

PC20 PLC maintenance

Rack version PLC

VERSION 0.0

Creation date: 2005-03-16

Last Modification date 2005-04-13

KTA b.v.

Under Construction

H.C.J.J Kanters

Table of contents

Table of contents.....	2
1. Introduction.....	4
1.1 Version control.....	4
1.2 About this document.....	4
2. Introduction PC20 PLC.....	5
2.1 Operating system problems for support programs.....	5
3. The PC20 PLC system.....	6
3.1 PLC system.....	7
3.2 Specifications	8
3.3 PLC 20 Hardware overview.....	8
CPU Card.....	8
Memory Cards.....	8
power supply.....	9
rack (backpanels)	9
Input / Output cards	9
Communication cards	9
Special cards	9
Wiring support Connection cards.....	9
3.4 Configuration and Addressing of the PLC PC20.....	10
Module Identification (MID).....	10
Separation Code for Inputs and Outputs (SCIO).....	11
Hardware lay out in PC20 Program.....	11
The CI20/21 for Programming	11
The MM20..26 memory card.....	12
The CP20..25 Central Processor Unit.....	12
The RS20 Communication between PC20's.....	12
The PM25 Servo positioning.....	13
The VI20/21/22 Bidirectional serial interface.....	13
The VI21 Bidirectional serial/network interface.....	14
The VI22 Bidirectional modbus interface.....	14
The Analog Input /Output cards.....	14
The DM20/21/22/23 memory cards.....	14
The EC20 high speed counter card.....	15
4. PLC Software and Software Tools.....	16
4.1 The Instruction set	16
The logic instructions AND, OR, AND NOT, OR NOT, TRIG	16
Execute instructions EQL,EQL NOT, SET1,SET0, FETCH BIT,STORE BIT	17
Execute instructions FETCH CONSTANT, FETCH DIGIT, STORE DIGIT.....	18
Execute instructions ADD, SUBTRACT, DIVIDE, MULTIPLY.....	18
Execute instructions COMPARE, COUNTDOWN, COUNTUP.....	19
Execute instructions SHIFT LEFT, SHIFT RIGHT.....	20
Jump instructions JUMP TO SUBROUTINE TRUE/FALSE ,RETURN.....	20
Jump instructions JUMP RELATIVE TRUE/FALSE.....	21
System Instructions END, LAST INPUT OUTPUT and NO OPERATION.....	21
4.2 Program examples	22
AND , OR EQL	22

CLOCK TIMING SIGNALS.....	25
COUNTER&TIMER.....	26
SUBROUTINES.....	27
CALCULATIONS.....	28
4.3 The program layout	29
Standard program structure	29
Often used program structure	30
Start up delay.....	31
4.4 PDS35.....	31
4.5 Developed Software tools by third parties.....	31
5. Converting PC20 to S7 (or other PLC,s).....	33
6. Working online with the PC20.....	34
6.1 PDS 35 online monitoring.....	34
6.2 MONI online monitoring.....	35
6.3 PC20 online tools for Windows XP and W2000.....	38
7. APPENDICES.....	39
7.1 Apendix A PLC Hardware environment in Source code.....	39
7.2 Apendix B Abbreviations and name explanations.....	40
7.3 Apendix C Short Instruction Set PC20.....	42
7.4 Apendix D Wiring diagram Programming cable PC20.....	44

1.Introduction

- This document gives a short description of how to understand and maintain the Philips PC20 PLC. The PC20 is an "old" PLC system and is not produced any more, at the time this document is written. For a full explanation of the PC20 and its functioning the "PC20 USER MANUAL" needs to be consulted that was issued by "N.V. Philips Gloeilampen Fabrieken". More recent information of the PC20 PLC can be achieved from Nyquist Control Systems which is supporting the PC20 PLC at the moment.
- This document is mainly a document for service purposes and for the software maintenance of the PC20 system and its related control system.
- In this document the PC20 rack version configuration that is common used is explained, with a selection of the hardware and examples of the software.
- The PC20 is a produced in more hardware versions. The most common used version is the PC20 in the rack version. Other versions are the MC20/MC30 /MC31 which is a flat "Pancake" version, this is often used in a smaller automation configurations. The PLC773 is a single rack card model and is used rarely. The MC40/41 is a single micro processor model and does not uses the Processor chip set that is used in the other members of the PC20 family.

1.1Version control

In order to trace document differences and to ensure project management quality this document will be under version control.

The following table shows the history of this document.

Version	date	description	name
0.1	10 march 2006	First created, draft version	H.Kanters

1.2About this document

The information placed in this document is constructed from other documents ,user manuals , existing PC20 programs and Pc20 hardware datasheets . This document does not imply to be complete it is mainly an setup for a maintaining and supporting still running applications. This document should also be helpful if there is a need to perform small changes or extensions to the hardware or software in existing applications.

If you, have questions, remarks or idea's for additions to complete or improve this document. Please contact.

Henri@ktautomation.nl

A list of information sources:

Name	Description	Number
Nyquist	Supplier PC20 and P8 hardware	contact.us@nyquist.com
G.Withagen	MID SCIO addressing information	Date:
H.Kanters	Program examples	Henri@ktautomation.nl
PC20 User Manual	User Hardware and Software description	N.V. Philips Gloeilampen Fabrieken
VHE	Several documents and examples	VHE Industrial Automation info@Vhe.nl
DM20 document	Product Data sheet	12nc: 9498 733 00413
PDS35 Handleiding	Dutch: user manual	Version B.7
VI20 document	Product Data sheet	12:nc: 9498 733 02611
PTE Philips	Several documents and examples	
PM25	Servo Control Card on PM25 H.Mansvelt	Philips Centre For Manufacturing Technology CFT
PM25	Technical Documentation CFT	12nc: 8122 968 5013

2.Introduction PC20 PLC

This chapter describes the specific features of the PC20 PLC in order to understand the functioning of the PC20 and its related peripherals. The description in this chapter does not intend to describe the complete set of features of the PC20 PLC but can be seen as an introductory of some of the most often used ones.

The PC20 is a dedicated processor system that handles the directly data, inputs and outputs that is interpreted by the CPU, that reads task information from the command instructions in the software Program.

Because the PC20 has no system software but system hardware, the PC20 is a relatively fast PLC compared to the actual speed of the Chipset compared to other PLC's. This advantage is also a disadvantage because the instruction set is hardware limited and can not be updated to a version that is normally used in more modern PLC's.

This disadvantage is the lack of instructions to communicate in a network or other sophisticated communication means. The lack of having sufficient Communication means is the greatest disadvantage of the PC20.

Another disadvantage is the lack of original support software and hardware for programming and debugging of the hardware and software. In the past many users of the PC20 PLC created their own development system en debugging programs in order to be able to program , debug and document the PC20 automation projects they were working on. Some of these software tools will be explained in this document. This will be a small collection of as the programs that were created during the "life cycle" of the PC20.

The programming on the PC20 can be done directly in the online object program (OBJ) by means of a desk terminal ,a hand terminal or a normal TTY monitor with the TTY ... protocol. In the past PDS35 , PDS5 , Top Promisys and were used in order to be able to create a text source program with symbolic names and explanation text. By means of this source program it was easier to document the functioning of the software in the application.

