ; Baud rate, parity and stop bits
 3:       ØØ     xxx ; TXB (Text-Busy Flag)
 4:       ØØ     xxx ; RBY (Receive Buffer)
 5:       ØØ     xxx ; TBY (Transmit Buffer)
 6:       ØØ       Ø ; BCC not possible
 7:       ØØ       Ø
 8:       ØØ       Ø
 9:       ØØ       Ø
1Ø:       ØØ       Ø
```

Characteristic of mode C

NUL

- Line 3: TXB (Text-Busy Flag)

  For xxx insert the address of a free flag (or output) which is "L" on switching on the PCA.

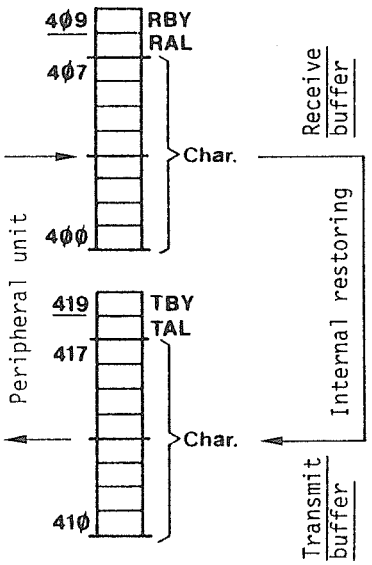
- Line 4: RBY (receive buffer)

  1Ø related free flags (or possibly outputs) must be reserved for the receive buffer, whereby xxx is the highest address.

- Line 5: TBY (transmit buffer) see RBY, line 4

  It is always necessary to define a receive and a transmit buffer, even if only one should be in use.

Example: Generation of an "echo" by the user program. The character arriving at the receive buffer is transmitted to the transmit buffer and is thus displayed on the monitor.



Program (for monitor VC 44Ø4)

```
Ø   NOP    Ø ;
1   PAS   1ØØ ;
2    ØØ   9Ø3 ; (96ØØ bauds)
3    ØØ   399 ; TXB
4    ØØ   4Ø9 ; RBY
5    ØØ   419 ; TBY
6    ØØ     Ø ;
.    .      .
.    .      .
1Ø   ØØ     Ø ;
11  WIL   4Ø9 ; Receiver clear?
12  SCR   256 ;  Read re-
13   24   4Ø7 ;  ceive buffer
14  REO   4Ø9 ; Receive buffer free!
15  WIH   419 ; Transmitter clear?
16  SCR   256 ;  Transfer to
17   21   417 ;  transmit buffer
18  SEO   419 ; Transmit
19  JMP    11 ;
```

Assignment (Initialization only when switching on)

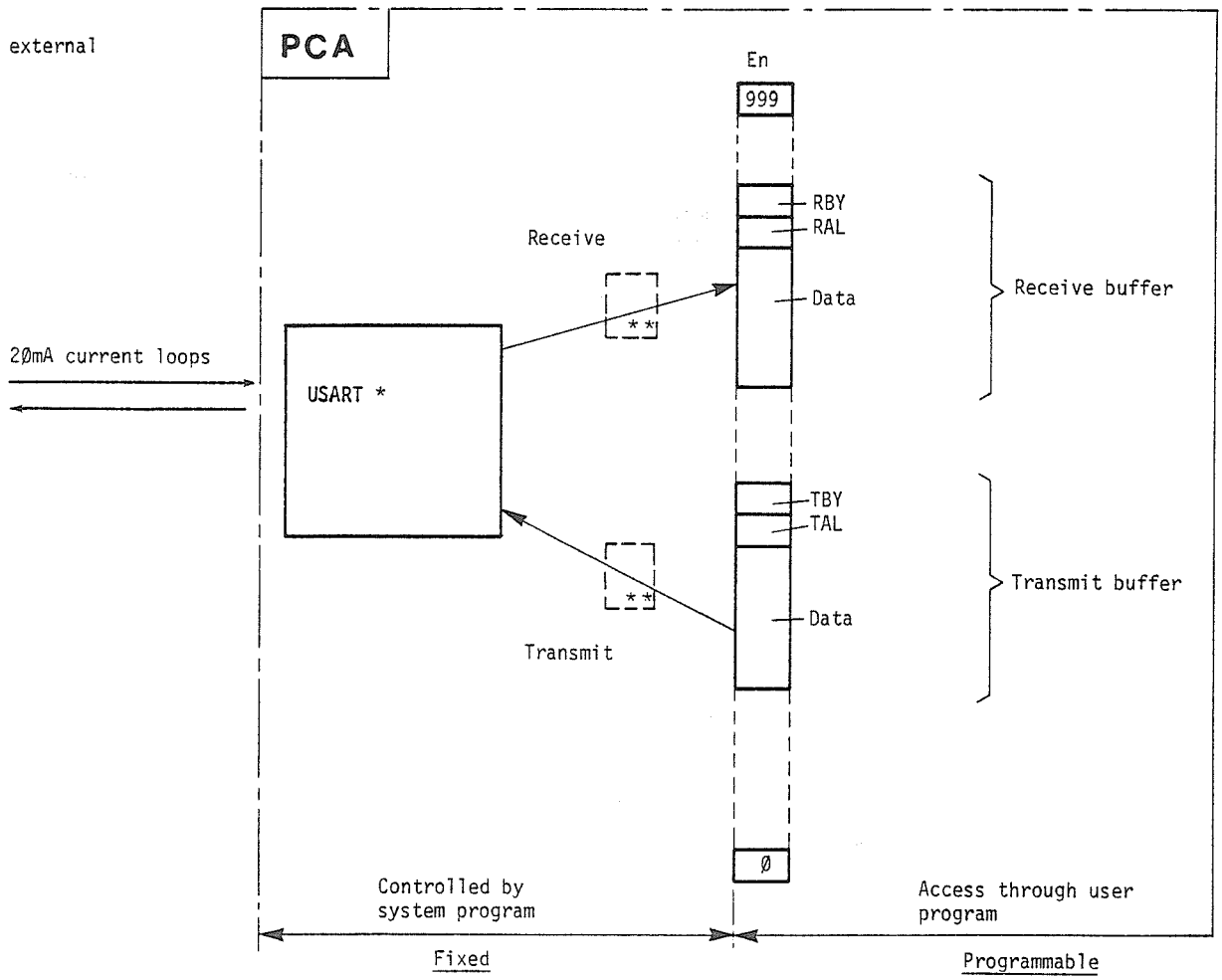
Program

K 6.3 Definition of mode N

In mode N, numeric data is exchanged in telegram format in accordance with DIN 66Ø19. In order to activate the serial interface of the PCA for this kind of transmission, appropriate program packages are contained in the system program which assume the function of interpreting the incoming information and formatting of the transmitted information. The information is transferred to element addresses (mostly flags) by the system program. They can be accessed via the user program.

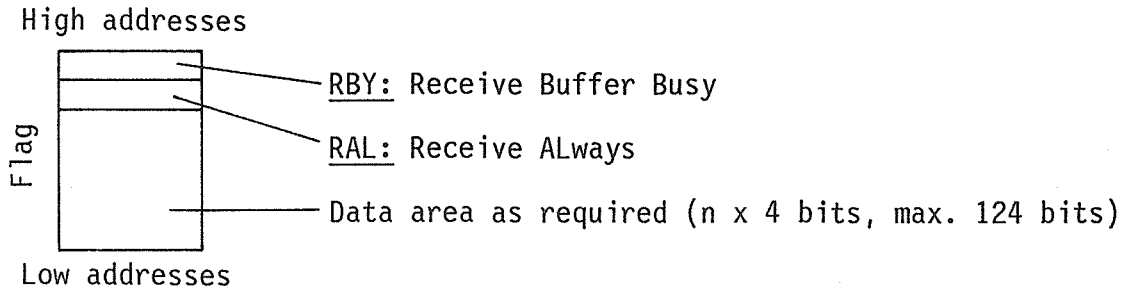
As shown in the following figure, the user must define addresses for the receive buffer (data input location) and for the transmit buffer (data output location) in the flag buffer.

The size of the receive and transmit buffer is variable and can be specified in accordance with the application.



*) USART = Universal Synchronous/Asynchronous Receiver Transmitter.
**) Strictly speaking, the telegram is first of all transmitted to an intermediate storage. There, it is checked for correctness and, in the case of a positive result, transmitted to the appropriate flag buffer by the system program.

Receive buffer for mode N



If no transmission errors (parity, framing etc.) were detected during reception, storage of data in the receive buffer is effected dependent on the logic states of RBY and RAL:

	RAL	RBY	1st sequence	2nd sequence
Receive always	H	L	Intermediate storage in buffer	→ RBY becomes H
	H	H		
One telegram only	L	(L)	Intermediate storage in buffer	→ RBY becomes H
	L	H	No effect	--

Receive always:

If the user program sets RAL = H and reception is not faulty, the data is stored in the receive buffer at any time.

One telegram only:

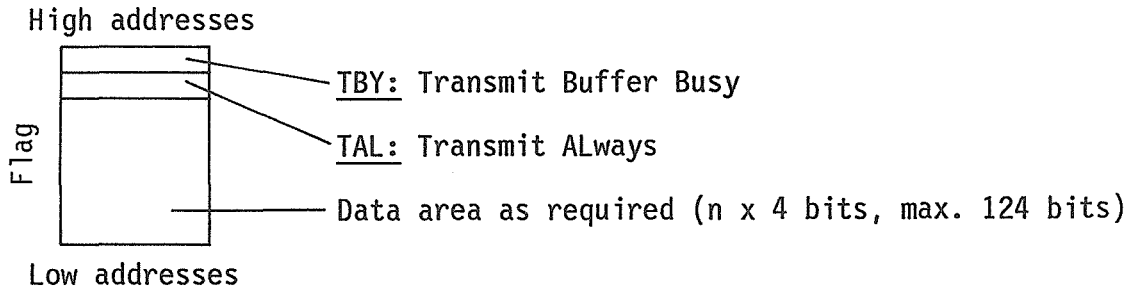
However, if RAL = L is set and reception is not faulty, the data is only stored in the buffer, if RBY also = L. RBY = H is then automatically set, preventing new data from overwriting the receive buffer. This means that the user program processes the buffer bits, before the following data will be accepted by REO RBY.

Program structure for individual reception (RAL = L)

```
        WIL   RBY     ; Waiting, if new data has not yet been received.
  ┌──────────────┐
  │ Reading the  │
  │ receive      │     ; The received data is read by the user program (and
  │ buffer       │       possibly transferred to the counter register).
  └──────────────┘
        REO   RBY     ; Receive buffer is ready for receiving new data.
```

Note: RBY becomes H on reception of the first character; therefore, it must be ensured after WIL that all characters have been received. The wait time depends on the baud rate.

Transmit buffer for mode N



Each 1/10s resp. 1/100s (depending on the time base selected) the USART can be ordered to output data. This is done depending on the states of the TBY and TAL flags:

	TAL	TBY	1st sequence	2nd sequence
Transmit always	H	L	Data output from buffer →	→ TBY becomes L
	H	H		
One telegram only	L	L	No effect	--
	L	Ⓗ	Data output from buffer →	→ TBY becomes L

Transmit always:

If TAL = H (set by the user program), the data formed by the buffer content is transmitted every 1/10s resp. 1/100s.

One telegram only:

However, if TAL = L, the data in the transmit buffer is only transmitted if the user program sets TBY = H. Then the system program automatically sets TBY = L to indicate the end of transmission to the user program (so-called handshaking).

Program structure for single transmission: (TAL = L)

```
        WIH   TBY     ; Wait, if previous transmission has not been finished yet.
  ┌──────────────┐
  │ Filling the  │ ; The data to be transmitted is transferred to the transmit
  │ transmit     │   buffer by the user program.
  │ buffer       │
  └──────────────┘
        SEO   TBY     ; Transmission of buffer content
```

Note: TBY becomes L on transmission of the first character; therefore it must be ensured after WIH that all characters have been transmitted. The wait time depends on the baud rate.

K 6.3.1 Mode N assignment

```
 1:  PAS (29)   1ØØ ; Activation of the interface

 2:       ØØ   xxxx ; Baud rate, parity and stop bits

 3:       ØØ    xxx ; TXB (Text-Busy Flag)
 4:        n*   xxx ; RBY (Receive Buffer)
 5:        n*   xxx ; TBY (Transmit Buffer)
 6:      { ØØ     d ; without BCC (var. no. 2)
         { Ø1     d ; with    BCC (var. no. 1)

 7:       ØØ     Ø
 8:       ØØ     Ø
 9:       ØØ     Ø
1Ø:       ØØ     Ø
```

Lines 3, 4, 5, 6: Characteristic of mode N

Lines 7, 8, 9, 10: NUL

- Line 3: TXB (Text-Busy Flag)

  For xxx insert the address of a free flag (or output) which is "L" when switching on the PCA.

- Line 4: RBY (receive buffer)*

  n = Ø1...31 corresponding to a data buffer of 4 x n bits (i.e. 4 to 124 bits).
  A total of 2 + 4 x n related free flags (or possibly outputs) must be reserved, whereby xxx is the highest address (RBY).

- Line 5: TBY (transmit buffer)* see RBY, line 4

  The transmit and receive buffers can also have different sizes.

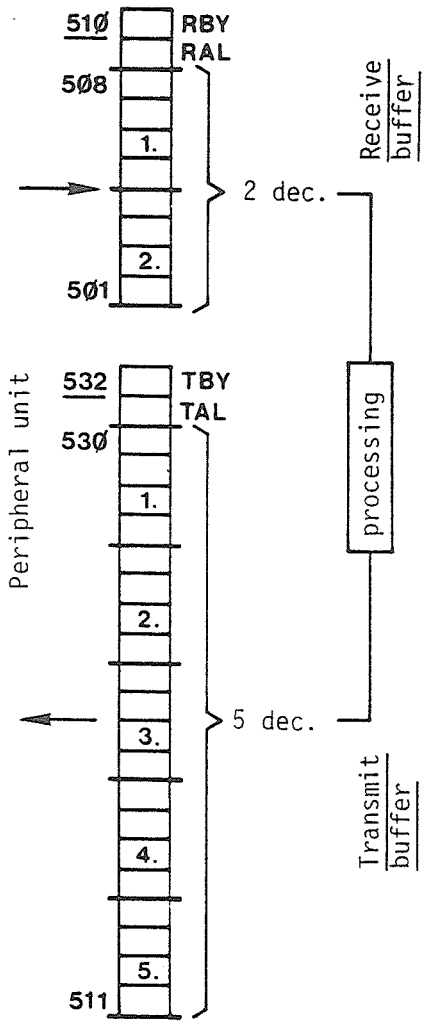
- Line 6:

  Line 6 has multiple significance: With the code it can be determined whether BCC is used or not.
  Code Ø1: With BCC (the peripheral unit must be able to process BCC)
  Mode variant N1
  Code ØØ: Without BCC (operation with a non-intelligent peripheral unit)
  Mode variant N2

  In case of operand d = Ø the unacknowledged telegram is repeated every 3Øth time unit (every 3s for 1/1Øs).
  In case of operand d = 1 the telegram is transmitted only once irrespective of whether it was acknowledged or not. If the acknowledgement is missing, TBY remains high. Time-out monitoring is necessary.
  In case of operand d = 3 is also transmitted only once, whereby the PLC does not expect acknowledgement. Mode variant N3.

*) If only one of the two buffers is used (only reception or transmission), the unused buffer must be specified nevertheless, e.g. Ø1 399, i.e. having a minimum size of 2 + 4 bits from 394...399.

Example: (mode N2) Reception of a 2-digit value, processing it and transmission of the 5-digit result. As communication must be effected with a "non-intelligent" operated unit, BCC is not used (mode N2). Enter CTRL/B (for STX) prior to each data input and CTRL/C (for ETX) following the data to transmit the telegram.



Program (for SILENT 743)