The PDS35, PDS5, TopPromisys and Fusion Had to convert the source program to a object program that had to be downloaded into the PC20 PLC. Also upload of the object program from the PLC into the PC on which the support programs were running. Each of these programs had specific features that the others did not have introduced specific way of working. An example is a kind of scope function with trigger possibilities in TopPromisys. Also Fusion was a graphic programming language that also could convert source code to Siemens S5 object programs.

Because this document has as a target to give clarity on the PC20 PLC system itself only a few of these features of the support programs are explained in this document. The use of the PDS35 support system is most commonly used and there fore some of the functions and possibilities of the TeHa PDS35 support system will be explained .

2.1Operating system problems for support programs

Because most of the support programs were created in the period that DOS, Windows 95 and Windows98 were used. Most of the programs have problems when used under Windows 2000 and windows XP and the generation operation systems that will be used in the future.

For maintenance purposes often an old PC with windows 98 or Windows 95 is used to support the PC20 software tools in order to maintain the PC20 PLC equipped machinery.

For maintenance purposes programs that work under windows XP and windows 2000 for monitoring, downloading, uploading and changing online are under construction. For development of new applications and large program changes the "old" programs that work under Dos .Window95 or Windows 98 should be used.

3.The PC20 PLC system

The PC20 cycle consists of a “data processing” phase and a I/O phase. During the data processing phase the Inputs and other internal data components are evaluated by the user application program. This results in controlling and defining a value for the outputs. During the I/O phase this outputs are copied to the physical outputs and inputs. Also data exchange between the PC20 PLC and SCADA systems or other communication partners is executed during the I/O phase.

The PC20 PLC has separate memory for the program and the stored data. The stored data that also includes the I/O is called the SMA (Scratch path Memory Address). The SMA is stored in the CPU of the PC20 PLC and can be battery backed up.

The first 12 bits of the SMA have a specified system function and can not be used freely by the user application program. In the following lines in the source program these bits are explained. Note that the fixed addressed bits are different in the MC30/31 and the MC41/40. These differences are not described in this document.

The first 12 bits as used in a PDS35 source program:

Example symbolic names for first 10 bits (11 and 12 can be used in the program)

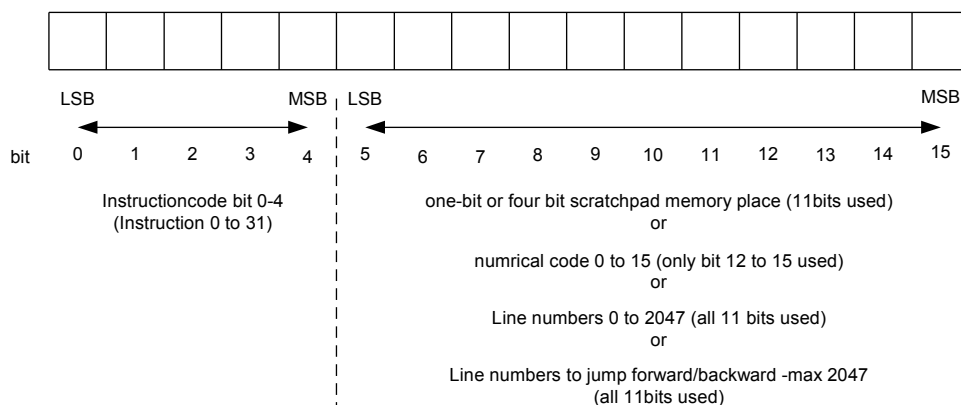
```
EQ *****
EQ          *** STANDARD AND GENERAL PLC BITS ***
EQ *****
EQ
EQ  END_0      =      0      PROGRAM END
EQ  OVERFL     =      0.0    ARITHMATIC OVERFLOW
EQ  ALWAYS     =      0.1    ALWAYS ACTIVE=(1)
EQ  ALARM      =      0.2    24VDC BELOW 17.5V
EQ  C10MS      =      0.3    CLOCK 10 MSEC.
EQ  C100MS     =      1.0    CLOCK 100 MSEC.
EQ  C1S        =      1.1    CLOCK 1 SEC.
EQ  C10S       =      1.2    CLOCK 10 SEC.
EQ  C60S       =      1.3    CLOCK 1 MIN.
EQ  PAGE0      =      2.0    BIT PAGING 2^0
EQ  PAGE1      =      2.1    BIT PAGING 2^1
```

The program memory can go up to 16K . This maximum limitation is caused by the use of the dedicated chipset of the PC20 PLC that only has 11 bits for addressing the data memory area. This addressing is organized in the following way:

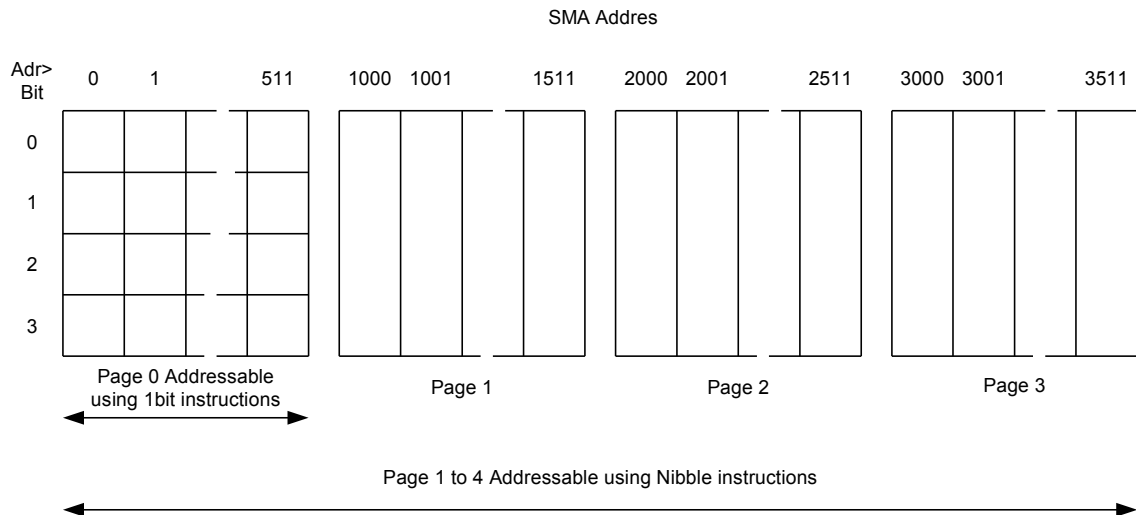
Address lines ADD0 until ADD8 describe the nibble 0 until 0511 in the page. Pages 0 until 3 are possible

Address lines ADD9 until ADD10 describe the bit in the nibble 0,1,2,3 or the Page 0,1,2,3 if a Nibble instruction is used

Organization of the Program word of PC20 program memory



This restriction causes that without using tricks the PC20 PLC can only address bits in the first page 0. In all other pages 1 until 3 bits can not be addressed and only Nibble instructions are possible. Page switching is possible but not recommendable because if it is used the cross reference tool is not reliable any more and debugging is much more difficult. For that reason Page switching is not described in this document. Other options to approach bits in the pages 1 to 4 are using subroutines that swap the SMA data from page 1-3 to page 0 and back again . see the sub chapter subroutines. The SMA area layout in the maximum configuration is seen in the following drawing.