```
Ø   NOP   Ø
1   PAS   1ØØ
2    ØØ   898  ; (3ØØ bauds)
3    ØØ    32  ; TXB
4    Ø2   51Ø  ; RBY
5    Ø5   532  ; TBY
6    ØØ    Ø  ; no BCC
7    ØØ    Ø
8    ØØ    Ø
9    ØØ    Ø
1Ø   ØØ    Ø
11  WIL   51Ø  ; Receiver clear?
12  SCR   256  }  Transmit
13   16   5Ø8  }  R-buffer
14  REO   51Ø  ; R-buffer clear!
15  WIH   532  ; Transmitter clear?
16  SCR   256  }  x 64
17   29    64  }
18  SCR   256  }  Result on trans-
19   2Ø   53Ø  }  mit buffer
2Ø  SCR   256  }  Result to
21   2Ø    63  }  outputs
22  SEO   532  ; Transmit
23  JMP    11
```

Assignment

Program

K 6.3.2 Mode N telegrams

a) Control characters

Several control characters of the "Control case" in the ASCII-table are used in a telegram. The following table provides an overview.

Mnemo codes	Designation	Location or dec. no.	CTRL/..
STX	Start of Text	2	B
ETX	End of Text	3	C
EOT 1)	End of Transmission	4	D
ENQ 2)	Enquiry	5	E
ACK	Acknowledge	6	F
NAK	Negative Acknowledge	21	U
#	Hash	35	

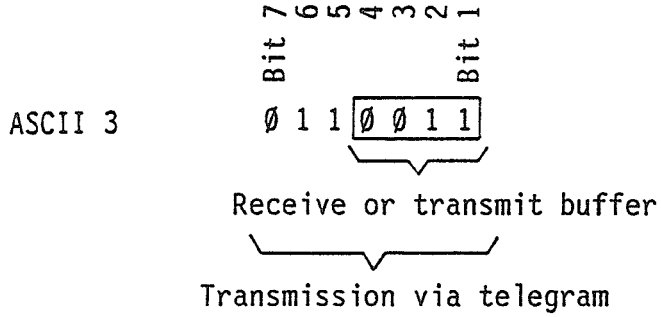
1) EOT causes the PCA to reassume quiescent state.
2) ENQ expects an acknowledgement of the PCA (telegram or NAK) or cancels a telegram which has already been started.

b) The transmitted telegram

In mode N only numeric data is exchanged. These are the characters of column 3, decimal no. 48 through 63 in the ASCII-table. The bit numbers 5, 6 and 7 are always the same for these characters (see ASCII-table). The receive and transmit buffers are therefore formed with groups of 4 bits, one group thus corresponds to a numerical character.

During transmission, the system adds the constant bits 5, 6 and 7 and performs the BCC sum check for all data bytes incl. ETX. If the check sum is identical to that of the receiving PCA, the data is transferred to the receive buffer without the constant bits 5, 6 and 7.

e.g.:



c) What does BCC mean?

BCC = Block Check Character (check sum)

The horizontal check sum of the part comprising data and ETX of a telegram which starts off with STX and ends with ETX is established according to the following example and then compared to the transmitted value. Thus, extremely high transmission reliability is achieved.

Telegram:

STX	Data					ETX	BCC
CTRL/B	3	6	9	=	8	CTRL/C	Block check of data plus ETX
B1 ØØ1Ø	ØØ11	Ø11Ø	1ØØ1	11Ø1	1ØØØ	ØØ11	1Ø1Ø
B7 ØØØ	Ø11	Ø11	Ø11	Ø11	Ø11	ØØØ	Ø11

d) Structure of a telegram

A telegram is composed of the following parts:

STX = Beginning of the telegram

DATA = Content of the telegram

ETX = End of the telegram

BCC = Block check character

Telegram: STX DATA ETX BCC ⟶

Normally, every telegram must be followed by one of the following acknowledgements:

ACK = The telegram was received correctly

NAK = The telegram was not received correctly

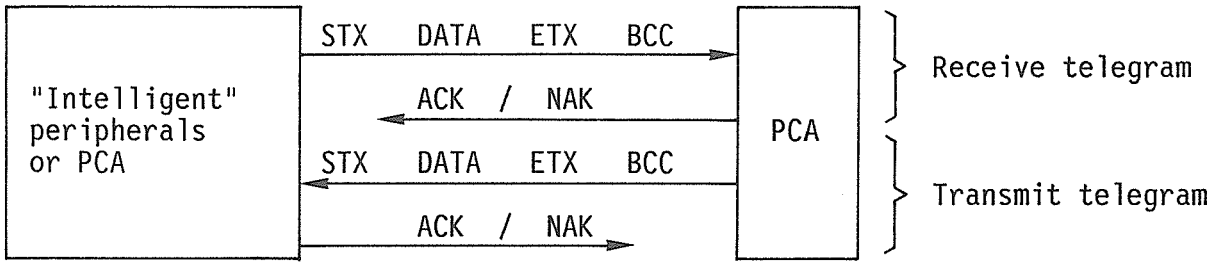
Response: ACK or NAK ⟵

e) Exchange of telegrams with the PCA in mode N

In order to meet the various requirements, the user can proceed according to DIN standard 66Ø19 as well as the following variants.

e1) Procedure according to DIN 66Ø19 - mode N1

The standard defines the exchange of data between two systems. It supposes that the two systems which are to communicate, are capable of interpreting the characters STX and ETX, the BCC and the communication characters ACK, NAK, ENQ and EOT.

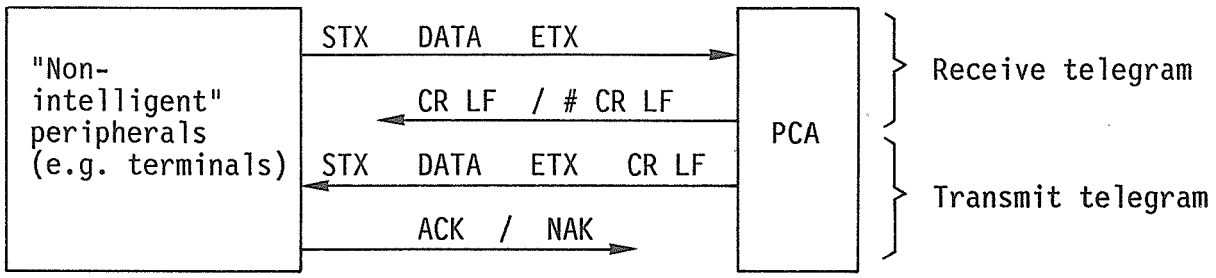


e2) Variant - mode N2

If a manually operated terminal is used for communication, it is almost impossible to determine the BCC. To allow the data exchange with a simple peripheral unit nevertheless (e.g. for establishing the program or for commissioning), a variant has been created which does not use the BCC check and in which the invisible characters ACK and NAK are replaced as follows:

ACK ---> is replaced by "CR LF" (carriage return with line feed)

NAK ---> is replaced by # "CR LF".



Determination of the variant in the PAS 1ØØ instruction

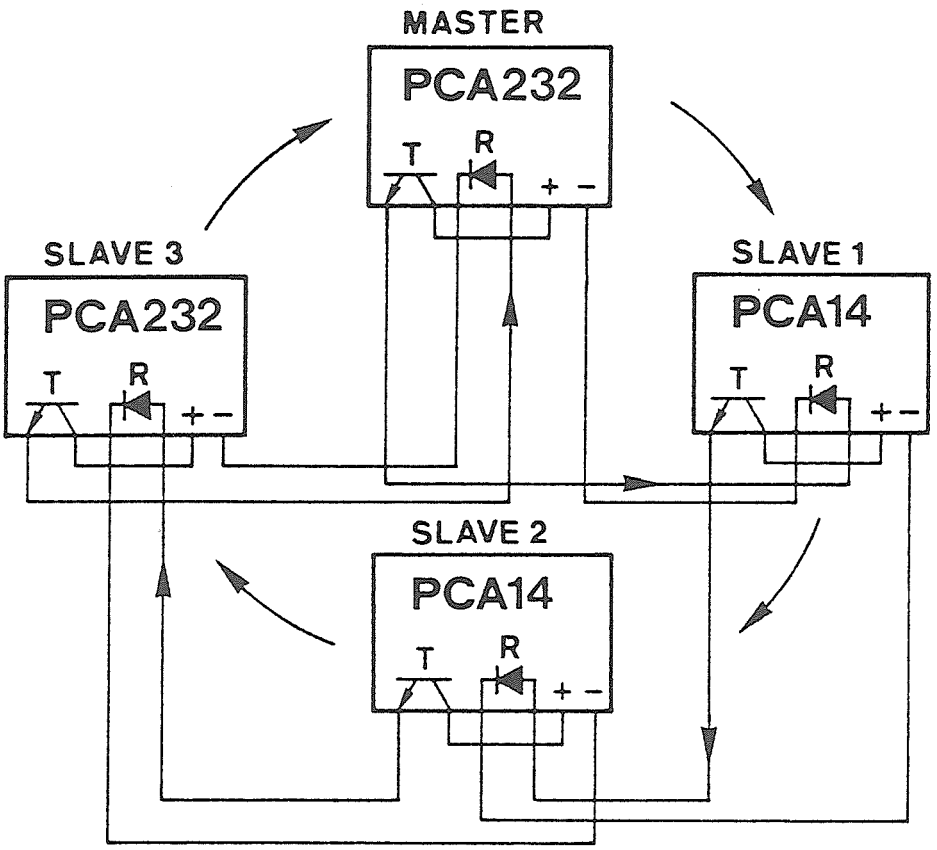
This is done in line 6:

CODE = Ø1 ---> with BCC, operation with "intelligent" unit according to DIN 66Ø19 ---> mode N1 (acknowledgement invisible)

CODE = ØØ ---> no BCC, operation with manually operated peripheral unit ---> mode N2 (acknowledgement visible)

e3) Mode N3

Mode N3 was created to allow the transmission of data among a cycle of several PCs. It is of advantage to define a PC as master.



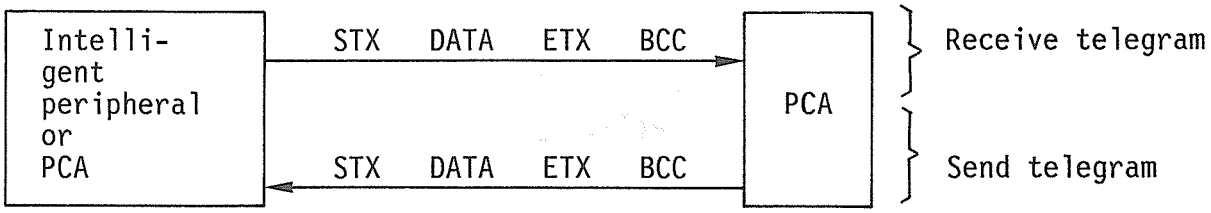
While in mode N1 and N2 a telegram received by the PCA must be acknowledged (ACK/NAK), this acknowledgement is not required for mode N3. This means, that the PLC which transmitted a telegram does not expect an answer of the system receiving the telegram.

The returning telegram is a confirmation to the master that the information was transmitted by all PCA of the cycle.

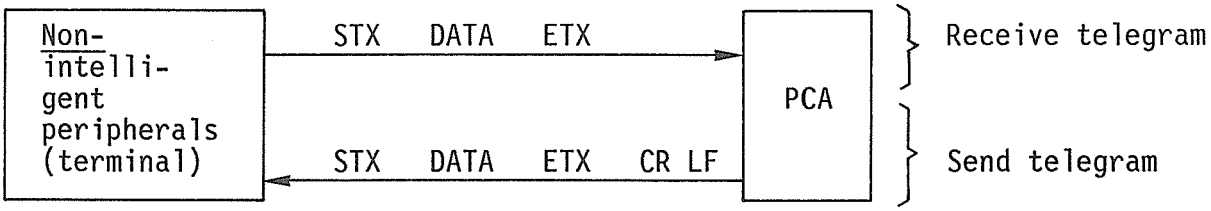
Due to the sum check BCC, transmission is very reliable. If a transmission error is detected, the arriving telegram is not transferred to the flag buffer.

The telegram mode N3

- With BCC:



- Without BCC:



Variant N3 is defined in line 6 of PAS 1ØØ (see chapter K 6.3.1).

K 6.4 Definition of mode P

Mode P allows direct access to the PCA data using telegram format. In modes C and N described so far, data is exchanged via a flag buffer. In mode P, an "intelligent" system has direct access to the PCA data, i.e. the data is accessed outside the user program function range (and thus without using a receive or transmit buffer), whereby it is urgently required that the PCA executes the instructions and can only answer with "ACK" (NAK, if the instruction could not be executed).

PCA data includes:

Elements	(addr. Ø.... 999)	---> character E
8 elements	(addr. 7.... 999)	---> character e
Timer register	(addr. 256.... 287)	---> character T
5 timer registers	(addr. 26Ø.... 287)	---> character t [1]
Counter register	(addr. 256.... 511)	---> character C [2]
5 counter registers	(addr. 26Ø.... 511)	---> character c [3]
Operand register		---> character O
Step addresses	(addr. Ø... 8191)	---> character S
2-byte text memory	(addr. Ø...15999)	---> character M [4]
5 x 2-byte text memory	(addr. 4...15999)	---> character m [4]
1 register word (8 bits)	(addr. Ø.... 999)	---> character R [5]
1 register block (5 x 8 bits)	(addr. 4.... 999)	---> character A [5]
3 register blocks (15 x 8 bits)	(addr. 14.... 999)	---> character B [5]
1Ø digits of data memory	(addr. 4...15999)	---> character N [6]
3Ø digits of data memory	(addr. 14...15999)	---> character n [6]
For writing use		----------> character (W) (write)
For reading use		----------> character (D) (display)

[1] PCAØ2 from version V6.13Ø on
PCA14 from version V6.Ø31 on
PCA222 from version V6.23Ø on
PCA232 from version V7.131 on

[2] PCAØ2 : addr. 256...319 (addr. 256...479 from version V6.132 on)
PCA14 : addr. 256...319 (addr. 256...479 from version V6.Ø34 on)
PCA222: addr. 256...319 (addr. 256...479 from version V6.23Ø on)
PCA232: addr. 256...511

[3] PCAØ2 from vers. V6.13Ø on (addr. 26Ø...319, addr.26Ø...479 from V6.132 on)
PCA14 from vers. V6.Ø31 on (addr. 26Ø...319, addr.26Ø...479 from V6.Ø34 on)
PCA222 from vers. V6.23Ø on (addr. 26Ø...479)
PCA232 from vers. V7.131 on (addr. 26Ø...511)

[4] PCAØ2 from vers. V6.13Ø on (highest address 4Ø95)
PCA14 from vers. V6.Ø31 on (highest address 8191)
PCA222 from vers. V6.23Ø on (highest address 8191)
PCA232 from vers. V7.131 on (highest address 15999)

[5] For PCA231, 232 only.

[6] Telegrams N and n only exist for PCA232 from version V7.131 on.
The two telegrams are only reasonable, if values were previously transferred from the word register to the data memory with PAS 57 instruction.

K 6.4.1 Mode P assignment

a) PAS 1ØØ instruction
The PAS 1ØØ instruction in mode P is identical with that in mode N.
The telegram, however, is different.

```
 1:  PAS (29)   1ØØ ; Activation of the data interface

 2:       ØØ   xxxx ; Baud rate, parity and stop bits

 3:       ØØ    xxx ; Element that is certainly "L"

 4:       Ø1    xxx ; Highest address of 6 flags

 5:       Ø1    xxx ; Highest address of 6 further flags

 6:      { ØØ      Ø ; Without BCC (variant P2)
         { Ø1      Ø ; With    BCC (variant P1)