Not all CPU's support all the pages, and some older types even only support parts of these pages. Also the MC30/31/40/41 types have specific differences in page layout using the first 12 bits of the SMA.

3.1PLC system

This sub-chapter describes all relevant details how the system of the PC20 is functions. This can work alighting during the rest of the explanation of the PC20 PLC system.

To understand the functioning of the PC20 PLC system it is important to understand the way the system hardware is organized. The PC20 PLC system consists of the following essential parts that are used by the PC20 instruction set. Not all essential parts are presented in the following schematic but only those whose names are used in the explanation of the instruction set.

The Address counter	(9Bits address + 2 Bits for Page"0-3" or 2 Bits for Bit address "0-3")
A-Register	(16Bit FIFO register used for most arithmetic operations)
B-Register	(16Bi FIFO register used in combination with the A-register)
M/Q Register	(16Bit register used for divide and multiply operations)
Jump Register	(4 Return addresses , 4 deep nesting is possible)
Comparator	(4 Bits)
Shift Register	(4 Bits)
Timing and Control	(Sequence generator that reads the instructions and activates the other parts)

The Timing and Control hardware reads the program memory and activates the different hardware parts depending on the instruction and the addressing that is in the user application software.

3.2 Specifications

This sub-chapter describes all relevant specifications of the PLC system in order to understand the limitations that can cause problems during maintenance of the software and hardware.

The PC20 PLC has the following features and missing features:

- A dedicated hardware CPU for high speed controller functions (Cycle time 10-30 mSec average)
- A Five bits data area for instructions resulting in a maximum of 32 instructions (only 31 used)
- A Eleven bits address area. Divided in a Nine Bits (0-511) and Two bits bit (0-3) bit selector Or two bits page selector (0-3)
- Internal power supply of 10VDC. SM20 or SO20 cards convert 24VDC to 10VDC.
- One Program file, the program consists of one large list of instructions. No sub divisions.
- A scratchpad data memory of 2K4 (2k of 4bit Nibbles)
- Up to 16K program lines .A instruction (operator) followed by data (operand).
- Hardware addressing is done by soldering paths on the backpanel rear side.
- Programming only possible over RS232 CI20/21 programming interface
- Networking possible RS485 PPCCOM via CI20/21 or V20/21.
- Indirect addressing not possible with Instruction from the instruction set. (No relative jump). This has to be solved with the VI20/21 or with the DM20 cards

3.3 PLC 20 Hardware overview

This sub-chapter describes general information of the hardware, for specific hardware information the specific data need to be used.

Some special cards are produced by companies and Philips departments in order to obtain a functionality that was not present in the official range of PC20 cards. Information of these cards can only be obtained from the manufacturer of these cards. Most of the information about the other cards can be found in the PC20 User manual and the specific data sheets.

TIP: If Configuration of the PLC fails remove most of the cards and leave only that card you want to check. First try to make this card work .If this functions ok then move the other cards in one by one and keep on checking the functioning of the card you wanted to check. Always switch of the power to the rack if you are exchanging PC20 cards.

CPU Card

CP20	Eprom card	0,25K4 Scratchpad and 2K16 Program battery external
CP21	C-MOS RAM	0,25K4 Scratchpad and 1K16 Program battery internal
CP22	C-MOS RAM	2K4 Scratchpad and 1K16 Program battery external
CP24	C-MOS RAM	0,25K4 Scratchpad and 2K16 Program battery internal
CP25	C-MOS RAM	2K4 Scratchpad and 2K16 Program battery internal

Memory Cards

MM20	Memory cards with 8K16 eprom
MM21	Memory cards with 8K16 cmos ram battery on board
MM24	Memory cards with 4K16 cmos ram battery on board
MM25	16K Memory cards with eprom 2764
MM26	16K Memory cards with eeprom

power supply

SM20	Power supply 24VDC to 10VDC
SO20	Power supply 24VDC to 10VDC + power outputs

rack (backpanels)

BP23a/25/26	Backpanels for the PC20
BP23b	Backpanel lower card for CPU,MM and CI cards (8K and 16K version)
SC20	Small controller Cabinet

Input / Output cards

IM20/21/22	Input cards
OM21/22	Output cards
RP20	bidirectional parallel interface input and output module
EC20	High speed counter card
DA20	Digital to Analog output card
DA20	Analog to digital input card
AO20	Analog output module card
AI20	Analog input module card

Communication cards

CI20/21:	Used for programming i(switch to VDU) and networking (switch to ppccom)
RS20	Bidirectional Serial Interface
VI20/21	Programmable Communication card for RS485/232 communication
VI21/1	Programmable Communication card communication with PPCCOM network
PU21/23	Specific programming card to use in combination with the PU20/40

Special cards

PM25	Servo control position unit.
DM20	Memory data controller card
DM21/22/23	Memory data storage modules 128K (64Kram/64Keprom)
LC20	PID Loop controller

Wiring support Connection cards

Many different cards were developed. Some by local divisions of Philips, this was done to avoid soldering and connect the I/O of the cards to screw connectors by using standard cable and connector sets. Some of these connector cards are:

AA22	Adaptor for IM22
AA23	Adaptor for OM22
AA28	Adaptor for RP20
AA29	Adaptor for RS20
AA34	Adaptor for VI21

3.4 Configuration and Addressing of the PLC PC20

This sub-chapter describes the hardware configuration. The addressing of each card specific is not explained in this section only a remark is made if specific addressing is necessary. The addressing of the cards in the panel is performed by interconnecting the solder paths on the rear side of the rack on the back panel.

The addressing of the inputs and outputs is directly coupled to the SMA and all cards can be addressed in all 4 pages of the SMA. However bit oriented cards are mostly addressed in page 0 and Nibble oriented cards are mostly addressed in the pages 1 until 3.

Module Identification (MID)

By giving the module the correct address by means of the MID soldering paths the card and its addresses are unique identified. And can be addressed from the software on that exact location in the SMA.

The MID addresses are binary coded and start with $2^1 = 2$ and not with the 2^0 as would be expected. Also if a card has an input or output range of 4 Nibble ($4 \times 4 = 16$ Bits) the start address should always be a multiple of 4 Nibbles. For example for a 4 Nibble input card the MID address should be 0,4,8 ect. And For a 8 Nibble output card the address should be 0,8,16,ect.

The MID soldering paths range from 1 to 8 for the addressing within a page and the MID soldering paths 9 and 10 are used for the page selection.

MID1	= 2^1	=2	Address
MID2	= 2^2	=4	
MID3	= 2^3	=8	
MID4	= 2^4	=16	
MID5	= 2^5	=32	
MID6	= 2^6	=64	
MID7	= 2^6	=128	
MID8	= 2^6	=256	
MID9	= 2^0	=1000	Page
MID10	= 2^1	=2000	

For example An IM20 Input Card with its inputs on 146.0 until 149.3.