 7:       ØØ      Ø }
 8:       ØØ      Ø }
                     } NUL
 9:       ØØ      Ø }
1Ø:       ØØ      Ø }
```

- Line 5: TxB (Text-Busy Flag)
  xxx must be replaced with the address of a free flag (or I/O), which is "L" on switching on.

- Line 4: RBY (Receive Buffer)
  If only mode P is used, 2x6 flags must be reserved. In computer mode (mode N + P), lines 4 and 5 can be assigned according to the N-mode used.

- Line 5: TBY (Transmitt Buffer)
  See line 4

- Line 6: BCC
  Code = ØØ ---> no BCC, operation with manually operated peripheral unit (acknowledgement visible) ---> mode P2
  Code = Ø1 ---> with BCC, operation with "intelligent" unit (acknowledgement invisible) ---> mode P1

K 6.4.2 Mode P telegrams

b) Control characters
Different characters are used in a telegram which are contained in the "Control Case" of the ASCII-table. The following table provides a summary:

Mnemo codes	Designation	ASCII dec. no.	CTRL/..
STX	Start of Text	2	B
ETX	End of Text	3	C
EOT 1)	End of Trans-mission	4	D
ENQ 2)	Enquiry	5	E
ACK	Acknowledge	6	F
NAK	Negative Acknowledge	21	U
#	Hash	35	-

1) EOT causes the PCA to return to the resting state.
2) ENQ expects an acknowledgement of the PCA (telegram or NAK) or cancels a telegram already started.

c) What does BCC means?

BCC = Block Check Character

The horizontal check sum of the part comprising data and ETX of a telegram which starts off with STX and ends with ETX is established according to the following example and then compared to the transmitted value. Thus, extremely high transmission reliability is achieved.

Telegram:

STX	Data					ETX	BCC	
CTRL/B	(W)	[E]	Ø	3	2	1	CTRL/C	CTRL/Q
B1 ØØ1Ø	Ø11	Ø1Ø1	ØØØØ	ØØ11	ØØ1Ø	ØØØ1	ØØ11	ØØØ1
B7 ØØØ	1Ø1	1ØØ	Ø11	Ø11	Ø11	Ø11	ØØØ	ØØ1

Block check of data plus ETX

c) Structure of a telegram

A telegram is composed of the following parts:

STX = Beginning of the telegram

DATA = Content of the telegram

ETX = End of the telegram

BCC = Block check character

Telegram: STX DATA ETX BCC ⟶

The response to this telegram will be:

ACK = The telegram was received correctly

NAK = The telegram was not received correctly

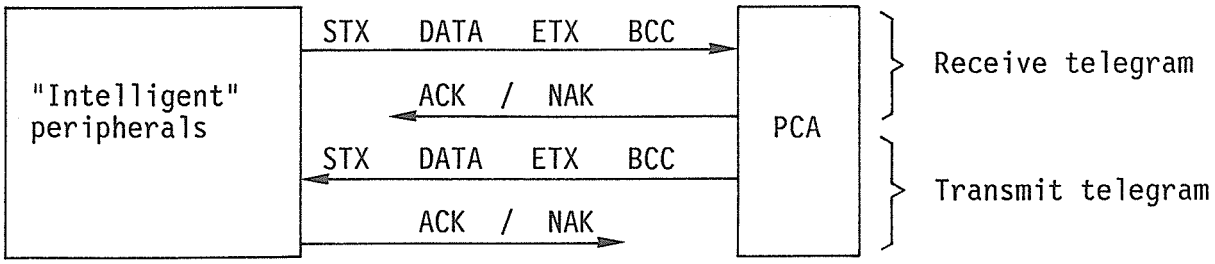
Response: ⟵ ACK or NAK

d) Exchange of telegrams with the PCA in mode P

In order to meet the various requirements the user can proceed according to DIN standard 66Ø19 as well as the following variants.

d1) Procedure according to DIN 66Ø19 - mode P1

The standard defines the exchange of data between two systems. It supposes that the two communicating systems are capable of interpreting the characters STX and ETX, the BCC and the communication characters ACK, NAK, ENQ and EOT.

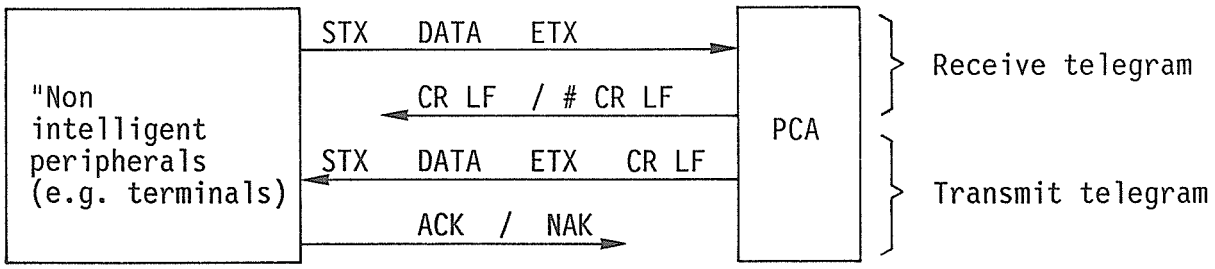


d2) Mode P2

If a manually operated terminal is used for communication, it is almost impossible to determine the BCC. To allow the data exchange with a simple peripheral unit nevertheless (e.g. for establishing the program or for commissioning), a variant has been created which does not use the BCC check and in which the invisible characters ACK and NAK are replaced as follows:

ACK ---> is replaced by "CR LF" (carriage return with line feed)

NAK ---> is replaced by # "CR LF".



Determination of the variant in the PAS 1ØØ instruction

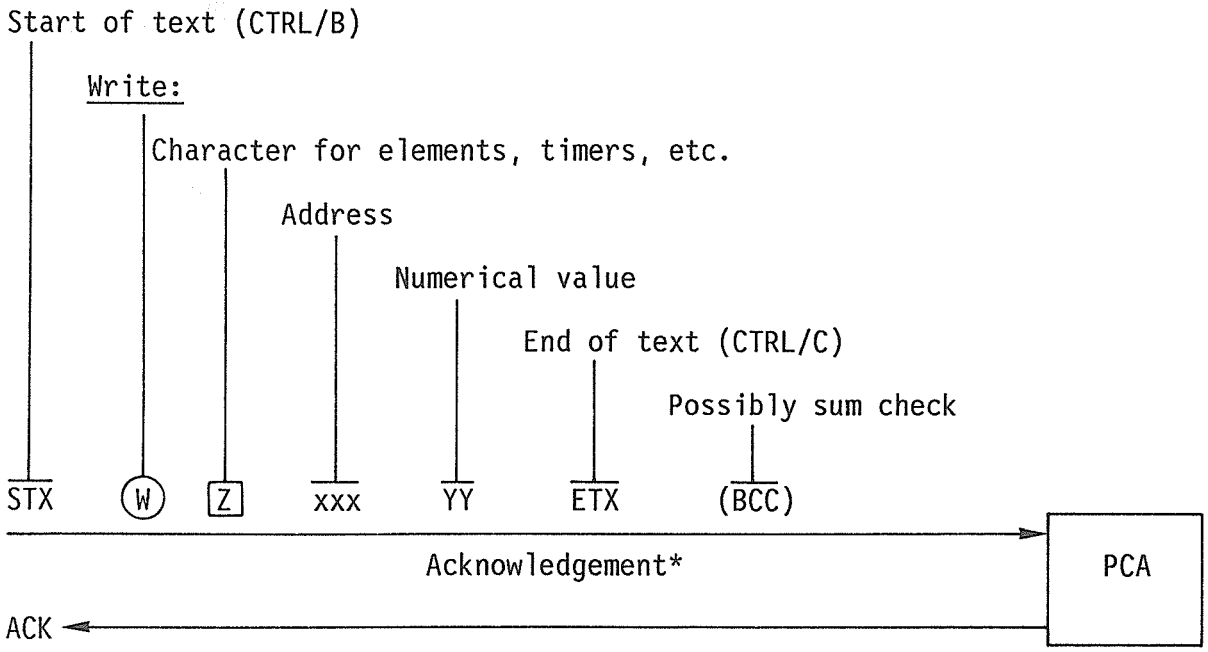
This is done in line 6:

CODE = Ø1 ---> with BCC, operation with "intelligent" unit according to DIN 66Ø19 ---> mode P1 (acknowledgement invisible)

CODE = ØØ ---> no BCC, operation with manually operated peripheral unit ---> mode P2 (acknowledgement visible)

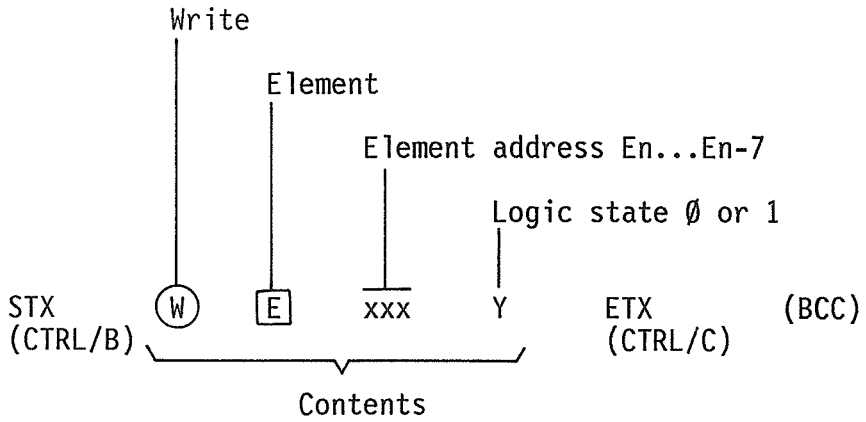
K 6.4.3 Writing of data to PCA

The telegram to be sent to the CPU is usually the following:



Positive reply by the CPU that the instruction was executed (also CR LF).
NAK (or # CR LF) if the instruction could not be executed.

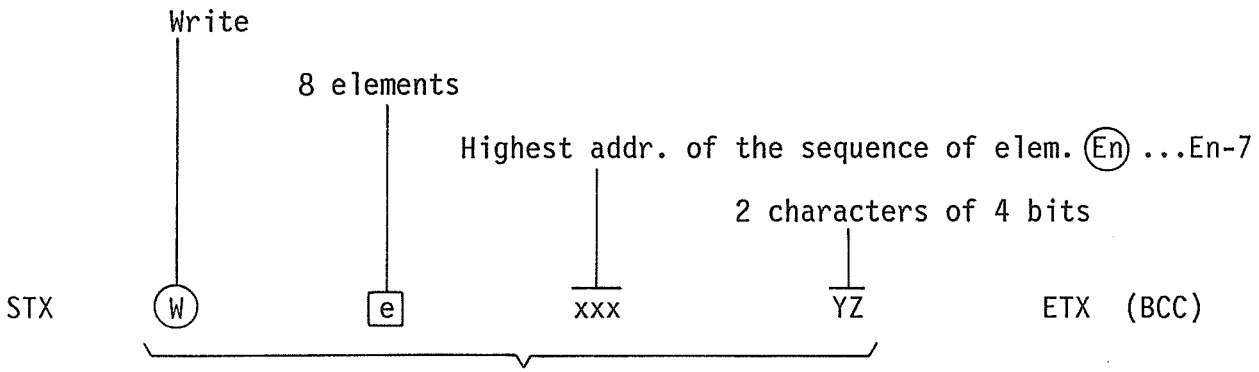
- Setting of individual elements En (E, A, T, C, M) with respect to their logic states (Ø or 1)



---> set element xxx (3 digits) to logic state Y (1 number Ø or 1)

*) Is given at once (without waiting for the clock pulse).

- Setting of 8 subsequent elements (A, M) with respect to their logic state (Ø or 1) by input of 2 characters of 4 bits



---> set the logic states of the sequence of elements, the highest address of which is xxx; the logic states are defined by the characters YZ.

(Y sets the elements En-4...En-7; Z sets the elements En...En-3)

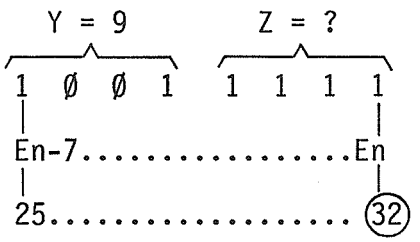
The following characters can be entered for Y and Z:

Character		Logic state of the elements			
	Y	En-7	En-6	En-5	En-4
	Z	En-3	En-2	En-1	En
Ø		Ø	Ø	Ø	Ø
1		Ø	Ø	Ø	1
2		Ø	Ø	1	Ø
3		Ø	Ø	1	1
4		Ø	1	Ø	Ø
5		Ø	1	Ø	1
6		Ø	1	1	Ø
7		Ø	1	1	1
8		1	Ø	Ø	Ø
9		1	Ø	Ø	1
:		1	Ø	1	Ø
;		1	Ø	1	1
<		1	1	Ø	Ø
=		1	1	Ø	1
>		1	1	1	Ø
?		1	1	1	1

Example:

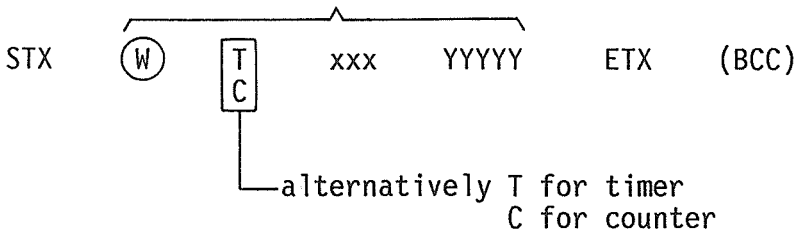
STX W e (32) 9? ETX

sets the outputs 25 ... (32) as follows:



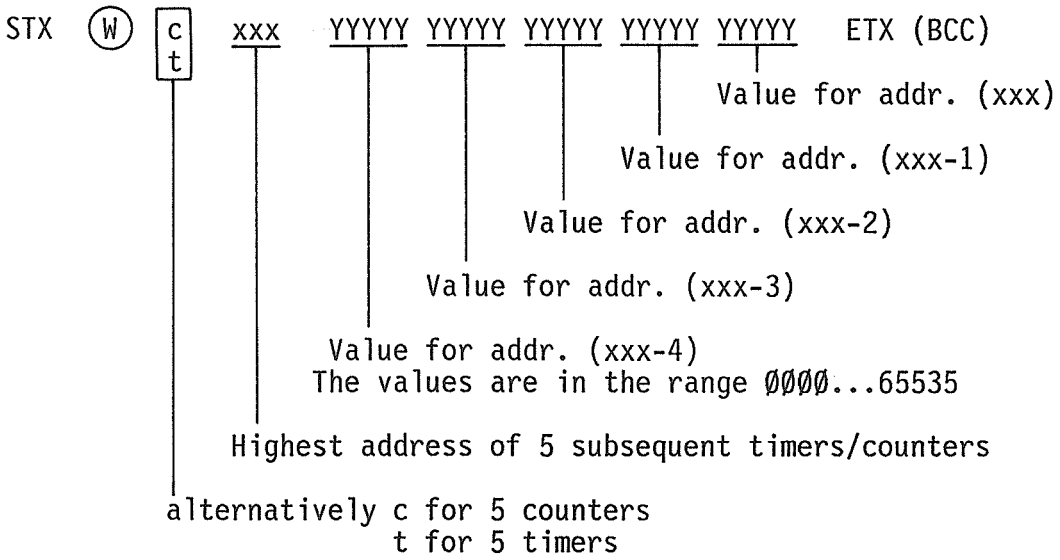
Compared to the processing of individual addresses this variant reduces the running time by the factor 7.

- Writing a value into the register of a timer or counter (Cn)

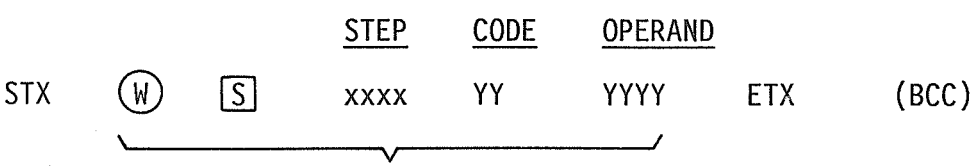


---> Write the value YYYYY (5 digits) into the timer/counter register with the address xxx (3 digits)

- Writing of 5 values to 5 subsequent counters or timers



- Writing a program line (STEP) into the RAM user memory



---> Write the value YYYYYY (6 digits) into RAM user memory (4 digits) at step xxxx (4 digits); 2 for the CODE Ø...31 and 4 for the OPERAND Ø...2Ø47)

- Writing a value into the display operand register

  By means of the bit processor instructions DOP and DTC, 4-digit numbers can be written into the operand register, which are displayed in the OPERAND field of the P1Ø programming unit. The register contents is displayed in the following second, for one second.

  STX (W) [O] YYYY ETX (BCC)

  ---> Write the value YYYY (4 digits) into the operand register

- Writing a 2-digit value (1 register word) into the word register Rn * (1 register word)

  STX (W) [R] xxx YY ETX (BCC)

  ---> Write the value YY (2 digits) into the word register with the address xxx (3 digits)

- Write a 1Ø-digit value (1 register block) into the word register * (5 register words)

  STX (W) [A] xxx YYYYYYYYYY ETX (BCC)

  ---> Write the value Y...Y (1Ø digits) into the word register, with xxx corresponding to the highest address of a register block.

- Writing a 3 x 1Ø-digit value (3 register blocks) into the word register * (15 register words)

  STX (W) [B] xxx Y...(3Øx)...Y ETX (BCC)

  ---> Write the value X...Y (3Ø digits) into the word register, with xxx corresponding to the highest address of 3 register blocks.

*) only PCA231/232

- Writing a 2-byte value to the text memory

STX (W) [M] xxxx YYYYY ETX (BCC)

YYYYY: 5-digit value ØØØØØ...65535

xxxx: Highest address of the text memory * (give odd address)

The value YYYYY is stored in binary notation in the text memory. The "H" byte is at the high text/memory address (xxxx), while the "L" byte is at the low text memory address (xxx-1).

---

*) The PCA2.M32 is known to have an 8K character text memory and an 8K-byte data memory which may also be used as a text memory. The address given here refers to the text memory, the addresses are therefore in the range Ø...15999, whereby the following characters must stand for 1Ø-thousands and 15-thousands.

1Ø : / 11 ; / 12 < / 13 = / 14 > / 15 ?

- Writing a 5 x 2 byte value to the text memory

STX (W) [m] xxxx YYYYY YYYYY YYYYY YYYYY YYYYY ETX (BCC)

YYYYY (last): Value for addresses (xxxx) and (xxxx-1)

YYYYY: Value for addresses (xxxx-2) and (xxxx-3)

YYYYY: Value for addresses (xxxx-4) and (xxxx-5)

YYYYY: Value for addresses (xxxx-6) and (xxxx-7)

YYYYY: Value for addresses (xxxx-8) and (xxxx-9)

xxxx: Highest address of 1Ø subsequent character numbers of the text memory (give odd address).

The individual values YYYYY are stored in the same way as described under telegram [M] and the information given under foot-note *) also holds true.

- Writing of 1Ø digits (1 register block) to the text memory

STX (W) [N] xxxx YYYYYYYYYY ETX (BCC)

Value for addr. (xxxx)
Value for addr. (xxxx-1)