The following paths should be soldered. ($130 = 128 + 16 + 2$)

MID1	= 2^1	=2	Address
MID4	= 2^4	=16	
MID7	= 2^6	=128	

For example An RP20 bidirectional parallel interface Card (16×8 Bit) on address 1032 until 1063.

The following paths should be soldered. ($1032 = 1000 + 32$)

MID5	= 2^5	=32	Address
MID9	= 2^0	=1000	Page

For some other cards the MID soldering paths are also used as setting for other features, it is wise to check the specific card data if you find a MID path soldered on an unexpected position. For example the RS20 card can only be addressed on value's 0,64,128 ect. The MID addresses from 1 until 32 have no addressing functioning but MID1 and MID2 can be in use.

MID1 (soldered) = RS20 card identified as active slave

MID2 (soldered) = RS20 card identified as passive slave

For the information active passive check the PC20 PLC manual and the data sheets of the RS20.

REMARK: If input card and output card with identical size are addressed on the same address the outputs of the input card are copied one on one to the inputs of the output card. This because both have a direct interface with the SMA area.

TIP: If addressing of card fails empty the PLC program only put a minimum test program in the PLC and check the whole data scratchpad area of the PLC if the card gives input or output response in an other area.

Minimum test program:

```
0      AND    0.1
1      LIO    2511
2      END    4
```

Separation Code for Inputs and Outputs (SCIO)

The Separation Code for Inputs and Outputs is used by the more complex data exchange cards like the RS20 and the RP20 card. These cards need to identify what part of the data exchange area is input and what part is output.

For some other cards the SCIO soldering paths are also used as setting for other features, it is wise to check the specific card data if you find a SCIO path soldered with an other card then RP20 or RS20.

The 4 SCIO soldering paths divide the data area in (2^4) 16 parts. This means for the RP20 a part is $2 \times 4\text{Bit}$ and for the RS20 a part is $4 \times 4\text{Bit}$. If the SCIO paths are soldered this indicates the address were the outputs will end and the inputs begins. See the following examples.

-If SCIO 3 is soldered first half (8 parts) of the data area will be output and second half (8 parts) will be inputs.

-If none of the SCIO paths is soldered the whole of the data area will be input

-If all of the SCIO paths are soldered the whole of the data area will be output except the highest part. So selecting only outputs is not possible with RP20 and RS 20.

For more example check the PC20 User manual

Hardware lay out in PC20 Program

In some of the Philips PC20 source programs a hardware layout is presented in the general purpose area, so the programmer has a clear view of the hardware and addressing used in the program. If this hardware layout list is up to date there is no need to go to the actual hardware to understand the addresses that are used in the program. An example of such a hardware environment layout can be found in Appendix A.

The CI20/21 for Programming

The CI20/21 is located in the most left position looking at the PC20 PLC rack. The CI21 is normally used for programming (switch on VDU) ,but can also be used as a communication card in a PPCCOM RS485/442 network for SCADA purposes.(switch on PPCCOM)

In this chapter only the CI21 used as a programming interface is described. Configuring the card is performed with the dipswitches on the CI21 card.

The setting most often used for the CI21 is the following:

9600 BAUD

RS232

Point to point

The connection diagram for the RS232 cable:

TIP: For checking the communication cable the MONI.EXE program tries connect continuously and indicates directly if the communication is established without an extra reset.

TIP: If no cable available a Serial connector converter from 9Pins male to 25pins female has the correct interface wiring and can also be applied together with a 1:1 cable as a PC20-RS232 interface cable.

The MM20..26 memory card

The Memory card that is mostly on the second position from the left needs no extra configuration. Using a 16K memory card also requires a small backpanel the BP23b of a 16K version. If this is not the case and a 8K version is used, this version can be adjusted by adding one connection on the BP23b. The specification how to add this connection can be found in the PC20 user manual.

TIP : If the software is changed that a download and stop of the PC20 is necessary. It is possible to download the software in an other Memory card in an other PC20 rack and swap the pC20 memory cards. In this way the PC20 is only stopped for a short period.

And if the software change does give the desired effect the old memory card can be placed back and the original program is restored quickly.

The CP20..25 Central Processor Unit

The CPU card is mostly positioned as third card from the left needs no extra configuration. A Reset of the program of the PC20 is possible with the RSE input on the backpanel B23b that connects the CI.., MM.. and the CP.. cards. If a reset of the program and a reset of the SMA in the CPU is required, then also the RSME input needs to be high during the use of the RSE signal.

On some BP23b backpanels the RSE is connected to a pushbutton and the RSME is connected to a switch.

TIP: If a software change caused a loop (a loop backwards in the program) that can not be stopped any more by the software tools simultaneously pushing the RSE and stop command over the CI20 help to program out of a never ending loop.

The RS20 Communication between PC20's

The RS20 card is a bidirectional serial interface card that can be used for remote I/O with a passive slave construction or it can be used between to PLC's for data exchange in a Master/Slave configuration. The data exchange between two PLC's is the most used option and will be explained in the subchapter.

The RS20 can exchange up to input and/or output 64Nibbles of with an other PLC. The SMA address these 64 Nibbles are positioned is configured by the MID address on the backpanel of the PLC rack. Normally the RS20 is addressed on Page 1 to 3, this is done because the shortage of bits that can become a problem in a large PC20 PLC program. The RS20 does on communication cycle per PCL scan cycle but there are applications in which the RS20 was activated more the once in a plc cycle in order to copy more data at once. The RS20 also has a start input that can be connected to the 24VDC or to an output of the PLC program, if this is not "1" (high) the communication will not work.

In a Master slave configuration it is necessary to identify the Master and the Slave. This is done by means of the MID solder pads (MID 1 to MID 5) that are not used because the lowest logical addressing pad is MID 6 (Address 64). These pads are used in the following way:

MID 1 Soldered	RS20 is a active slave	Mode B
MID 2 Soldered	RS20 is a passive slave	Mode C
MID 1 and MID 2 not soldered	RS20 is the Master	Mode A

The SCIO solder pads identify the data exchange input and output area's of the RS20 cards. The SCIO setting is presented in the following table.

SCIO code	Master		Active Slave		Passive Slave	
M 3 2 1 0	Output	Input	Input	Output	Input	Output
00 0 0 0 0	Not	allowed	Not	allowed	Not	allowed
01 0 0 0 1	4	60	4	60	4	60
02 0 0 1 0	8	56	8	56	8	56
03 0 0 1 1	12	52	12	52	12	52
04 0 1 0 0	16	48	16	48	16	48
05 0 1 0 1	20	44	20	44	20	44
06 0 1 1 0	24	40	24	40	24	40
07 0 1 1 1	28	36	28	36	28	36
08 1 0 0 0	32	32	32	32	32	32
09 1 0 0 1	36	28	36	28	36	28
10 1 0 1 0	40	24	40	24	40	24
11 1 0 1 1	44	20	44	20	44	20
12 1 1 0 0	48	16	48	16	48	16
13 1 1 0 1	52	12	52	12	52	12
14 1 1 1 0	56	8	56	8	56	8
15 1 1 1 1	60	4	60	4	60	4

TIP: If addressing the RS20 cards in a communication configuration between two PLC's causes problems. Replace the program by a minimum test program that only performs an I/O scan and check the complete SMA data area for data value's different then "0".I this way addressing errors and defect cards or backpanels can be detetcted. An example of a minimum test program is presented in the following lines.

```

0      AND    0.1
1      LIO    2511
2      END    4

```

It is sufficient to write down the original program lines 0 to 2, and restore these after the problem is solved.

The PM25 Servo positioning

The PM25 is used for servo positioning control and is not explained in this document. The PM25 is complex and needs more then a short introduction to be able to work with. (12nc 8122 968 5013)

The VI20/21/22 Bidirectional serial interface

The VI21is a bidirectional serial interface card. This card can be used as a communication card but as a co-processor that can perform special tasks for the PC20 PLC.

The VI21 is programmed in a similar way as the PC20 but has some restrictions. The data exchange with the PC20 PLC is arranged by means of a control register that is mapped on the SMA of the PC20. The PC20 starts the VI20 and waits for a Ready signal, then the PC20 knows new data is received from the VI21 and starts the VI21 again. Ect.

The program has to transferred on to eeproms (2) and they need to be placed in the specified sockets in the VI20. Standard software modules are available, for RS232 ,RS485, ect.

Online debugging is not possible, and make troublesome to program the VI20.

The PC20 defines in this control register a data exchange area that the VI21 can use to obtain its data from and to which the VI21 can write the results.

The control register bit functions are.

SYMBOL	VI in or out	Description
VIR	out	Vi Ready VIR=1, VI20 in idle state
Ci0,1,2	out	General purpose control bit originating in VI20
xx		No assignment
CO0,1	in	General purpose control bit originating in PC20
VIS	in	VI start, (if VIR="1") Start VI20 on edge VIS
I/O	in	I/O = "0" data from SMA to VI buffer . I/O = "1" data from VI buffer to SMA
A5 –A10	in	Location of the SMA area in the PC20. same addressing principle as MID5 to MID10

The VI21 Bidirectional serial/network interface

Like the VI20 but with PPCCOM network options and more standard software modules available.

The VI22 Bidirectional modbus interface

Like the VI21 but only with predefined Modbus network options and more standard software modules available.

The Analog Input /Output cards

For the working with analog value's in the PC20 the following PC20 cards listed in the table below are available. Use the datasheets for the exact wiring and handling these PC20 cards.

AD20	PC20 card analog inputs 8 , 0..10 V , 0..20 mA , 4..20 mA , 4 digits per input 1000bit range, each channel scantime 0,1 ms
AD21	PC20 card analog inputs 8, 0..10 V , 0..20 mA , 4..20 mA , 4 digits per input 1000bit range, each channel scantime 0,1 ms, R.C. time 0,5 ms
AI20	PC20 card analog input 1 ,0..10 V 0..20 mA 4..20 mA, digits per input adjustable 1000bit range/ scantime 0,04 ms
DA20	PC20 card analog outputs 8, 0..10 V, 0..20 mA , 4..20 mA, 4 digits per output 1000bit range
DA21	PC20 card analog outputs 4, 0..10 V, -10V..10V ,max 20 mA , 3 digits per output

The DM20/21/22/23 memory cards

Because the PC20 PLC system is not equipped with a large memory and does not have instructions to handle large quantities of data. If large data handling is required the DM20 /21/22/23 memory cards can be used for these data handling functions.

The minimum configuration consists of the DM20 card in combination of one of the storage cards DM21/22/23, depending on the desired data specification required. The maximum data storage in any configuration is 1Mb. The DM20 controls the data exchange with the PC20 for the data storage modules (DM21/22/23). A 16 word control register that is configured with the MID solder paths on the location of the DM20 is used to define the following elements of the data exchange commands,

- SMA begin address data exchange area.
- SMA end address data exchange area.
- DM memory begin address.
- Control bits for read and write.

This control register is filled in by the PC20 program and the data is exchanged with the DM cards on the by the PC20 defined area's. Often special read and write subroutines are used for exchanging the data , in that case the numeric addressing information is hidden in the subroutines and symbolic indexes can be used for the variables.

REMARK: This feature opens the possibility to read and write in a table based on a variable index. This is a feature that is not foreseen in the PC20 instruction set.

REMARK: Because the DM20 works with a 8Bit data words and the PC20 with 4bit Nibbles it is important always to check the sequence of the LSD and MSD.

DM card cards and attributes are:

DM20	PC20 card Data memory controller 1Mb Max
DM21	PC20 card Data memory storage 64Kram , 64Keprom. Bat backup
DM22	PC20 card Data memory storage 128Kram .Bat backup
DM23	PC20 card Data memory storage 128Keprom
BIDM23	DM20 backpanel for use in a PC20 rack configuration
BIDM20/21/22	DM20 backpanel cable for use in a PC20 rack configuration

The EC20 high speed counter card

Pending.

4. PLC Software and Software Tools

Online changing the PC20 program is possible but not easy, and has to be performed careful in a running application. An error is easily made and an error can easily make the PC20 PLC system crash.

If downloading is necessary the PLC has to be stopped any way. If a short stop is required the swapping of MM.. cards that contain the program is recommendable.

4.1 The Instruction set

In this subchapter the instruction set is explained briefly for a more complete explanation check the PC20 user manual. The function of each instruction is explained and some examples are given how they are normally used. For a short list of the instruction set see Appendix C Many different sequences can be build from a combination of instructions of the PC20. The more common used program sequences will be described in the following sub chapters. Other special tricks s that use the instruction set very effectively like copying 8 Nibbles instead of the Maximum of 4 in one program chain are not explained.

The logic instructions AND, OR, AND NOT, OR NOT, TRIG

In this chapter the logic instructions are explained. Because the PC20 has no brackets to control the sequence of executing the logic instructions, the sequence or chain in which they are placed is eminent for the result of the logic program.

A chain mostly exists of a chain of logic instructions that influences the RR (result register) and is followed by a Execute instruction that copies the RR to the operand of the Execute instruction. If "AND" and "OR" instructions are used in a combination the result can always be seen as a "OR" logic with one output and multiple 'AND" chain inputs.

See Example 10.0 = (0.1*0.2) + (1.0*1.1*1.2) + 2.0

figure

The Software code in the PC20 look like this

1000	AND	0.1
1001	AND	0.2
1002	OR	1.0
1003	AND	1.1
1004	AND	1.2
1005	OR	2.0
1006	EQL	10.0

In the PC20 no other construction is possible. A problem like 11.0 = (3.0 + 3.1) * (4.0 + 4.1) can only be solved by help bits or by rewriting the logic formula

Solution with helpbits:

1000	AND	3.0
1001	OR	3.1
1002	EQL	H.1
1010	AND	3.0
1011	OR	4.1
1012	EQL	H.2
1020	AND	H.1

1021	AND	H.2
1022	EQL	11.0

Solution with rewriting the logic formula:

$$11.0 = (3.0 + 3.1) * (4.0 + 4.1) = 3.0*4.0 + 3.0*4.1 + 3.1*4.0 + 3.1*4.1$$

1000	AND	3.0
1001	ND	4.0
1002	OR	3.0
1003	AND	4.1
1004	OR	3.1
1005	AND	4.0
1006	OR	3.1
1007	ND	4.1
1008	EQL	11.0

Both solutions have disadvantages.