Value for addr. (xxxx-2)
Value for addr. (xxxx-3)
Value for addr. (xxxx-4)

Highest address of 5 subsequent character numbers of the text memory ending with 4 or 9. *
(acc. to organisation of the register blocks)

*) see foot-note *) under telegram [M]

- Writing of 3Ø digits (3 register blocks) to the text memory

STX (W) [n] xxxx YY.....YY ETX (BCC)

3Ø digits starting with value for address (xxxx-14), ending with the value for address (xxxx)

Highest address of 15 subsequent character numbers of the text memory ending with 4 or 9. *
(acc. to organisation of the register blocks)

*) see foot-note *) under telegram [M]

Some of the telegrams listed here do not work with every CPU of the PCA-series.
Please refer to the summary in chapter K 6.4 " Definition of mode P " for the corresponding information.

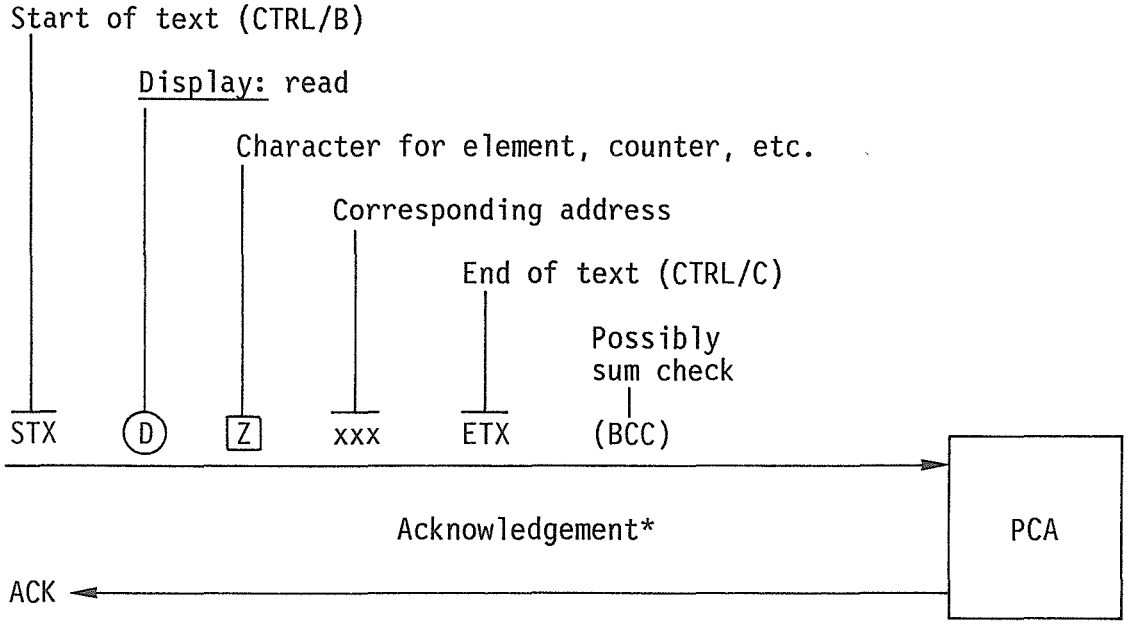
P.S.: Before the last character of a telegram is transmitted, a telegram can be declared invalid with ENQ (CTRL/E). The PCA then acknowledges with NAK (# CR LF). A new telegram can be transmitted afterwards.

K 6.4.4 Reading out data via the PCA

This kind of interrogation allows reading out values via the PC-values, such as: logic states of elements En, register contents of timers, counters, step contents in the user program or the display operand register.

This interrogation process is effected in two steps:

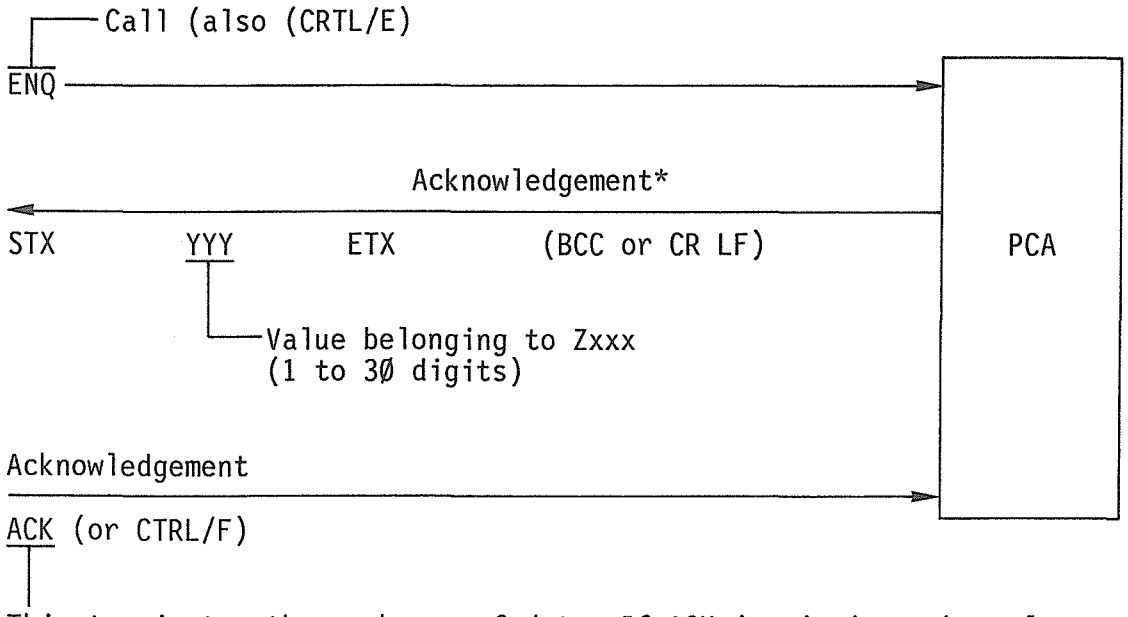
(1) Storage instruction to the CPU M32



Positive reply by the CPU that the instruction has been received (possibly also CR LF). NAK (or # CR LF) means that the data could not be stored.

*) Is given at once (without waiting for the clock pulse).

② Instruction to output the numerical value (YY..) belonging to the element previously entered (Zxxx).



This terminates the exchange of data. If ACK is missing, the telegram is repeated every 3Øth cycle by the PCA (provided that ØØØØ is in the 6th line of the operand). If the CPU receives a NAK (CTRL/U), the PCA repeats the above acknowledgement at once.

P.S.: If no address was stored before (step 1), the PCA replies to ENQ with NAK (or # CR LF).

---

*) Is given at once (without waiting for the clock pulse).

The various PLC-data is interrogated as follows:

- Reading the logic state of the individual elements En with address xxx

  En = (E, A, T, C, M)

  STX (D) [E] xxx ETX (BCC) , xxx = element address with 3 digits

- Reading of 8 subsequent elements (I, O, M) in the form of 2 characters of 4 bits each

  STX (D) [e] xxx ETX (BCC) , xxx = highest element address En (3 digits) of eight subsequent elements

  The PCA answers with

  STX YZ ETX (BCC or CR LF)

  Example of YZ

Character	Y	Z
Bit pattern with En = 32	Ø Ø 1 1 25	1 1 1 Ø 32
Output character	3	>

- Reading the register contents of timer or counter (Cn)

  STX (D) [T C] xxx ETX (BCC) , xxx = timer/counter address with 3 digits

- Reading of 5 subsequent counters/timers

  STX (D) [t c] xxx ETX (BCC) ; xxx = highest address of 5 subsequent timers/counters

- Reading the contents of a program line (STEP)

  STX (D) [S] xxxx ETX (BCC) , xxxx = step address of 4 digits (to 8191)

- Reading the display operand register (of DOP and DTC)

  STX (D) [O] ETX (BCC)

- Reading the contents of the register word (Rn) of 2 digits (1 register word)

STX (D) [R] xxx ETX (BCC) , xxx = register address with 3 digits

- Reading the register block (Rn) of 1Ø digits (5 register words)

STX (D) [A] xxx ETX (BCC) , xxx = highest register address of the register block

- Reading 3 register blocks of 3Ø digits (15 register words)

STX (D) [B] xxx ETX (BCC) ; xxx = highest register address of 3 register blocks

- Reading a 2-byte value from the text memory

STX (D) [M] xxxx ETX (BCC) ; xxxx = higher address of the text memory (give odd address)

- Reading 5 x 2-byte values from the text memory

STX (D) [m] xxxx ETX (BCC) ; xxxx = highest address of 1Ø subsequent character numbers of the text memory (give odd address)

- Reading 1Ø digits (1 register block) from the text memory

STX (D) [N] xxxx ETX (BCC) ; xxxx = highest address of 5 subsequent character numbers of the text memory ending with 4 or 9 (acc. to register block organisation)

- Reading of 3Ø digits (3 register blocks) from the text memory

STX (D) [n] xxxx ETX (BCC) ; xxxx = highest address of 15 subsequent character numbers of the text memory ending with 4 or 9 (acc. to register block organisation)

The data required in the reading telegrams are transmitted in the acknowledging telegram of the CPU using the same format as in the corresponding writing telegram.

Some of the telegrams listed here do not work with every CPU of the PCA-series. Please refer to the summary in chapter K 6.4 "Definition of mode P" for the corresponding information.

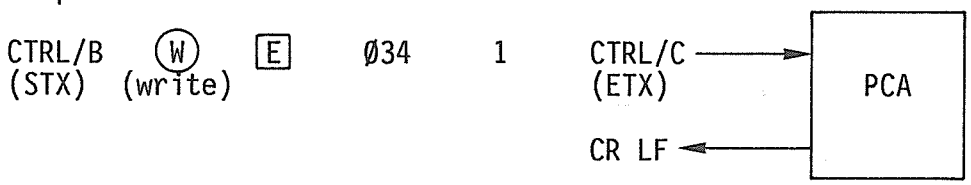
P.S.: A telegram may be cancelled with ENQ (CTRL/E) before the last character of a telegram is transmitted. The PCA acknowledges with NAK (# CR LF). A new telegram may then be transmitted.

Example:

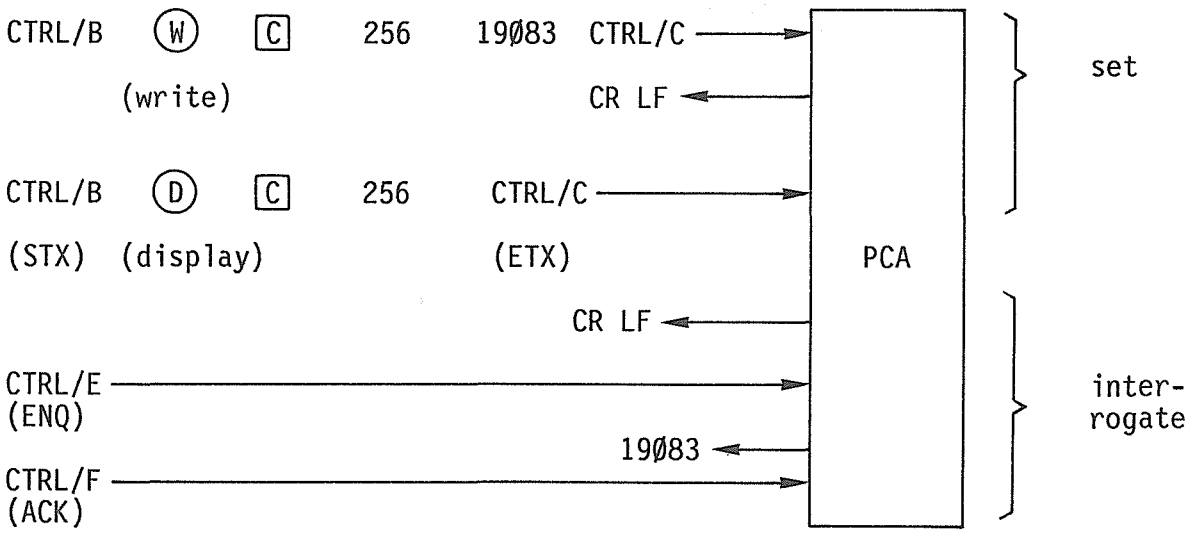
The following examples can be realized with a simple terminal by assigning the interface in mode P2 (code ØØ in line 6).

a) Setting an output via mode P

Output address: 34



b) Setting a counter and interrogation of this counter via mode P



K 6.5 Assignment of combined modes

In the following cases, only one interface assignment can be used when combining different modes for sending and receiving:

Menu mode (receiving of single characters and sending of text):
Mode C assignment

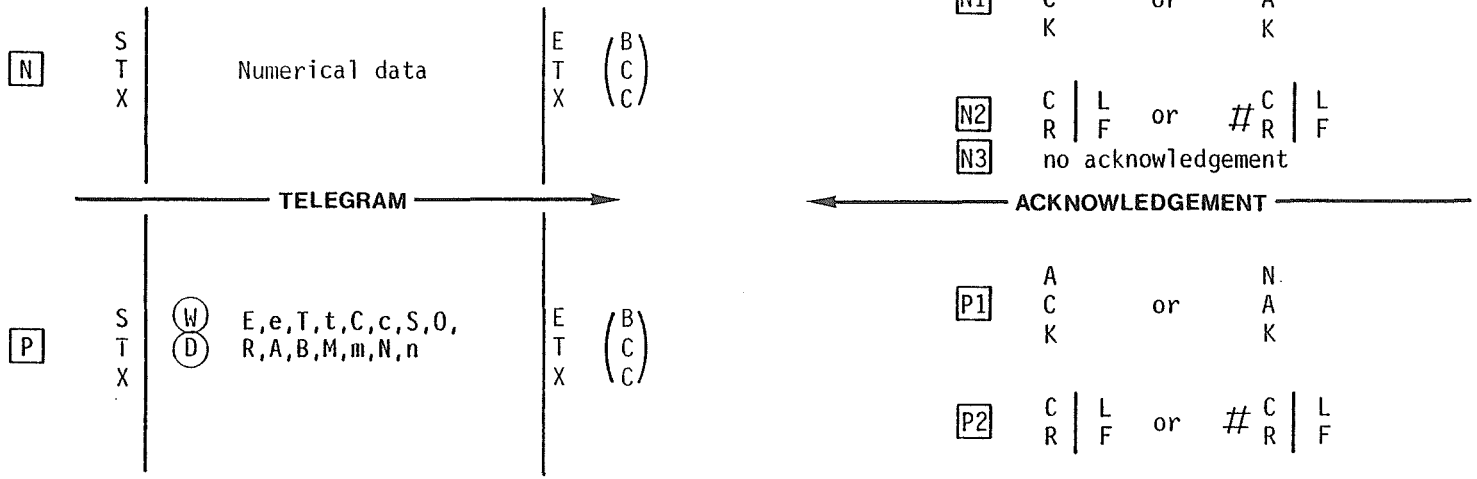
Computer mode (exchange of mumerical data and PCA-related data):
Mode N assignment

The following applies generally:

Assignment	Operation possible in
for text output	mode T
for single characters	modes C + T
for numerical data	modes N + T + P
for PCA data	modes P + T

Comment to the CODE:	Line nr	CODE	OP	Comment to the OPERAND:
	1:	PAS	1ØØ	Assignment of the serial data interface
	2:	ØØ	xxxx	BAUD, bits/character, parity, stop bit
[E] Text editor mode	3:	(Ø1)	TXB	Text busy Flag
Other operating modes		ØØ		
[N] Reception of 1...31x4 bit data telegram	4:	[Ø1...31]	RBY	Receive Buffer busy Flag
[C] Reception of single characters		[ØØ]		
[C] Transmission of 1...31x4 bit data telegram	5:	[Ø1...31]	TBY	Transmit Buffer busy Flag
[C] Transmission of single characters		[ØØ]		
[N1, P1, N3] With BCC for "intelligent" peripherals	6:	(Ø1)	Ø	Without acknowledgement the telegram is repeated
[N2, P2, N3] Without BCC for simple peripherals		ØØ	(1)	Telegram is transmitted only once
			(3)	Telegram is transmitted only once with no reply being expected
	7..1Ø:	ØØ	Ø	NUL

N and P are different only as regards their telegram contents:

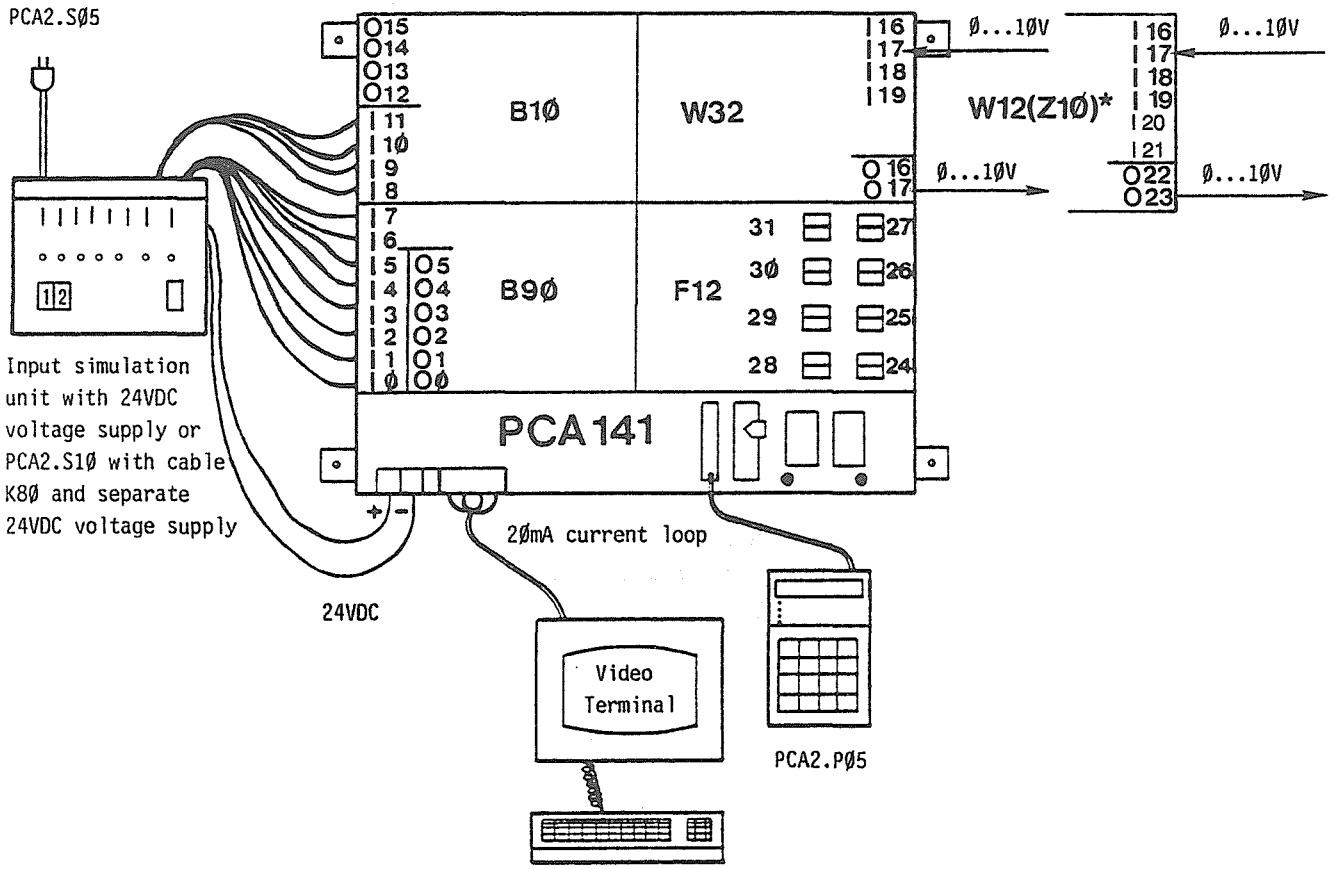


PART L PROGRAMMING EXAMPLES

PART L PROGRAMMING EXAMPLES

There is a wide range of possibilities, however, the following examples 1 through 11 use mainly typical training examples from the fields of text output, communication, PID-control as well as shift and FIFO register. Included are also typical programs for using date-time and check sum.

In order not to render the documentation confusing, all examples apply to be most commonly used PCA using the same I/O configuration as in the following figure.



All examples may be realized similarly with a PCA2 by setting a digital I-module and digital O-module to basic address O. It is recommended to use a PCA2.W14 as analog module at address 176.

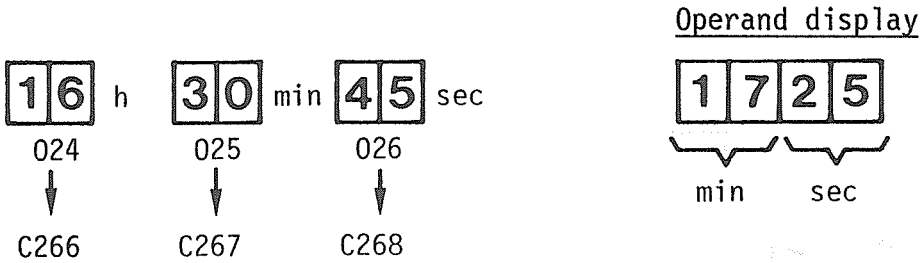
---

*) Module PCA1.W12 Z1Ø permits input voltages of Ø...1ØV (instead of Ø...5V).

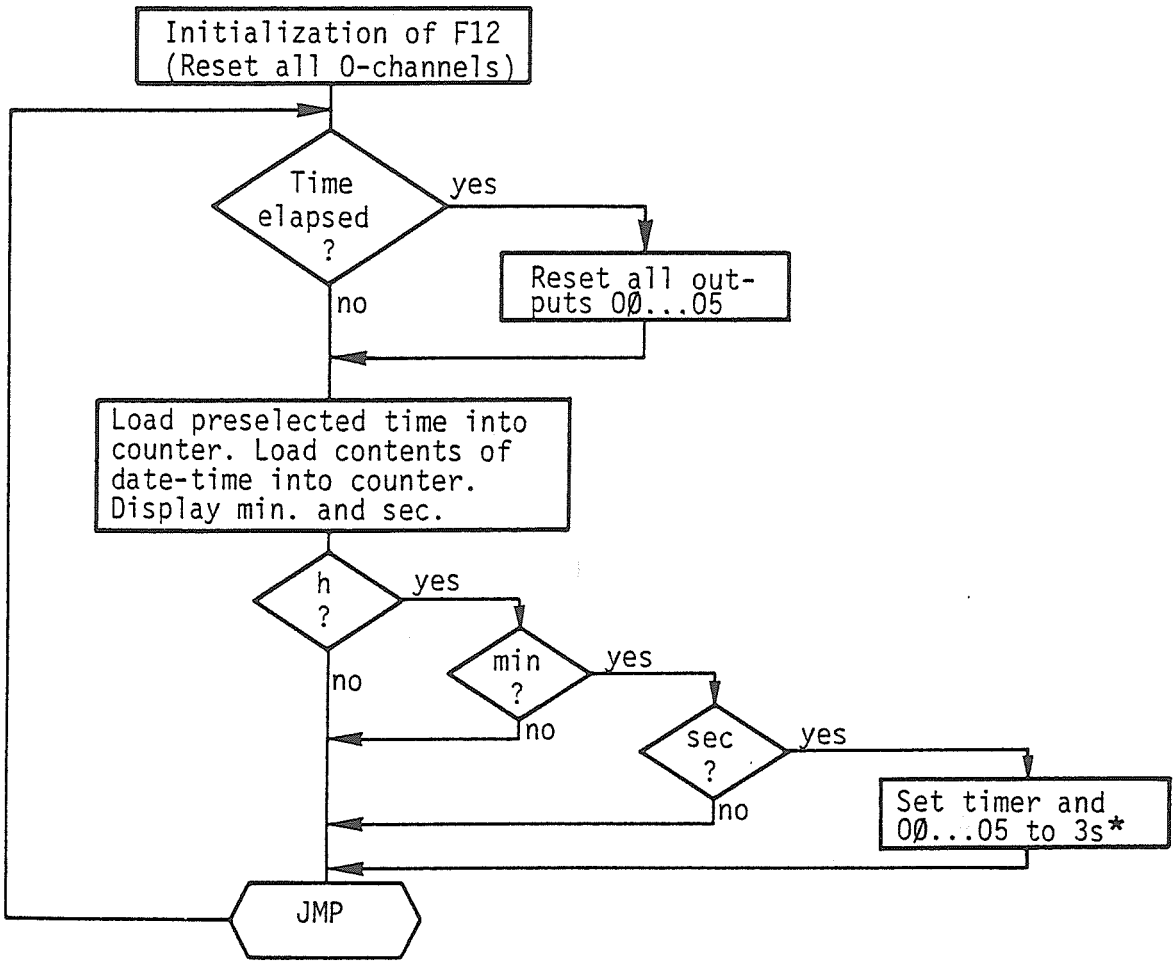
Example 1 Switching function and text output at a preselected time (using PAS 5Ø)

Problem 1a

The date-time must be set in the operating mode MAN. A date-time, at which the outputs 0Ø...05 will light up for 4s*, must be preselected via the two digits of the addresses 024...026 respectively. The minutes and seconds of the date-time must be displayed in the operand display.



Flowchart



*) The timing element is continuously reset during one second. In order to activate the outputs for 4s, the timing element must be set to 3s.

Solution 1a

```
360  16  SEI    0
361  12  REO  1024
362  27  INI     7
363  21  JIO   361

366  01  STH   279
367  21  JIO   374
368  16  SEI     0
369  12  REO  1000
370  27  INI     5
371  21  JIO   369

374  16  SEI     0
375  11  SEO  1024
376  15  SCR  1266
377  16   16    31
378  12  REO  1024
379  27  INI     2
380  21  JIO   375

383  29  PAS    50
384  15   15   270   h
385  29  PAS    50
386  16   16   271   min
387  29  PAS    50
388  17   17   272   sec

391  15  SCR  273
392  31   31  271
393  15  SCR  273
394  29   29  100
395  15  SCR  273
396  27   27  272
397  31  DTC  273

400  15  SCR  266
401  28   28  270
402  02  STL  266
403  22  JIZ  366

405  15  SCR  267
406  28   28  271
407  02  STL  267
408  22  JIZ  366

410  15  SCR  268
411  28   28  272
412  02  STL  268
413  22  JIZ  366

416  16  SEI     0
417  11  SEO  1000
418  14  STR   279
419  00   00    30
420  27  INI     5
421  21  JIO   417
422  20  JMP   366
```

Initialization of F12

Reset outputs 00...05 after the time has elapsed

Load digits 024...026 into counter C266...C268

Load date-time into C270...C272

Load minutes and seconds of the date-time into C273 and display

Comparison h

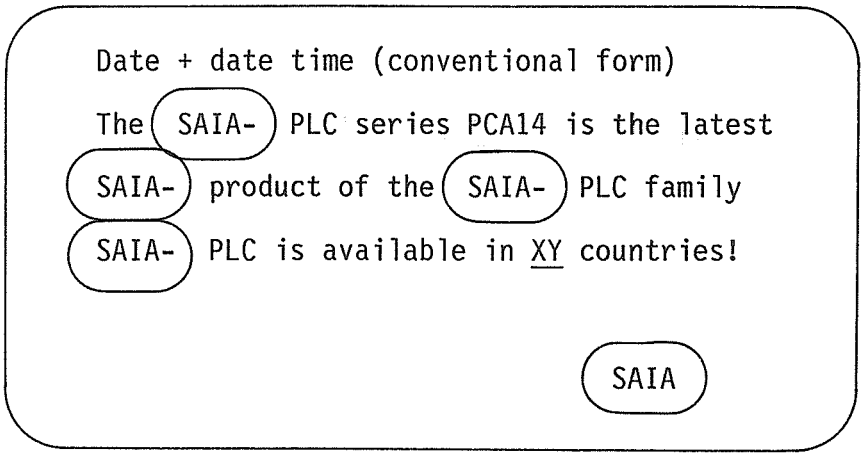
Comparison min.

Comparison sec.

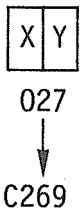
Set outputs 00...05 and timer to "4s"

Additional problem 1b

On correspondance of the date-time with the preselected time, the following additional text must be output:



- SAIA- must be written as a text subroutine.
- The value XY is evident from the BCD-switch 027 (without preceding zeros).



Solution 1b

```
431  29   PAS    100  }
432  00    00    902  }  4800 bauds
433  00    00    399  }
434  00    00    254  }
435  00    00    254  }  Interface
436  00    00      0  }  assignment
437  00    00      0  }
438  00    00      0  }
439  00    00      0  }
440  00    00      0  }

443  16   SEI      0  }
444  12   REO   1024  }  Initialization
445  27   INI      7  }  of F12
446  21   JIO    444  }

449  01   STH    286  }
450  21   JIO    457  }  Reset outputs
451  16   SEI      0  }  after the time
452  12   REO   1000  }  has elapsed
453  27   INI      5  }
454  21   JIO    452  }

457  16   SEI      0  }
458  11   SEO   1024  }  Load digits
459  15   SCR   1266  }  024...027 into
460  16    16     31  }  C266...269
461  12   REO   1024  }
462  27   INI      3  }
463  21   JIO    458  }

466  29   PAS     50  }
467  15    15    270  }
468  29   PAS     50  }  Load date-
469  16    16    271  }  time into
470  29   PAS     50  }  C270...C272
471  17    17    272  }

474  15   SCR    273  }
475  31    31    271  }
476  15   SCR    273  }  Load min. and
477  29    29    100  }  sec. into C273
478  15   SCR    273  }  and display
479  27    27    272  }
480  31   DTC    273  }

(1)

483  15   SCR    266  }
484  28    28    270  }  Comparison h
485  02   STL    266  }
486  22   JIZ    449  }

488  15   SCR    267  }
489  28    28    271  }  Comparison min.
490  02   STL    267  }
491  22   JIZ    449  }

493  15   SCR    268  }
494  28    28    272  }  Comparison sec.
495  02   STL    268  }
496  22   JIZ    449  }

499  16   SEI      0  }
500  11   SEO   1000  }  Set outputs
501  14   STR    286  }  00...05 and
502  00    00     40  }  timer to "4s"
503  27   INI      5  }
504  21   JIO    500  }

507  01   STH    287* }
508  21   JIO    449  }
509  29   PAS     23  }
510  00    00      5  }  Text output
511  14   STR    287* }
512  00    00     11  }
513  20   JMP    449  }
```

Text:

```
 5   .$. .0.2.0.$.H^M^J^J
 6   .$. .0.2.0.T.H.E. .$
 7   .L.0.2.3.P.L.C. .S.E
 8   .R.I.E.S. .P.C.A. .1
 9   .4. .I.S. .T.H.E. .N
10   .E.W.E.S.T^M^J.$. .0
11   .2.0.$.L.0.2.3.P.R.O
12   .D.U.C.T. .O.F. .T.H
13   .E. .$.L.0.2.3.P.L.C
14   . .F.A.M.I.L.Y..^M^J
15   .$. .0.2.0.$.L.0.2.3
16   .P.L.C. .I.S. .A.V.A
17   .I.L.A.B.L.E. .I.N.$
18   .c.2.6.9. .C.O.U.N.T
19   .R.I.E.S.!^M^J^J^M^M
20   ^M.$. .0.5.4^M^J^@^@

==> 23  .S.A.I.A.-.$.U^@^@^@
```

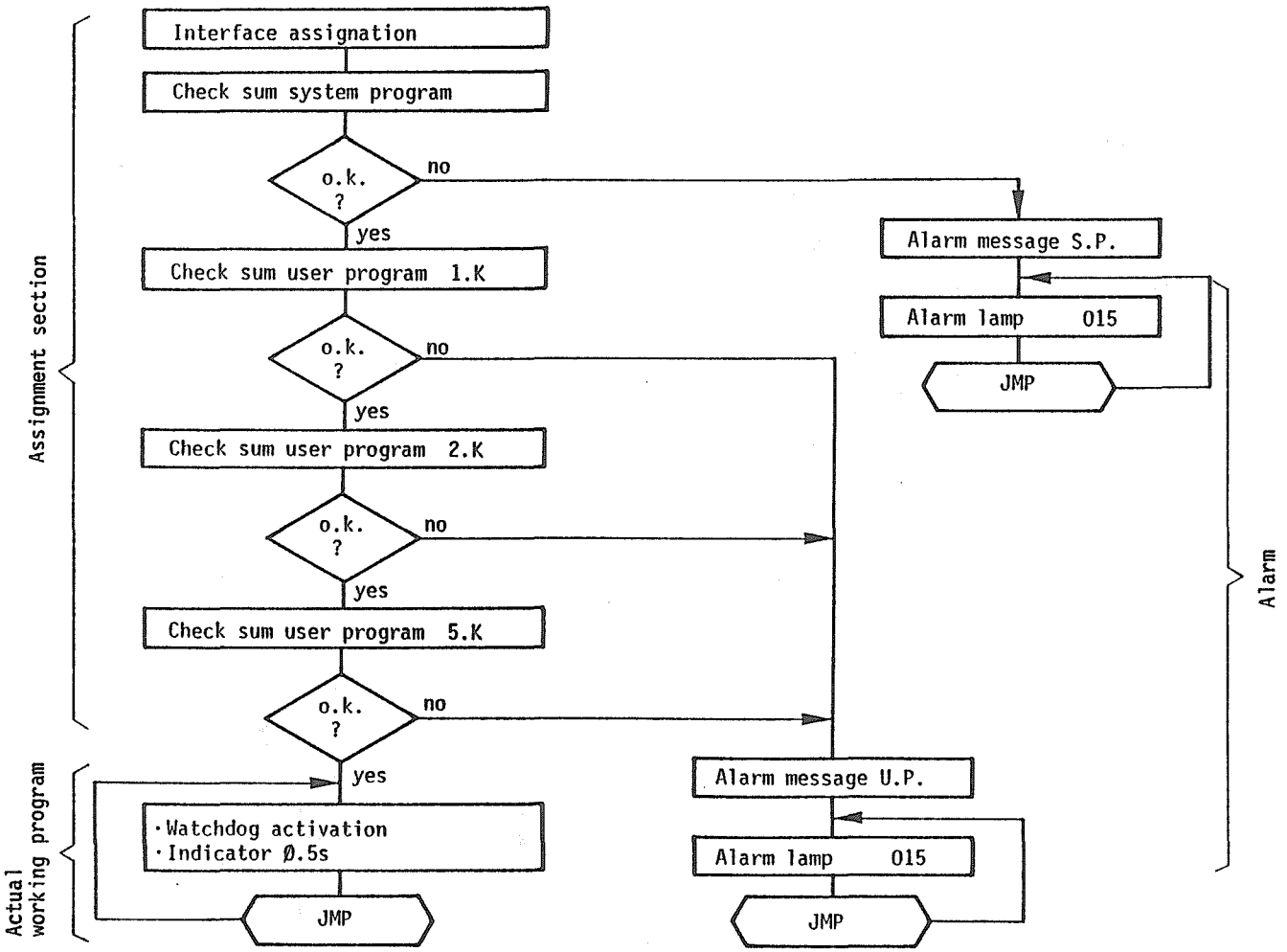
*) The text would be output several times during the second at which the conditions for the text output are fulfilled. Due to the timer (1.1s) the text is output only once.

Example 2 "Check sum" of the system and user program and activation of the "Watchdog" (using PAS 3Ø...38)

Problem 2

During the first run of this program a check sum for the system program as well as the user program (1.K, 2.K and 5.K)* is to be performed. The Watchdog must be activated and the output O12 must flash every Ø.5s in a small working program. If an error is detected in the system program, the message "check sum system program" must be delivered. If an error is detected in the user program the message "check sum user program" must be delivered. In both cases output O15 (alarm) must be activated.

Flowchart



*) The 1.K and the 2.K contain the user programs, the 5.K contains the text.

Errors in the system program cannot be simulated. In the RAM-memory of the user program, however, an input can be effected using vacant addresses in order to trigger an alarm message.

Solution 2

```
521  29  PAS   1ØØ
522  ØØ  ØØ    9Ø2
523  ØØ  ØØ    399
524  ØØ  ØØ    254
525  ØØ  ØØ    254
526  ØØ  ØØ      Ø
527  ØØ  ØØ      Ø
528  ØØ  ØØ      Ø
529  ØØ  ØØ      Ø
53Ø  ØØ  ØØ      Ø

532  29  PAS    3Ø
533  ØØ  ØØ      Ø
534  22  JIZ   554
535  29  PAS    31
536  ØØ  ØØ   1929
537  22  JIZ   559
538  29  PAS    32
539  Ø3  Ø3   1397
54Ø  22  JIZ   559
541  29  PAS    35
542  ØØ  ØØ   1815
543  22  JIZ   559
544  12  REO    15  ;

546  13  COO   255
547  Ø2  STL   285
548  14  STR   285
549  ØØ  ØØ      5
55Ø  13  COO    12
551  2Ø  JMP   546

554  29  PAS    23
555  ØØ  ØØ    1ØØ
556  11  SEO    15
557  2Ø  JMP   556

559  29  PAS    23
56Ø  ØØ  ØØ    1Ø4
561  11  SEO    15
562  2Ø  JMP   -561
```

48ØØ bauds

Interface assignment

Check sum system program

Check sum 1.K

Check sum 2.K

Check sum 5.K

Reset alarm

Watchdog

Working program

Output of texts and alarm output in case of an error in the system program

Output of texts and alarm output in case of an error in the user program

Explanation:
The test via the check-sum is usually only performed when switching on the PLC. It is therefore also part of the assignment section of the program (lines 521 to 544).

The actual working program ranges from 546 to 551. The other program parts are used for text output in case of errors.

Text 1ØØ:

```
100  .$.H^M^J^J.C.H.E.C.K
101  .S.U.M. .S.Y.S.T.E.M
102  . .P.R.O.G.R.A.M^M^M
103  ^M^M^M^M^M^M^J^J^@^@
```

Text 1Ø4:

```
104  .$.H^M^J^J.C.H.E.C.K
105  .S.U.M. .U.S.E.R. .P
106  .R.O.G.R.A.M^M^M^M^M
107  ^M^M^M^M^M^M^M^M^J^J
108  ^@.*.U.*.*.U.*.*.U.*
```

*) Remaining in the alarm loop is deliberate. If the check sum has a negative result, either the CPU or a user memory must be replaced. Further safety measures can be taken, while the WD is active.

Example 3 Special text characters

Problem 3

All special text characters must be displayed on a screen. These are:

a) Normal text

b) Output of a counter register Cn, without suppression of preceding zeroes

c) Output of a counter register Cn, with suppression of preceding zeroes

d) Output of elements (logic states of 8 subsequent elements)

e) Output of the ASCII-character* (formed by 8 subsequent elements)

f) Contents of date-time, industrial form

g) Contents of date-time, conventional form

h) Contents of date-time, year, month, day

i) Repetition of characters

k) Text subroutine

l) Output of character $

---

*) All characters in the range of Ø...127 are evaluated. The visible characters are displayed, while the control characters have a direct effect and are replaced by SPACE in the display.

```
SAIA AG              a)              MURTEN

                   $C                $c
                b) 00016          c)   16

                   $E                $e
             d) 01011001           e)  Y


   $T                      $H                        $D
f) :47:04/17:01:46    g) 83-11-24/17:01:46      h) 83-11-24

   $-
i) ----------------------------------------------------
     $L, $U
k) THE COMPACT (PCA14) FOR COMPLEX FUNCTIONS:
   WITH THE (PCA14) YOUR CONTROL IS MORE EFFICIENT!
```

Solution 3

(The parameters as well as the various cursor instructions refer to the terminal VC 4404).

```
951  29   PAS   100
952  00    00   902 ; 4800 bauds
953  00    00   333 ; TXB
954  00    00   254
955  00    00   254
956  00    00     0
957  00    00     0
958  00    00     0
959  00    00     0
960  00    00     0

962  29   PAS    23
963  00    00     0 ; Text no. 0

966  25   WIH   333 ; TXB
967  29   PAS    23
968  00    00    50 ; Text no. 50

971  15   SCR   290
972  00    00     0
973  25   WIH   333 ; TXB
974  14   STR   259
975  00    00    10
976  29   PAS    23
977  00    00    30 ; Text no. 30
978  25   WIH   259
979  17   INC   290
980  20   JMP   973
```

Text:

```
→0   ^X^@^@^@^@^@^@^@^@^@
→30  ^P.#.*.$.C.2.9.0.$.
 31  .0.2.0.$.c.2.9.0^M^P
 32  .(.(.$.E.0.0.7.$. .0
 33  .2.0. ^H.$.e.0.0.7^M
 34  ^P.+. .$.T.$. .0.0.5
 35  .$.H.$. .0.0.5.$.D^J
 36  ^J^J^J^J^J^J^J^J^M^@
→50  .$. .0.0.7.S.A.I.A.
 51  .A.G.$. .0.2.0. . .
 52  .M.U.R.T.E.N. . ^M^J
 53  ^J.$. .0.1.1.$.$.C.$
 54  . .0.2.3.$.$.c^M^J^J
 55  ^J^J^J.$. .0.1.1.$.$
 56  .E.$. .0.2.3.$.$.e^M
 57  ^J^J^J.$.$.T.$. .0.1
 58  .8.$.$.H.$. .0.2.0.$
 59  .$.D^M^J^J^J.$.$.-^M
 60  ^J.$.-.0.5.0^M^J^J.$
 61  . .0.0.4.$.$.L.,.$.$
 62  .U^M^J.$. .0.0.4.T.H
 63  .E. .C.O.M.P.A.C.T.
 64  .$.L.0.8.0. .F.O.R.
 65  .C.O.M.P.L.E.X. .F.U
 66  .N.C.T.I.O.N.S.:^M^M
 67  ^M^J.$. .0.0.5.W.I.T
 68  .H. .T.H.E. .$.L.0.8
 69  .0. .Y.O.U.R. .C.O.N
 70  .T.R.O.L. .I.S. .M.O
 71  .R.E. .E.F.F.I.C.I.E
 72  .N.T.! ^M^J^J^@^@.U.*
⇒80  ^N.P.C.A.1.4^O.$.U^@
```

Clear screen

Contents of counter register, sequence of elements and date-time

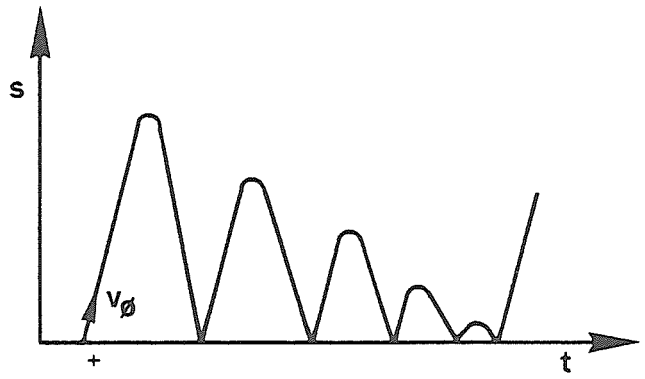
Basic text

Text subroutine

Example 4 Ball bounce

Given:

A ball which has a starting speed of $V_{ø}$ is thrown up vertically. Every time it falls on a horizontal surface it loses X % of its velocity.



$$s = V_{ø} \times t - \frac{g}{2} \times t^2$$

$$g = 1ø \text{ m/s}^2$$

$$t = t/1ø \text{ (1/1ø s)}$$

$$\Longrightarrow s = \frac{v_{ø} \times t}{1ø} - \frac{1ø \; t^2/1øø}{2} \text{ [m]}$$

$$\boxed{s = 1/1ø \; (v_{ø} \times t - t^2/2)} \text{ [m]}$$

Problem 4

Every 1/1øs the height s must be calculated and displayed on the screen.

Every time it drops, $v_{ø}$ is reduced to the proportional value preselected via 024 (of F12).

8	6

% of $v_{ø}$

024

C313

Solution 4

$v_{ø}$ is read into C314.

The formula $v_{ø}$ x $t^2/2$ is calculated within the counter register.

The text is astonishingly short due to the use of special characters.
Only 16 characters!