The TRIG or trigger instruction is used to generate a rising or falling edge signal that is only one PLC cycle or scan high.

The bit (operand) used in the TRIG instruction itself keeps the same value as the logic chain of program lines preceding the TRIG instruction but the RR that is affected by the TRIG is only one plc cycle High.

Solution Rising Edge

1000	AND	1.0	
1001	TRIG	100.0	= Copy of 1.0 with one plc cycle delay
1002	EQL	12.0	= Rising Edge Signal of 1.0 one scan high

Solution Falling Edge

1000	AND NOT	1.0	
1001	TRIG	100.1	= Copy of NOT 1.0 with one plc cycle delay
1002	EQL	12.1	= Falling Edge Signal of 1.0 one scan high

Execute instructions EQL,EQL NOT, SET1,SET0, FETCH BIT,STORE BIT

In this chapter the bit execute instructions are explained.

The EQL instruction copies the RR to the Operand following the instruction

The EQL NOT instruction copies the inverse of the RR to the Operand following the instruction

Both these instructions are not depended of the result of RR for its execution

The SET1 instruction sets the SMA operand to "1" if the RR is TRUE

The SET0 instruction sets the SMA operand to "0" if the RR is TRUE

The FETCH BIT instruction copies the SMA operand to the A-Register. First Fetch instruction resets the A-register

The STORE bit is often used to store the result following on arithmetic operations.

If preceded by Fetch instruction FHB,FHD or FHC ,the STORE BIT instruction copies the A-Register bit to the SMA operand. First Fetch instruction resets the A-register

If preceded by shift instruction SFL ,SFR ,the STORE BIT instruction copies the Overflow OVF bit to the SMA operand.

If preceded by compare instruction CMP ,the STORE BIT instruction copies "Compare result = equal" to the SMA operand. "1" is equal.

If preceded by divide instruction DIV ,the STORE BIT instruction copies M/Q Register to the SMA operand.

Example:

1000	AND	0.1	Execute Always
1001	FHC	8	= Copy 8 to A-Register
1002	CMP	1000	= Compare SMA 1000 with A-Register
1003	STB	12.1	= "1" if Compare SMA 1000 = A-Register

Execute instructions FETCH CONSTANT, FETCH DIGIT, STORE DIGIT

In this chapter the Nibble execute instructions that move Nibble data are explained. All of these instructions are only executed if RR is true. And the first of its kind resets the A & B register.

FETCH CONSTANT copies a constant program value to the A-Register (Also the B-register)

FETCH DIGIT copies a Nibble from the scratch path memory to the A-Register (Also the B-register)

STORE DIGIT copies a Nibble from the A-Register to the scratch path memory . If preceded by a COMP/DIV/ADD/SUB/MULT instruction the first will start the actual COMP, ADD,SUB, DIV , MULT and if done the result from A-register or M/Q Register will be copied to the SMA.

Execute instructions ADD, SUBTRACT, DIVIDE, MULTIPLY

In this chapter the Nibble execute instructions that perform arithmetic operations are explained. The actual arithmetic operation is performed on the first STORE instruction. The Reason for the functioning is that up to 4 arithmetic operations can be executed and only if the first STD instruction is performed the CPU knows that no more arithmetic operations will follow.

ADD DIGIT copies a Nibble from the scratch path memory to the B-Register. On the first STD instruction the A-Register + B-Register will be placed in the A-Register.

Example:

1000	AND	0.1	Execute Always
1001	FHC	8	= Copy 8 to A-Register
1002	ADD	1000	= ADD SMA 1000 place in B-Register
1003	STD	1001	= ADD A-Reg + B-Reg = A-Reg and copy A-Reg to SMA

SUB DIGIT copies a Nibble from the scratch path memory to the B-Register. On the first STD instruction the A-Register - B-Register will be placed in the A-Register.

Example:

1000	AND	0.1	Execute Always
1001	FHC	8	= Copy 8 to A-Register
1002	SUB	1000	= SMA 1000 place in B-Register
1003	STD	1001	= SUB A-Reg - B-Reg = A-Reg and copy A-Reg to SMA

DIV DIGIT copies a Nibble from the scratch path memory to the B-Register. On the first STD instruction the A-Register / B-Register will be placed in the M/Q-Register. The Rest value of the deviation will be placed in the A-Register. See PC20 user manual.

Example:

1000	AND	0.1	Execute Always
1001	FHC	8	= Copy 8 to A-Register
1002	DIV	1000	= SMA 1000 place in B-Register
1003	STD	1001	= A-Reg / B-Reg = M/Q-Reg and copy M/Q-Reg to SMA

MUL DIGIT copies a Nibble from the scratch path memory to the B-Register. On the first STD instruction the A-Register * B-Register will be placed in the M/Q-Register. See PC20 user manual for more details.

Example:

1000	AND	0.1	Execute Always
1001	FHC	8	= Copy 8 to A-Register
1002	MUL	1000	= SMA 1000 place in B-Register
1003	STD	1001	= A-Reg * B-Reg = M/Q-Reg and copy M/Q-Reg to SMA

Execute instructions COMPARE, COUNTDOWN, COUNTUP

In this chapter the Nibble execute instructions that perform counter and timer operations are explained.

COMPARE DIGIT compares a Nibble from the scratch path memory with the value in the A-Register. On the first STD instruction the result of comparing the A-Register with the SMA is placed in the COMP-Register will be copied to the SMA following the STD instruction. See PC20 user manual for more details.

Example:

1000	AND	0.1	Execute Always
1001	FHC	8	= Copy 8 to A-Register
1002	FHC	8	= Copy 8 to A-Register
1003	CMP	1000	= Compare 1000 and place in COMP-register
1004	CMP	1001	= Compare 1001 and place in COMP-register
1005	STD	100	= Compare A-Reg with SMA1000/1 = COMP-Reg and copy to 100

The result of the comparison is:

100.0	= 1 Then	A-Register	= SMA (Equal)
100.1	= 1 Then	A-Register	< SMA (Smaller Then)
100.2	= 1 Then	A-Register	> SMA (Larger Then)
100.3	= 1 Then	A-Register	# SMA (Not Equal)

Remark1: The comparison always has to start with the (MSD) the most significant digit

Remark2: Because of the size of the A-register up to 4 COMP instructions are possible in a sequence.

COUNT UP Counts up the value of SMA directly. The actual counting is performed in the M/Q Register and then placed back to SMA. The counting sequence is first performed on the LSD and then on the MSD so overflow will be handled correctly.

Example:

1000	AND	0.1	Execute Always
1001	CNU	1001	= place in M/Q-register +1 back to SMA
1002	CNU	1000	= on Overflow place in M/Q-register +1 back to SMA
1003	STB	14.0	= Overflow(>99) is copied to SMA 14.0

COUNT DOWN Counts down the value of SMA directly. The actual counting is performed in the M/Q Register and then placed back to SMA. The counting sequence is first performed on the LSD and then on the MSD so overflow will be handled correctly.

Example:

1000	AND	0.1	Execute Always
1001	CNU	1001	= place in M/Q-register -1 back to SMA
1002	CNU	1000	= on Overflow place in M/Q-register -1 back to SMA
1003	STB	14.1	= Overflow (<0)is copied to SMA 14.1

Remark1: If not AND 0.1 is used but a clock signal that generates a High edge every second a timer function can be made. This is the way timers are constructed in the PC20.

Remark2: The number of sequential Count instructions is only limited by number of program lines, and not by the size of one of the registers like in the other instructions.

Execute instructions SHIFT LEFT, SHIFT RIGHT

In this chapter the Nibble execute instructions that perform shift operations are explained. SHIFT LEFT Moves the bits in a Nibble one position to the left. From .0 to .3. If .3 bit is high during the Shift and OVF overflow is generated that can activate the next Shift instruction.

Example:

1000	AND	0.1	Execute Always
1001	SFL	20	in SHIFT-register "shift" back to SMA. SMA.3 to OVF
1002	SFL	21	if OVF, in SHIFT-register "shift" back to SMA.3 to OVF
1003	STB	16.0	Overflow is copied to SMA 16.0

SHIFT RIGHT Moves the bits in a Nibble one position to the right. From .3 to .03. If .0 bit is high during the Shift and OVF overflow is generated that can activate the next Shift instruction.

Example:

1000	AND	0.1	Execute Always
1001	SFR	20	in SHIFT-register "shift" back to SMA. SMA.0 to OVF
1002	SFR	21	if OVF, in SHIFT-register "shift" back to SMA. SMA.0 to OVF
1003	STB	16.1	Overflow is copied to SMA 16.1

Remark1: The number of sequential Shift instructions is only limited by number of program lines, and not by the size of one of the registers like in the other instructions.

Jump instructions JUMP TO SUBROUTINE TRUE/FALSE ,RETURN

In this chapter the jump to subroutine are explained.

The JUMP TO SUBROUTINE instruction is limited in the address of the subroutine because in the instruction only 13 bits are available for the subroutine address. This restricts the subroutine begin address area from 0 until 2047.

JUMP TO SUBROUTINE AT FALSE is identical to jump at subroutine at true only the go to condition is inverse.

The RETURN instruction is used to mark the end of a subroutine. It performs a jump back to the line after the line the subroutine call was on.

Example:

1000	AND	0.1	Begin Subroutine
..			
1010	RET		Return to Position Subroutine call was executed
..			
5001	AND	0.1	Condition to go to subroutine
5001	JSAT	1000	Subroutine Call

Remark: The return address is stored in a return stack that is only 4 positions large. So the maximum nesting depth of the subroutines is 4.

Jump instructions JUMP RELATIVE TRUE/FALSE

In this subchapter the jump to subroutine are explained.

The jump relative forward false and jump relative backward false are identical instructions with only one difference that the jump is in an other direction. The Jump to relative instruction is limited in the size of the jump because in the instruction only 13 bits are available for the subroutine address. This restricts the maximum relative jump to 2047 lines. If a larger jump is required a extra jump to a "jump area in between is necessary"

Example:

1000	AND	0.1	Condition
1001	JFRF	199	Relative jump to 1001 + 199

1200	AND	0.1	First line not jumped over
------	-----	-----	----------------------------

Remark1: The RR status if the jump is performed is True.

Remark2: a Jump backward is always with the risk of placing the PLC in a loop that it can not get out. So Caution is recommended when using the jump backwards.

System Instructions END, LAST INPUT OUTPUT and NO OPERATION

In this subchapter the END , LIO and NOP are explained. The instructions LIO and END are used together to define the Inputs and output that need to be updated during the I/O cycle. But also if other data coming from SCADA or an other PLC needs to be updated. Also is it possible by means of these instructions to update data from the I/O in during the PLC execution program.

Example::

0000	AND	0.1
0001	LIO	123
0002	END	120

This is a small program that reads the I/O from SMA 120 until SMA 123. No other inputs and outputs are updated. This functionality is only achieved if both instructions are used combined as shown above. The LIO is often used on the first lines of the program to identification in network environments PPCCOM the value in the LIO instruction identifies the PLC for the network. Also the LIO is often used to place a version number in the PLC for version control.

The NO OPERATION instruction does affect the result of a logic chain in the program.

Example

1000	AND	17.0
1001	AND	17.1
1002	EQL	18.0

Is not the same as:

1000	AND	17.0
1001	NOP	
1002	AND	17.1

1003 EQL 18.0

Remark: The LIO if not used with end is a better No Operation then the NOP instruction itself

4.2Program examples

For the examples PDS35 program examples are used. In the PDS35 program it is possible to use instruction names of different lengths. The name of the instruction can differ from one example to the other. For example the following "dialects" in the instruction set have the same meaning.

Count Up: CU, CNU, CNTU

Shift Right: SFR,SHFTR

Ect.

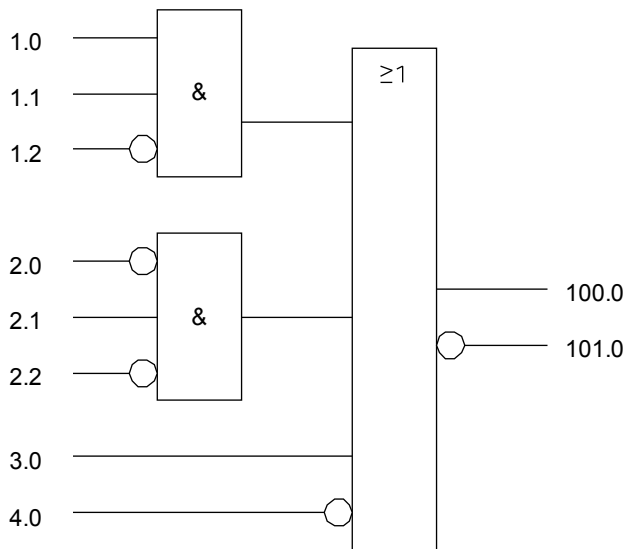
The compiler of the PDS35 program will not except instructions of 5positions long if it is set on instructions of 3positions long and generate an error.

More program examples can be found in the PC20 User manual.

AND , OR EQL

The following logical hardware circuits are programmed with the PC20 instruction set.