```
651  29  PAS   1ØØ
652  ØØ   ØØ   9Ø2  ; 48ØØ bauds
653  ØØ   ØØ   6ØØ  ; TXB
654  ØØ   ØØ   254  ; RBY
655  ØØ   ØØ   254  ; TBY
656  ØØ   ØØ     Ø
657  ØØ   ØØ     Ø
658  ØØ   ØØ     Ø
659  ØØ   ØØ     Ø
66Ø  ØØ   ØØ     Ø
661  11  SEO    24          } Read in the
662  15  SCR   313          } amplitude of
663  16   16    31          } Ø24 in %
664  12  REO    24
665  15  SCR   314  <-
666  ØØ   ØØ    11  vø
667  15  SCR   315
668  ØØ   ØØ     Ø  t = Ø
669  15  SCR   317  <-
67Ø  31   31   314  v
671  15  SCR   317  v      } vxt
672  29   29   315  t      }
673  15  SCR   316  <-
674  31   31   315  t
675  15  SCR   316  t      } t²
676  29   29   316  t      }
677  15  SCR   316  t²     } t²/2
678  3Ø   3Ø     2  2      }
679  15  SCR   317  vxt    } vxt - t²/2 = s
68Ø  28   28   316  t²/2   }
681  15  SCR   318  <-
682  31   31   317  s
683  17  INC   315  t + 1
684  21  JIO   692
685  15  SCR   314  v      } vx%
686  29   29   313  %      }
687  15  SCR   314  vx%    } vx%
688  3Ø   3Ø   1ØØ  1ØØ    } 1ØØ
689  Ø1  STH   314          } v = Ø ?
69Ø  21  JIO   667          }
691  2Ø  JMP   661  v = Ø --> New start
692  29  PAS    23          } Text no.3
693  ØØ   ØØ     3          }
694  25  WIH   6ØØ  ; Wait as long as
695  ØØ  NOP     Ø    TXB = H
696  2Ø  JMP   669
```

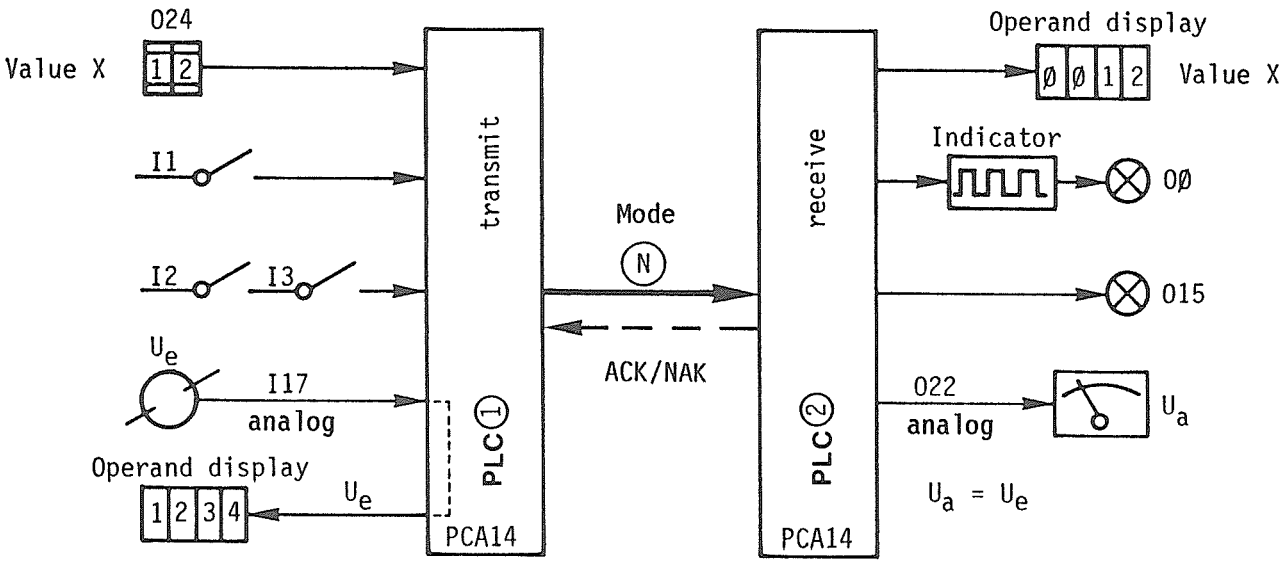
Text: Output C-register 318

```
3   .$.c.3.1.8. .:.$.-.3
4   .1.8.*^M^J^@^@^@^@^@
```

Example 5 Data transfer between two PCA14 in mode N via the serial data interface

Problem 5

Various inputs made on the 1st PLC must be transferred in mode N via the serial data interface, processed by the 2nd PLC and output. These are the following:



Problem for PLC ① (transmitter)

a) Define PAS 1ØØ starting with address 6Ø1

- TXB = 254
- RBY = 525
- TBY = 425

b) Establish the read-in program according to the above configuration. The analog value V_{in} entered via input channel I17 must be displayed as binary value in the operand display.

c) Send a telegram.

d) The current loop must be supplied by PLC ① (PLC ① is active).

Problem for PLC (2) (receiver)

a) Define PAS 1ØØ starting with address 6Ø1

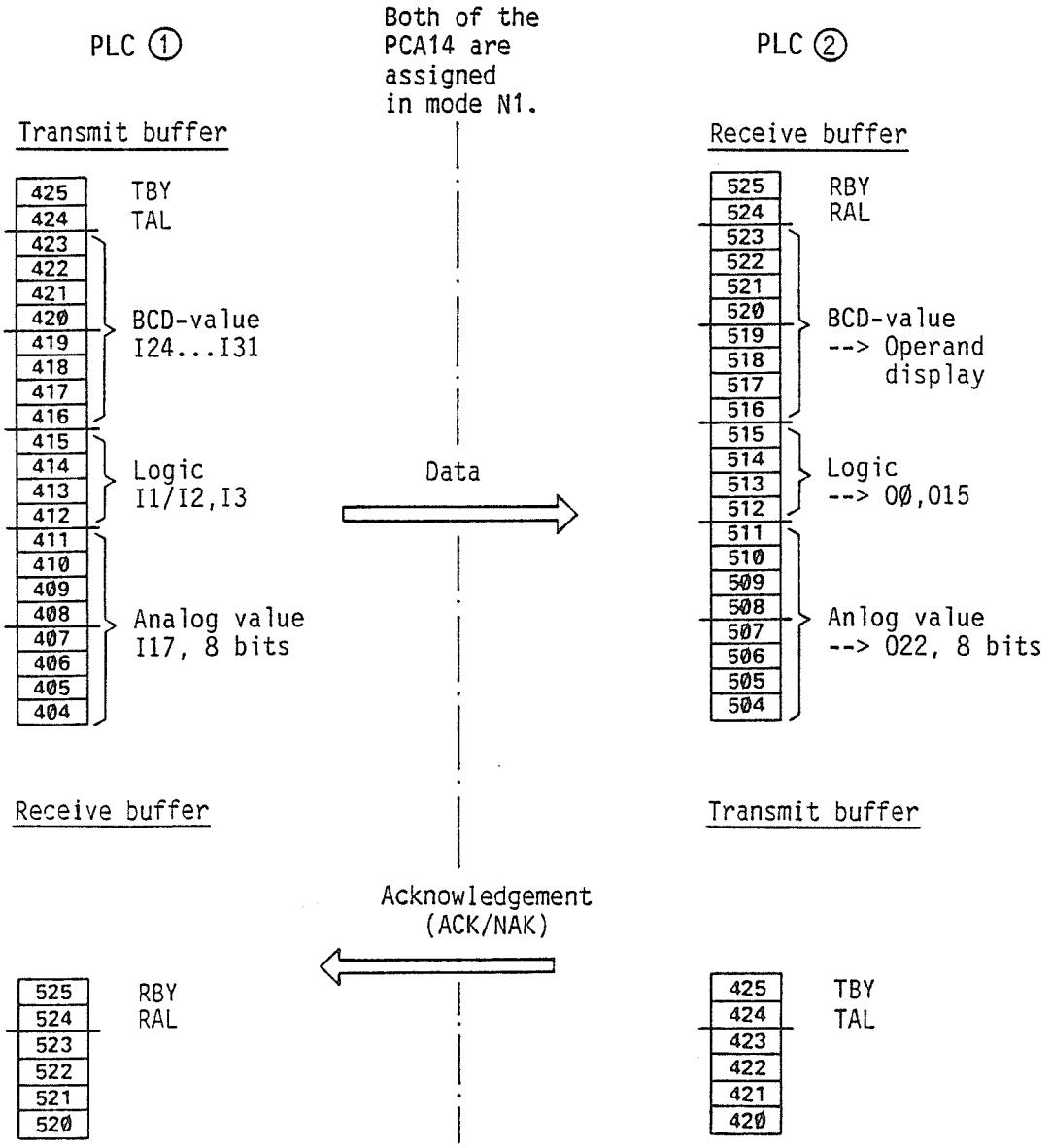
- TXB = 254
- RBY = 525
- TBY = 425

b) Establish the processing program according to the above configuration.
Output OØ must blink according to the BCD-value X.

c) Enable reception.

d) PLC (2) must be passive.

Solution 5



PLC (1)

Transmitter

```
601  29  PAS   100
602  00   00   902 ;4800 baud
603  00   00   254 ;TXB
604  01   01   525 ;RBY
605  05   05   425 ;TBY
606  01   01     0 ;BCC
607  00   00     0
608  00   00     0
609  00   00     0
610  00   00     0

613  01  STH   425 } Transmitter
614  21  JIO   639 } clear?

616  11  SEO    24 }
617  15  SCR   299 } Transmit BCD-
618  24   24    31 } value of 024
619  12  REO    24 } to transmit
620  15  SCR   299 } buffer
621  21   21   423 }

623  01  STH     1 }
624  10  OUT   412 } Transmit logic
625  01  STH     2 } results to trans-
626  03  ANH     3 } mit buffer
627  10  OUT   413 }

629  19  SEA     0 }
630  10  OUT    17 } Read in analog
631  15  SCR   298 } value of channel
632  24   24    23 } I17 and transmit
633  15  SCR   298 } to transmit
634  21   21   411 } buffer

636  19  SEA     0 } Send
637  11  SEO   425 } telegram

639  31  DTC   298 ;Display analog
640  20  JMP   613  value
```

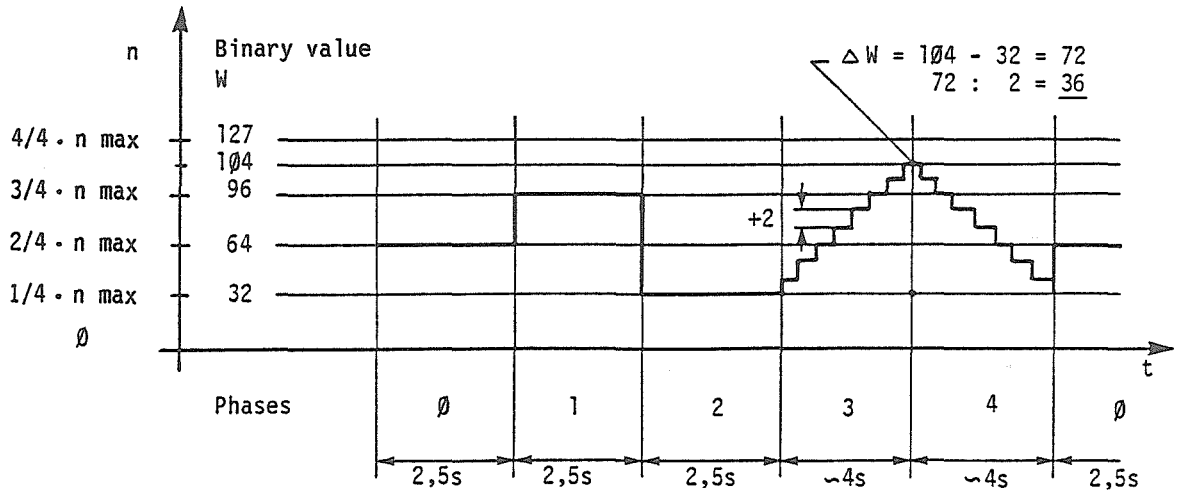
PLC (2)

Receiver

```
601  29  PAS   100
602  00   00   902 ;4800 baud
603  00   00   254 ;TXB
604  05   05   525 ;RBY
605  01   01   425 ;TBY
606  01   01     0 ;BCC
607  00   00     0
608  00   00     0
609  00   00     0
610  00   00     0

613  01  STH   525 } Receiver
614  22  JIZ   633 } clear?

616  12  REO   525 ;End of reception

618  15  SCR   257 } BCD-value
619  16   16   523 } to operand
620  31  DTC   257 } display

622  15  SCR   258 }
623  24   24   511 }
624  15  SCR   258 } Transmit
625  21   21    23 } analog value to
626  11  SEO    23 } channel 022
627  05  ORH    22 }
628  12  REO    23 }

630  01  STH   513 } Output 015
631  10  OUT    15 }

633  02  STL   512 }
634  11  SEO   400 }
635  01  STH   512 }
636  04  ANL   256 } Indicator
637  14  STR   256 } of 00
638  16   16   523 } (t = BCD-
639  13  COO   400 } value)
640  01  STH   400 }
641  10  OUT     0 }
642  20  JMP   613 }
```

Example 6 Fast PID-control loop by means of PAS 2Ø2 on PCA14 (motor tachometer generator)

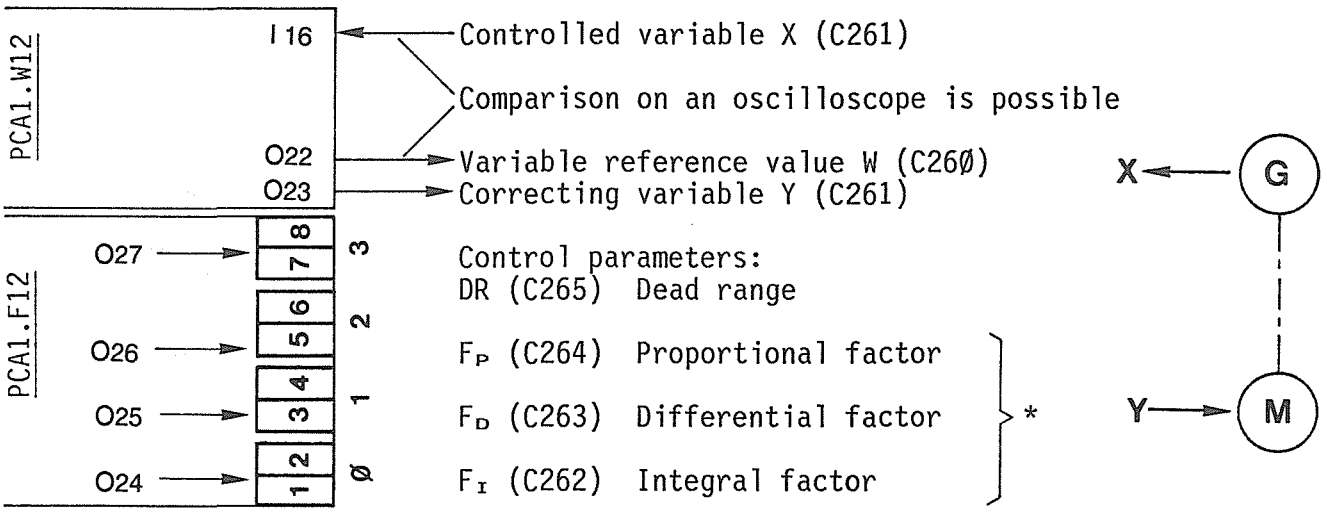
A DC-motor with tachometer generator must be controlled according to a variable reference value curve W = f (t). This reference value curve has the following shape:



The five phases Ø...4 can be determined according to the following program:

```
1ØØ   STR   281   ;  Phase Ø
      ØØ     25   }
      SCR   26Ø   }  W = 1/2 n max
      ØØ     64   }
  ┌─>JMS   15Ø --> PID
  │   STH   281
  └──JIO   1Ø4

11Ø   STR   281   ;  Phase 1
      ØØ     25   }
      SCR   26Ø   }  W = 3/4 n max
      ØØ     96   }
  ┌─>JMS   15Ø --> PID
  │   STH   281
  └──JIO   114

12Ø   STR   281   ;  Phase 2
      ØØ     25   }
      SCR   26Ø   }  W = 1/4 n max
      ØØ     32   }
  ┌─>JMS   15Ø --> PID
  │   STH   281
  └──JIO   124

13Ø   SCR   3ØØ   ;  Phase 3
      ØØ     36

  ┌─>INC   26Ø   }  W + 2
  │   INC   26Ø   }
36x   JMS   15Ø --> PID
  │   DEC   3ØØ
  │   STH   3ØØ
  └──JIO   132

14Ø   SCR   3ØØ   ;  Phase 4
      ØØ     36   }
  ┌─>DEC   26Ø   }  W - 2
  │   DEC   26Ø   }
36x   JMS   15Ø --> PID
  │   DEC   3ØØ
  │   STH   3ØØ
  └──JIO   142

      JMP   1ØØ -->
```

The control parameters can be entered or continuously adapted by means of the preselection module PCA1.F12. The motor tachometer generator is connected to the analog module PCA1.W12 as follows:



The PID-subroutine 15Ø is therefore:

```
--> 15Ø   SEI      Ø
          SEO    1Ø24    FI
          SCR    1262    FD   } *
           17*     31     FP
          REO    1Ø24
          INI       2          Read BCD-switch O24...O27
          JIO     151          into counter C262...265
          SEO      27
          SCR     265    DR
           16      31
          REO      27

   17Ø    WIH     28Ø         } Wait for sampling period
          OUT      16         } Read in controlled variable X
          SCR     261         } via I-channel 16
           24      23         } (into C261)

   174    PAS     2Ø2         Data block 31
     5    ØØ       31         26Ø, 261, 262, 263, 264, 265
     6    ØØ      26Ø
     7    ØØ        Ø
     8    ØØ        Ø
     9    ØØ        Ø
   18Ø    ØØ        Ø
     1    ØØ        Ø
     2    ØØ        Ø
     3    ØØ        Ø
     4    JMP     185   **

   185    SCR     261         } Output correcting variable Y at
           21      23         } O-channel 23
          SEO      23         }
          ORH      23         }
          REO      23         }

   19Ø    SCR     26Ø         } Output variable reference value W
           21      23         } at O-channel 22
          SEO      23         }
          ORH      22         }
          REO      23         }

   195    STR     28Ø         } Sampling period
           ØØ       1         } Tø = Ø.1 sec
   197    RET       Ø  ---------->
```

*) The BCD-preselection values are multiplied by 1Ø and read into the respective counter. If the preselection value is 5Ø, the practical value for F_I, F_D, F_P is calculated as follows:
5Ø x 1Ø = 5ØØ / 5ØØ : 256 = 1.95

**) JMP allows a PP-change.

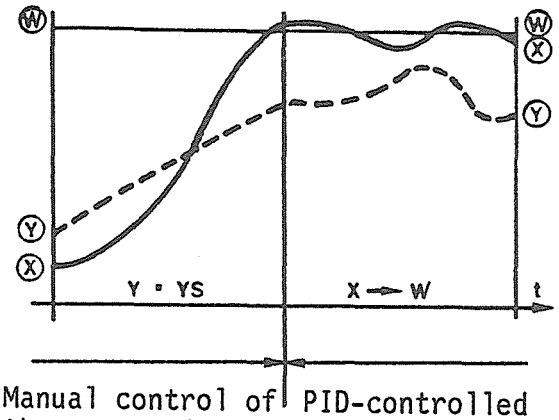
Example 7 Manual PID-operation using PAS 211 on PCA14
(analog modules 12 bits PCA1.W32)

Problem 7

The following program shows the transition from direct manual control of the correcting variable to automatic PID-control. The input and output routines from the documentation of the analog module PCA1.W32 are used as sub-routines.

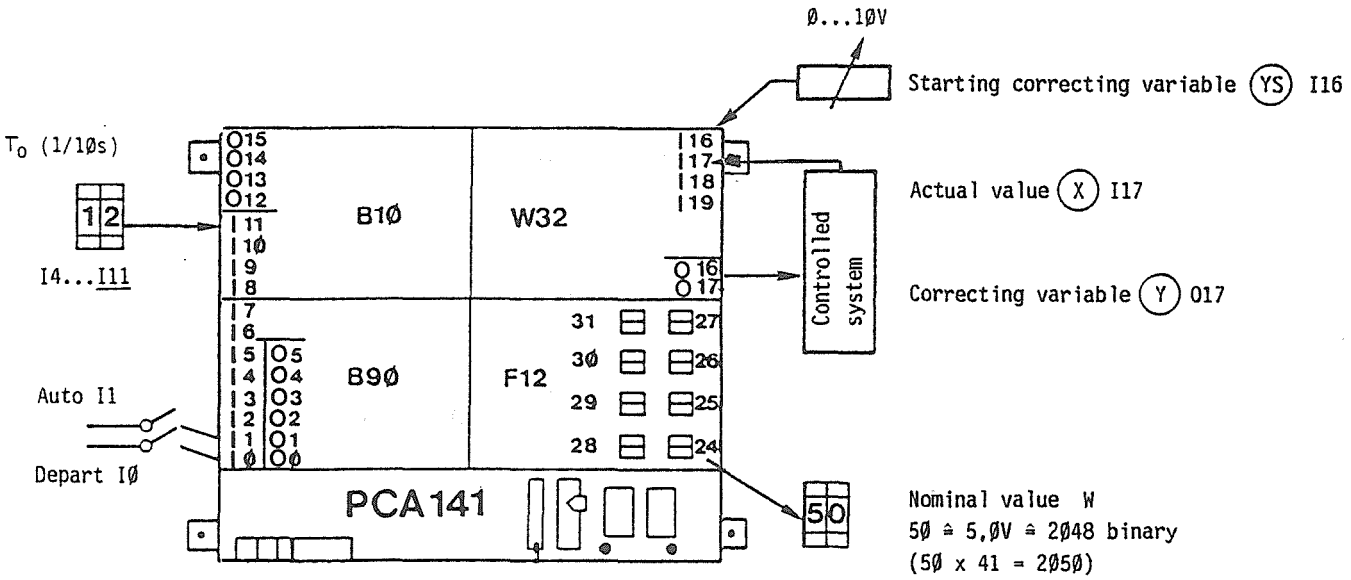
The nominal value W is preselected in 0.1V on BCD-switch 024. The sampling time T_o can be set in Ø.1 sec on BCD-switch I4...11. The process is started with I0.

If I1 = L, the correcting variable Y can be modified manually (potentiometer) via analog input I16, until X = W.
If I1 = H, PID-control is activated, causing automatic control of disturbance quantities X ---> W.

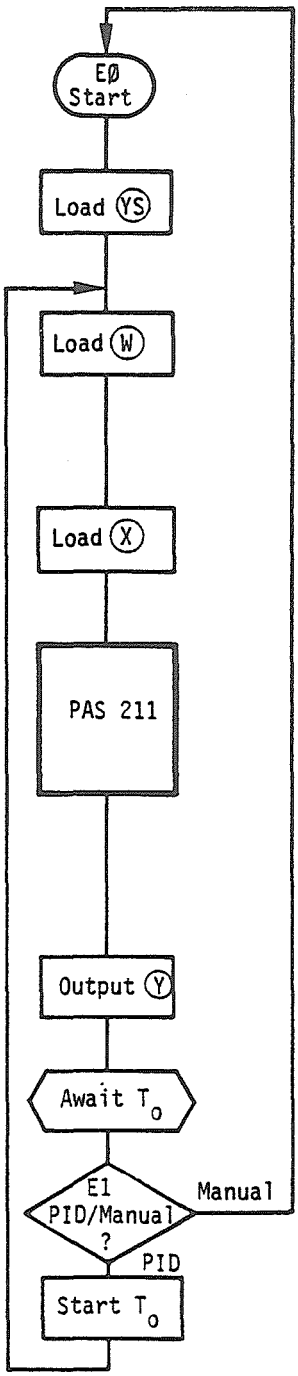


Manual control of PID-controlled the correcting variable (Y)

Hardware configuration



Solution 7




```
5Ø   WIL    Ø      Start

51   JMS    55Ø**  ⎫ Read in correcting variable (YS)
52   SCR    282    ⎬ via analog channel I16 and trans-
53    31    3ØØ    ⎭ fer to C282

54   SEO     24    ⎫
55   SCR    276    ⎪ Load nominal value
56    16     31    ⎬ (W) of BCD-switch 024
57   REO     24    ⎪ into C276
58   SCR    276    ⎪ * 41, i.e. convert to
59    29     41    ⎭ binary value

6Ø   JMS    5ØØ*   ⎫ Read in actual value (X) via
61   SCR    277    ⎬ analog channel I17 and trans-
62    31    3ØØ    ⎭ fer to C277

63   PAS    211
      12     31     12 bits, data block 31
     ØØ     276     (W)
     ØØ     277     (X) / (Y)
       FI
       FD
       FP
       DR
     ØØ     282     (YS)
     ØØ       Ø

73   SCR    3Ø1    ⎫ Output correcting
74    31    277    ⎬ variable (Y) to
75   JMS    7ØØ*   ⎭ analog channel Ø17

76   WIH    256    Wait for sampling time TØ

77   STH      1    ⎫ Manual
78   JIZ     5Ø    ⎭ PID interrogation

79   STR    256    ⎫ Load sampling time TØ
8Ø    16     11    ⎭ from BCD-switch and start

81   JMP     54
```

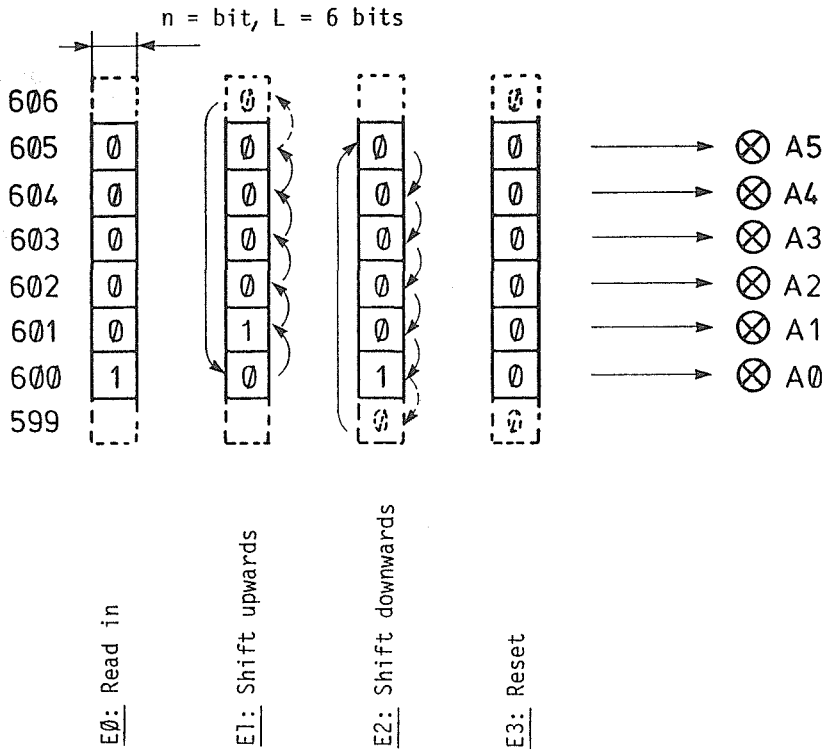
As explained in the problem description this program allows placing the actual value manually near the nominal value (W) by changing potentiometer (YS) (I16). I1 is then used to switch over to PID-control smoothly.

---

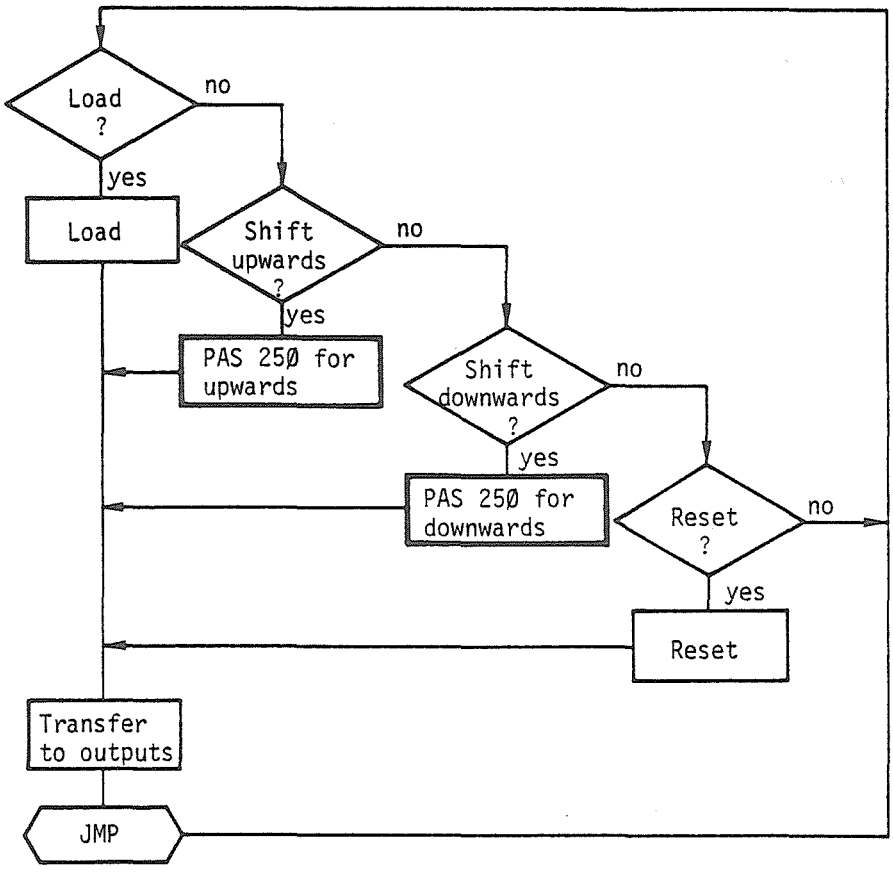
*) For subroutines refer to chapter "PCA1.W3.." in Hardware manual PCA1.
**) Subroutine 55Ø corresponds to subroutine 5ØØ by replacing the second line of the latter by OUT 16.

Example 8 1-bit rotation register upwards/downwards with reset (using PAS 25Ø)

Functional description:



Flowchart:



Solution 8

```
1001  01   STH     0
1002  09   DYN   500
1003  22   JIZ  1007
1004  10   OUT   600
1005  20   JMP  1055 --->

1007  01   STH     1
1008  09   DYN   501
1009  22   JIZ  1026

1011  29   PAS   250
1012  00    00   600 ; SADD
1013  00    00   605 ; DADD
1014  00    00     1 ; 1 bit
1015  00    00     0
1016  00    00     0
1017  00    00     0
1018  00    00     0
1019  00    00     0
1020  00    00     0

1022  20   JMP  1055 --->

1026  01   STH     2
1027  09   DYN   502
1028  22   JIZ  1044

1031  29   PAS   250
1032  00    00   605 ; SADD
1033  00    00   600 ; DADD
1034  00    00     1 ; 1 bit
1035  00    00     0
1036  00    00     0
1037  00    00     0
1038  00    00     0
1039  00    00     0
1040  00    00     0

1042  20   JMP  1055 --->

1044  01   STH     3
1045  09   DYN   503
1046  22   JIZ  1055 --->
1047  16   SEI     0
1048  12   REO  1599
1049  27   INI     7
1050  21   JIO  1048

1055  16   SEI     0
1056  01   STH  1600
1057  10   OUT  1000
1058  27   INI     5
1059  21   JIO  1056
1060  20   JMP  1001
```

Load

Shift upwards

Shift downwards

Reset

Transfer flags to outputs 040...045

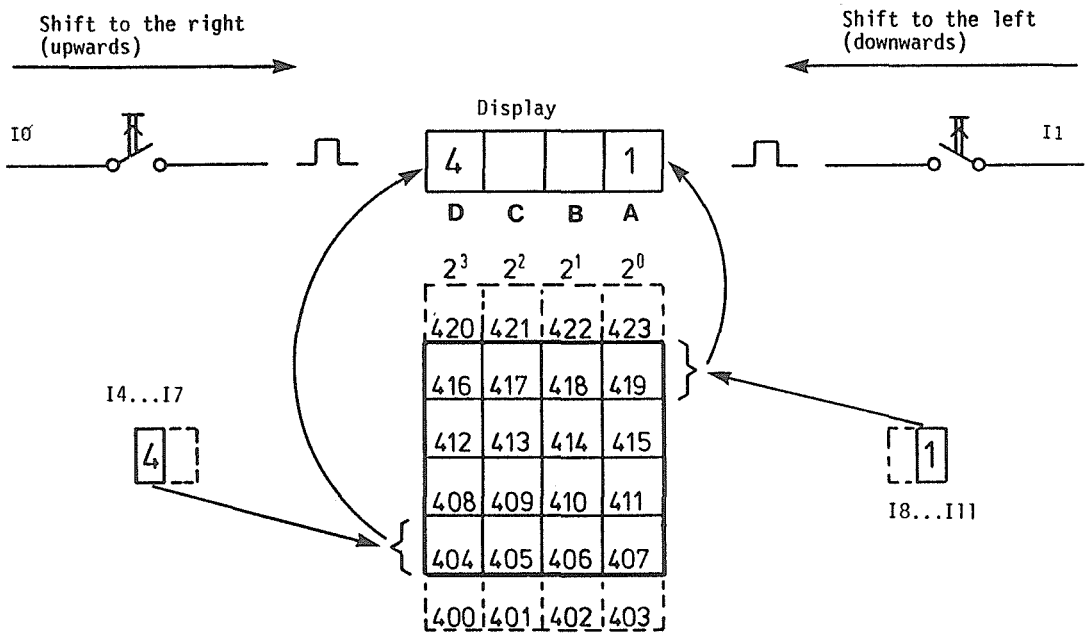
The present example shows a simple principle model. Compared to the result the procedure is rather complicated. The advantages of the PAS 250 make themselves felt in the execution time (in this example 250µs) and when more comprehensive register structures are involved.

Example 9 BCD-shift register upwards/downwards (using PAS 25Ø)

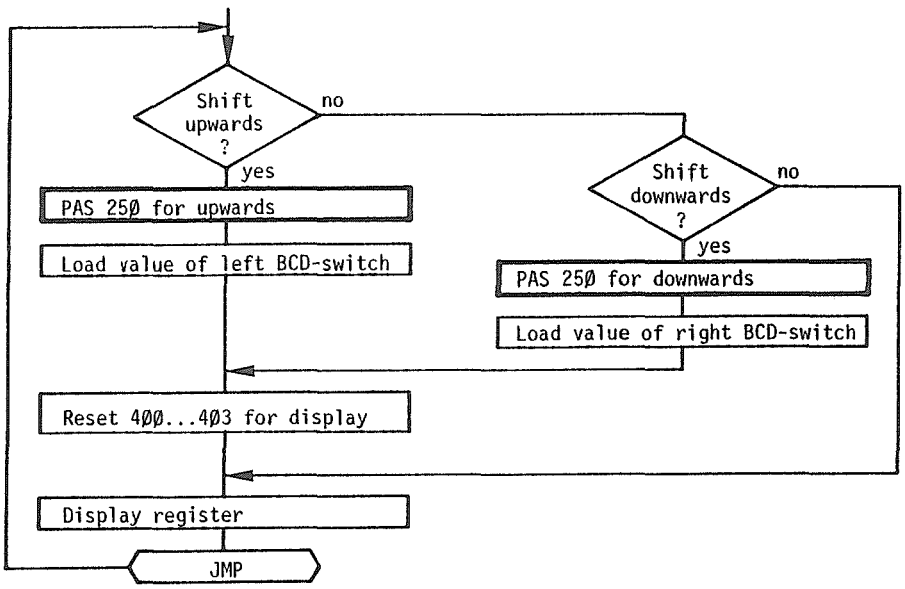
Description

The shift register is 4 bits long and 4 bits wide. The BCD-switches on the inputs I4...I11 are used for entering the status, the left-hand number (I4...I7) for shifting upwards, the right-hand number (I8...I11) for shifting downwards.

The register consists of flags 4Ø4...419. The shift process is triggered by input IØ (shifting upwards) or by I1 (shifting downwards). The numbers are displayed in the operand display.



Flowchart



*) The flags 4ØØ...4Ø3 and 42Ø...423 must be reserved as circulating buffers.

Solution 9

```
1101  01  STH     0
1102  09  DYN   511
1103  22  JIZ  1131

1111  29  PAS   250
1112  00   00   404  ; SADD
1113  00   00   419  ; DADD
1114  00   00     4  ; N = 4 bits
1115  00   00     0
1116  00   00     0
1117  00   00     0
1118  00   00     0
1119  00   00     0
1120  00   00     0

1123  16  SEI     0
1124  01  STH  1004
1125  10  OUT  1404
1126  27  INI     3
1127  21  JIO  1124
1128  20  JMP  1161

1131  01  STH     1
1132  09  DYN   512
1133  22  JIZ  1173

1141  29  PAS   250
1142  00   00   419  ; SADD
1143  00   00   404  ; DADD
1144  00   00     4  ; N = 4 bits
1145  00   00     0
1146  00   00     0
1147  00   00     0
1148  00   00     0
1149  00   00     0
1150  00   00     0

1153  16  SEI     0
1154  01  STH  1008
1155  10  OUT  1416
1156  27  INI     3
1157  21  JIO  1154

1161  16  SEI     0
1162  12  REO  1400
1163  27  INI     3
1164  21  JIO  1162

1171  15  SCR   300
1172  19   19   419
1173  31  DTC   300
1174  20  JMP  1101
```

Shifting upwards

Load value of left BCD-switch

Shifting downwards

Load value of right BCD-switch

Reset flags 400...403 *

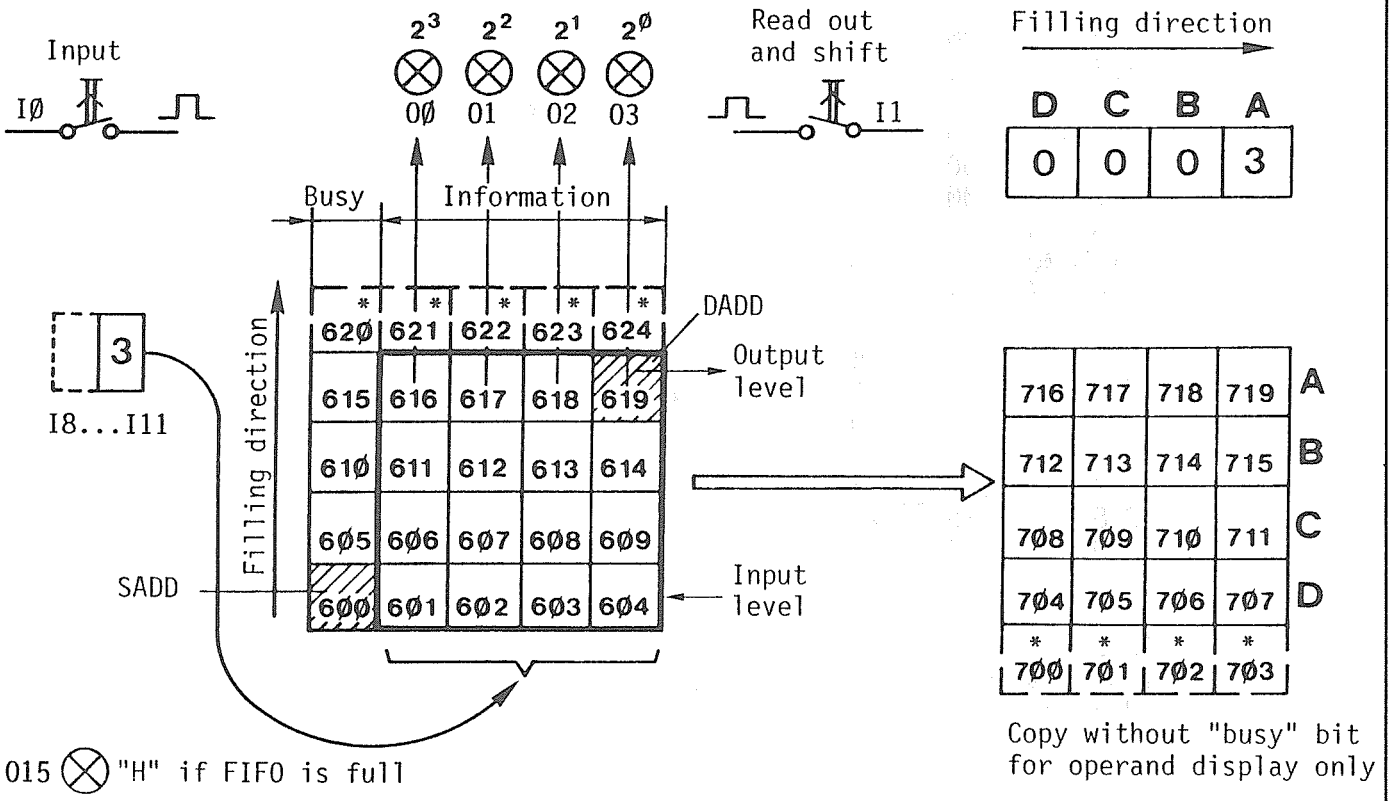
Transfer shift register to display and counter

*) Flags 400...403 must be reset, as 20 continuous bits BCD are read into the counter due to code 19.

Example 1Ø FIFO-register (using PAS 251)

Description

The FIFO-register is 4 bits long and 4 bits wide. The BCD-switch at the inputs I8...I11 is used to enter information. The register consists of flags 6ØØ...619. A BCD-value is written into the FIFO via input IØ, a value is read out at the outputs 0Ø...03 via I1. Output 015 is activated, if the register is full. The numbers are displayed in the operand display.



In order to be able to display the contents of the FIFO in the operand display it must have a form which consists of the information part only. This is achieved by copying the register without the "busy" bit into another flag range. It can now be loaded into a counter via code 19 (5 x 4 bits BCD) and displayed.

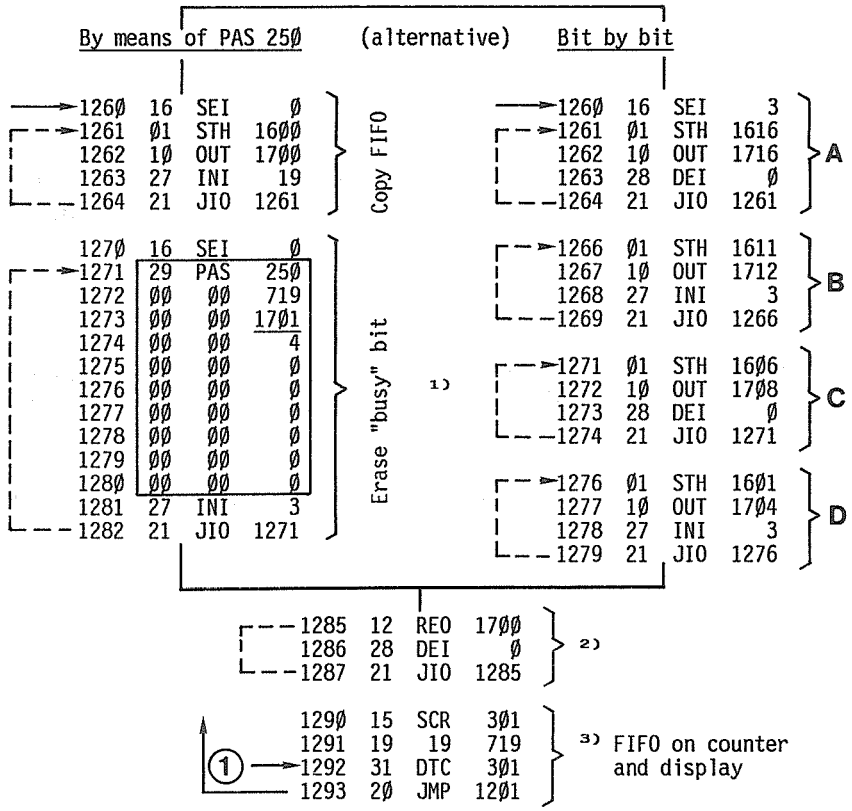
---

*) Flags 62Ø...624 must be reserved as circulating buffer when using PAS 25Ø.

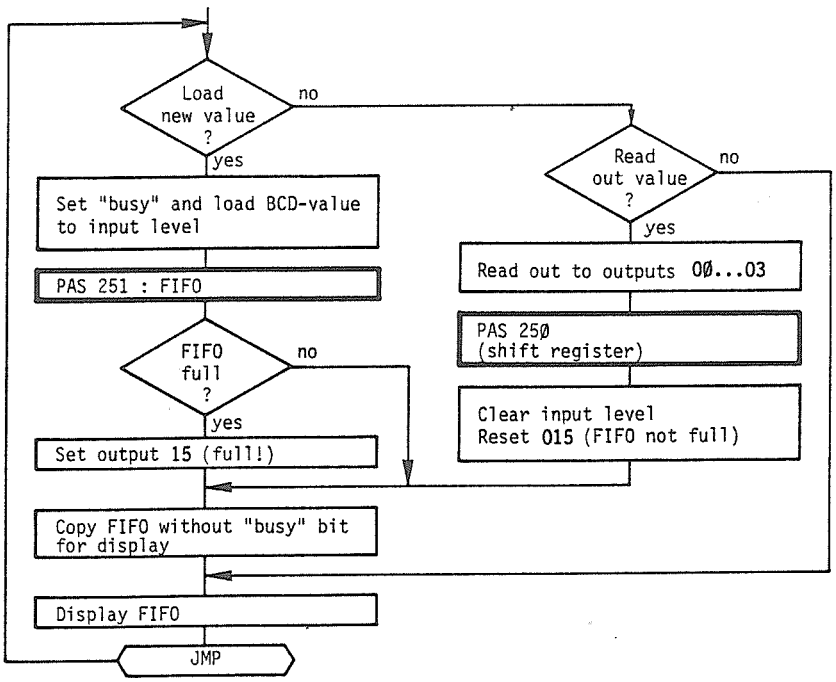
Solution 1Ø

```
1201  Ø1  STH     Ø
1202  Ø9  DYN   52Ø

1203  22  JIZ  1227
1204  11  SEO   6ØØ  ; Set "Busy"
1205  16  SEI     Ø
1206  Ø1  STH  1ØØ8
1207  1Ø  OUT  16Ø1      Load
1208  27  INI     3      BCD-value
1209  21  JIO  1206

1211  29  PAS   251
1212  ØØ  ØØ   6ØØ  ; SADD                 Load input level
1213  ØØ  ØØ   619  ; DADD                 of the FIFO
1214  ØØ  ØØ     5  ; 5 bits (4 + busy)
1215  ØØ  ØØ     Ø
1216  ØØ  ØØ     Ø
1217  ØØ  ØØ     Ø
1218  ØØ  ØØ     Ø
1219  ØØ  ØØ     Ø
122Ø  ØØ  ØØ     Ø

1222  Ø1  STH   6ØØ  ; Busy ?
1223  1Ø  OUT    15  ; FIFO full
1224  2Ø  JMP  126Ø ---->

1227  Ø1  STH     1
1228  Ø9  DYN   521
1229  22  JIZ  1292

1231  16  SEI     Ø
1232  Ø1  STH  1616      Read out             Read out and
1233  1Ø  OUT  1ØØØ      BCD-value            transfer to out-
1234  27  INI     3                           puts ØØ...Ø3
1235  21  JIO  1232

1241  29  PAS   25Ø
1242  ØØ  ØØ   6ØØ  ; SADD
1243  ØØ  ØØ   619  ; DADD
1244  ØØ  ØØ     5  ; N = 5 bits
1245  ØØ  ØØ     Ø
1246  ØØ  ØØ     Ø
1247  ØØ  ØØ     Ø                           Shift and enter
1248  ØØ  ØØ     Ø                           zero
1249  ØØ  ØØ     Ø
125Ø  ØØ  ØØ     Ø

1252  16  SEI     Ø
1253  12  REO  16ØØ
1254  27  INI     4
1255  21  JIO  1253
1256  12  REO    15  ; FIFO empty

(1)
```

Copying FIFO onto flag range 7Ø4...719 for DTC-display



Flowchart



---

1) The four flags 697...7ØØ must be reserved as flag buffers to ensure correct execution of the instruction PAS 25Ø.
2) The flags 7ØØ...7Ø3 must be reset, as 5 x 4 continuous bits BCD are read into the counter due to code 19.
3) The whole program on this page is only necessary if the values in FIFO must be displayed via DTC.

Example 11 Trouble reports
(storage in the FIFO-register using PAS 251)

Description

Various trouble reports must be output as clear text. It must be noted that the malfunctions must be printed in the sequence in which they occur. Moreover, further malfunctions must not be lost while a trouble report is being printed.

Problem 11

Four different malfunctions must be simulated via inputs Ø...3 and - provided with date and time - they must be printed in the sequence in which they occur. This is achieved as follows:

(For the sake of simplicity, the point of time is given at which the respective malfunction is printed out, not the time at which it has occured.

IØ : Date, time
alarm Ø
comment

I1 : Date, time
alarm 1
comment

I2 : Date, time
alarm 2
comment

I3 : Date, time
alarm 3
comment

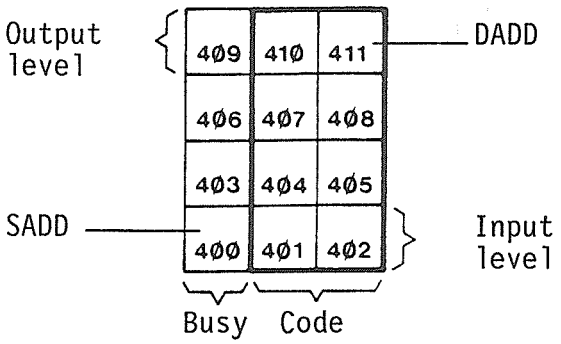
Solution 11

The problem is solved with the aid of a FIFO-register. Each trouble report is assigned a certain identification code. If a malfunction occurs, the respective code is read to the input level of the FIFO and the busy-bit (SADD) is simultaneously set to "H". PAS 251 is then executed, thus shifting the information (code + busy) to the last vacant position of the FIFO. (If FIFO is empty: to the output level). If the busy-bit of the output level is "H", the code is used to find out what kind of a malfunction it is. (Code ØØ means alarm Ø, code 1Ø ---> alarm 1, code Ø1 ---> alarm 2, code 11 ---> alarm 3).
Depending on the code, the counter C319 is set to the value Ø, 1Ø, 2Ø or 3Ø. The value of the counter C319 is then loaded into the index register. Due to the indexed text output

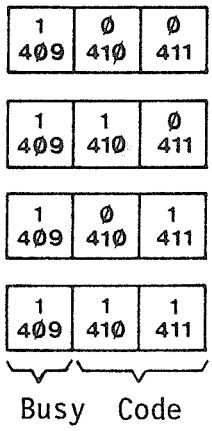
```
PAS   23
 ØØ 111Ø
```

the value of the index register is added to the text number 11Ø and text number 11Ø, 12Ø, 13Ø or 14Ø is consequently output, which corresponds to the alarm message Ø, 1, 2, 3.
After each text output the FIFO-register is shifted by means of PAS 25Ø.

The FIFO register is the following:



Interpretation of the output level:



Alarm Ø ---> load value Ø into C319

Alarm 1 ---> load value 1Ø into C319

Alarm 2 ---> load value 2Ø into C319

Alarm 3 ---> load value 3Ø into C319

Program

```
1301  29   PAS   100
1302  00    00   902 ; 4800 bauds
1303  00    00   425 , TXB
1304  00    00   254
1305  00    00   254
1306  00    00     0
1307  00    00     0
1308  00    00     0
1309  00    00     0
1310  00    00     0

1321  01   STH     0 ; Alarm 0
1322  09   DYN   420
1323  22   JIZ  1327
1324  11   SEO   400 ; SADD
1325  12   REO   401 } Code for
1326  12   REO   402 } alarm 0
1327  01   STH     1 ; Alarm 1
1328  09   DYN   421
1329  22   JIZ  1333
1330  11   SEO   400 ; SADD
1331  11   SEO   401 } Code for
1332  12   REO   402 } alarm 1
1333  01   STH     2 ; Alarm 2
1334  09   DYN   422
1335  22   JIZ  1339
1336  11   SEO   400 ; SADD
1337  12   REO   401 } Code for
1338  11   SEO   402 } alarm 2
1339  01   STH     3 ; Alarm 3
1340  09   DYN   423
1341  22   JIZ  1345
1342  11   SEO   400 ; SADD
1343  11   SEO   401 } Code for
1344  11   SEO   402 } alarm 3
1345  01   STH   400 ; SADD = "H"?
1346  22   JIZ  1361

1351  29   PAS   251 ; FIFO
1352  00    00   400 ; SADD
1353  00    00   411 ; DADD
1354  00    00     3 ; Width = 3 bits
1355  00    00     0
1356  00    00     0
1357  00    00     0
1358  00    00     0
1359  00    00     0
1360  00    00     0

1361  02   STL   409 ; Read flag 0
1362  05   ORH   425   or TXB
1363  21   JIO  1321
1364  15   SCR   319
1365  00    00     0
1366  01   STH   410 ; DADD-1
1367  15   SCR   319
1368  27    27    10
1369  01   STH   411 ; DADD
1370  15   SCR   319
1371  27    27    20
1372  16   SEI   319
1373  29   PAS    23
1374  00    00  1110 ; Text 110+C319
1375  12   REO   409 ; Busy output
                       level

1381  29   PAS   250 ; Shift
1382  00    00   400 ; SADD
1383  00    00   411 ; DADD
1384  00    00     3 ; Width = 3 bits
1385  00    00     0
1386  00    00     0
1387  00    00     0
1388  00    00     0
1389  00    00     0
1390  00    00     0
1391  20   JMP  1321
```

Text:

```
110  .$.H^M^J.A.L.A.R.M.   } Alarm 0
111  .0^M^J.$.L.1.4.5^M^@  }
120  .$.H^M^J.A.L.A.R.M.   } Alarm 1
121  .1^M^J.$.L.1.4.5^M^@  }
130  .$.H^M^J.A.L.A.R.M.   } Alarm 2
131  .2^M^J.$.L.1.4.5^M^@  }
140  .$.H^M^J.A.L.A.R.M.   } Alarm 3
141  .3^M^J.$.L.1.4.5^M^@  }
145  . .C.O.M.M.E.N.T. .:
146  .$.X.0.7.0^M.$. .0.1
147  .0.$.Y.0.7.0^M.$. .0
148  .1.0.$.Z.0.7.0.$.U^@
```

Summary of the instruction set 2 + 3 in alphabetical order

Notes:

SAIA AG
Industrial Electronics and Components
3280 Murten/Switzerland

Telephone 037 727111
Telefax 037 71 44 43
Telex 942127

Further representatives

Belgique Landis & Gyr Belge SA, Dépt. Industrie
Avenue des Anciens Combattants 190, B-1140 Bruxelles
☎ 02 244 02 11, Tx 65930, Fax 02 242 88 31

Danmark Skandia-Havemann
Vallensbækvej 46, DK-2625 Vallensbæk
☎ 02 64 33 33, Tx 33383, Fax 02 64 22 45

Deutschland SAIA GmbH
Flinschstrasse 67, D-6000 Frankfurt 60
☎ 069 420993-0, Ttx 6999375, Fax 069 425654

España Landis & Gyr BC SA
Batalla del Salado 25, Apartado 575, 28045 Madrid
☎ 91 467 19 00, Tx 22976, Fax 91 239 44 79

France SAIA Sàrl
10, Blvd. Louise Michel, F-92230 Gennevilliers
☎ 1 40860345, Tx 613189, Fax 1 47 91 40 13

Great Britain A.S.A.P. Ltd.
Unit 15D, Compton Place, Surrey Avenue, Camberley, Surrey GU15 3DX
☎ 0276 691 580, Fax 0276 69 15 81

Italia SAIA S.r.l.
Via Cadamosto 3
20094 Corsico MI

Nederland Landis & Gyr BV, Div. Electrowater
Kampenringweg 45, Postbus 444, NL-2800 AK-Gouda
☎ 01820 65683, Tx 20657, Fax 01820 32437

Norge Malthe Winje & Co A/S
Cort Adelersgt. 14, Postboks 2440, Solli, N-0202 Oslo 2
☎ 02 55 86 40, Tx 19629, Fax 02 55 22 11

Österreich COMECON Landis & Gyr Gesellschaft m.b.H
Breitenfurterstrasse 148, Postfach 9, A-1230 Wien
☎ 0222 842626-0, Tx 132706, Fax 0222 842626313

Portugal Infocontrol Electronica e Automatismo LDA.
Av. da Igreja No. 68–1° Esq., P-1700 Lisboa
☎ 01 77 51 61-65, Tx 63454, Fax 01 77 56 87

Suomi Finland OY Landis & Gyr AB
SF-02430 Masala
☎ 8 02 97 31, Tx 100 11 53, Fax 8 02 97 55 31

Sverige Beving Elektronik AB
St. Eriksgatan 113a, Box 21104, S-10031 Stockholm
☎ 08 15 17 80, Tx 10040, Fax 08 33 68 63

USA After sales services: Maxmar Controls Inc.
99 Castleton Street, Pleasantville, New York 10570-3403
☎ 914 747 3540, Fax 914 747 3567

Australia Landis & Gyr (Australia) Pty Ltd
411 Ferntree Gully Road, P.O. Box 202, Mount Waverley, Vic. 3149
☎ 3 544-2322, Tx 32244, Fax 3 543 74 96

Argentina Electromedidor S.A.I. y C.
Defensa 320, RA-1065 Buenos Aires
☎ 1 337125, Tx 23377, Fax 1 331 95 82

Subject to change without notice

Printed in Switzerland 26/722 E1 1.90 G02