EXAMPLE 1:



In formula form:

$$\text{NOT } 101.0 = 100.0 = (1.0 * 1.1 * \text{NOT } 1.2) + (\text{NOT } 2.0 * 2.1 * \text{NOT } 2.2) + 3.0 + \text{NOT } 4.0$$

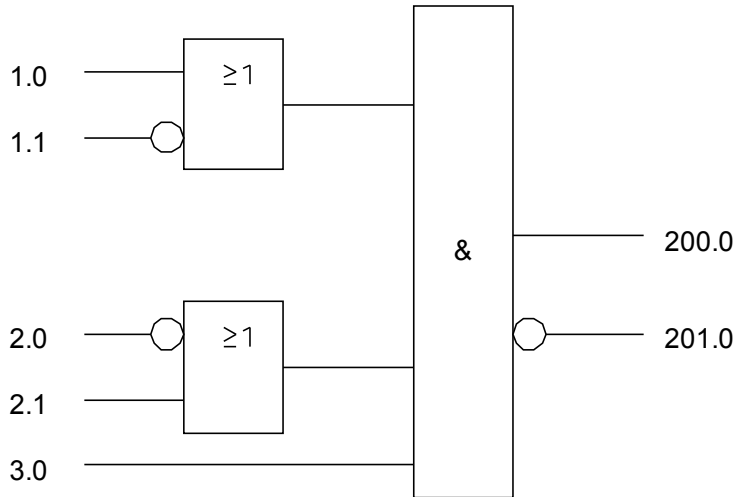
1000	AND	1.0	Start of logic chain
1001	AND	1.1	
1002	ANN	1.2	
1003	ORN	2.0	Start new OR chain
1004	AND	2.1	
1005	ANN	2.2	
1006	OR	3.0	
1007	ORN	4.0	
1008	EQL	100.0	

1009 EQN

101.0

EXAMPLE 2:

The following equivalent of a hardware diagram is more difficult for the PC20 instruction set.



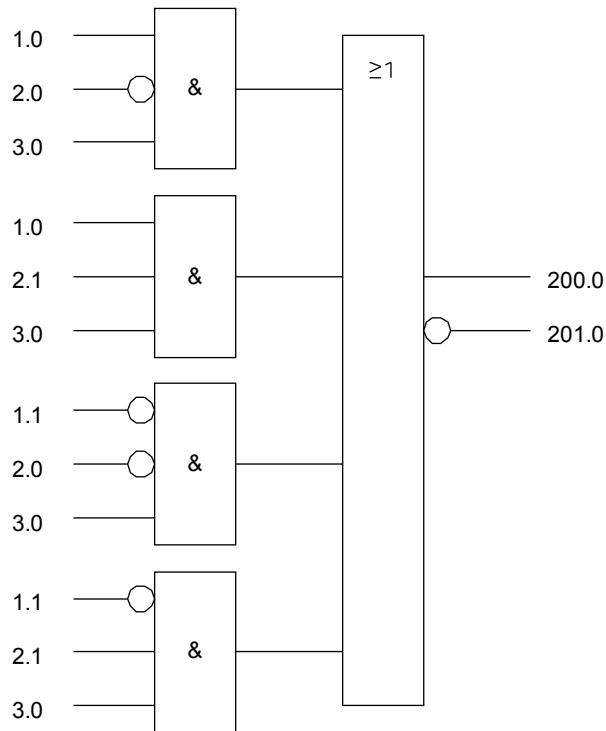
In formula form:

$$\text{NOT } 201.0 = 2100.0 = (1.0 + \text{NOT } 1.1) * (\text{NOT } 2.0 + 2.1) * 3.0$$

The form $(1.0 + \text{NOT } 1.1) * (\text{NOT } 2.0 + 2.1) * 3.0$ has to be converted to

$$=(1.0 * \text{NOT } 2.0 * 3.0) + (1.0 * 2.1 * 3.0) + (\text{NOT } 1.1 * \text{NOT } 2.0 * 3.0) + (\text{NOT } 1.1 * 2.1 * 3.0)$$

The following is the representation of a hardware diagram equivalent that easily can be programmed in the PC20 instruction set. Clear is the disadvantage for the readability of the software program



The software in the PC20 would look like the following program.

1000	AND	1.0	Start of logic chain
1001	ANN	2.0	
1002	AND	3.0	
1003	OR	1.0	Start new OR chain
1004	AND	2.1	
1005	AND	3.0	
1006	ORN	1.1	Start new OR chain
1007	ANN	2.0	
1008	AND	3.0	
1009	ORN	1.1	Start new OR chain
1010	AND	2.1	
1011	AND	3.0	
1012	EQL	200.0	
1013	EQN	201.0	

An other way of programming this logic diagram is making use of help bits. This would result in the following program.

1000	AND	1.0	Start of logic chain 1
1001	ORN	1.1	
1002	EQL	9.0	Help bit end of logic chain1
1003	ANN	2.0	Start of logic chain 2
1004	OR	2.1	
1005	EQL	9.1	Help bit end of logic chain2
1006	AND	9.0	Help bit end of logic chain1
1007	AND	9.1	Help bit end of logic chain2

1008	AND	3.0
1009	EQL	200.0
1010	EQN	201.0

The disadvantage is the need of extra help bits. But the program logic is better to see if it is compared with the original diagram.

REMARK: Other PLC's like the Siemens S7 PLC's interpreted the logic AND , OR structures in a different way and conversion from one to the other system is not easy for that reason. For each chain the actual logic aim has to be checked.

CLOCK TIMING SIGNALS

The Clock signals for timing purposes are available in the SMA addresses 0,3 to 1.3. This bitmap signal specification is different for each PC20 PLC type, please check the specific data sheets before using these bits.

The Clock signal bits in the PC20 Rack model are:

SMA 0.3	= 0.01 Sec	(0.01Sec High / 0.01Sec Low)
SMA 1.0	= 0.1 Sec	(0.1Sec High / 0.1Sec Low)
SMA 1.1	= 1.0 Sec	(1.0Sec High / 1.0Sec Low)
SMA 1.2	= 10 Sec	(10Sec High / 10Sec Low)
SMA 1.3	= 60 Sec	(60Sec High / 60Sec Low)

In this example the following edge signals are programmed:

CLP01S	100mSEC CLOCK PULSE
CLP01S -	100mSEC CLOCK PULSE NEG. EDGE
CLP50MS	50mSEC CLOCK PULSE
CLP1S	1SEC CLOCK PULSE
CLP10S	10SEC CLOCK PULSE
CLP1MIN	60SEC CLOCK PULSE
CLP3S	3SEC CLOCK PULSE

The most of the edges are programmed with the TRG and EQL instructions. But for the 3Sec edge pulse also the Countdown instruction is used.

Symbol list presentation:

EQ	C10MS	=	0.3	CLOCK 10 MSEC.
EQ	C100MS	=	1.0	CLOCK 100 MSEC.
EQ	C1S	=	1.1	CLOCK 1 SEC.
EQ	C10S	=	1.2	CLOCK 10 SEC.
EQ	C60S	=	1.3	CLOCK 1 MIN.

Program example:

```
* *****
HD PLC PULSE CYCLE CLOCKS                                     S
* ++++++
*
* CLOCK PULSE 100 MSEC.
*
01966      AND    C100MS      CLOCK 100 MSEC.
01967      TRG    TRG01      TRIGGER
01968      EQL    CLP01S      100mSEC CLOCK PULSE
*
01969      ANN    C100MS      CLOCK 100 MSEC.
01970      TRG    TRG02      TRIGGER
```

```

01971      EQL    CLP01S-    100mSEC CLOCK PULSE NEG. EDGE
*
* CLOCK PULSE 50 MSEC.
*
01972      AND    CLP01S      100mSEC CLOCK PULSE
01973      OR     CLP01S-     100mSEC CLOCK PULSE NEG. EDGE
01974      EQL    CLP50MS     50mSEC CLOCK PULSE
*
* CLOCK PULSE 1 SEC.
*
01975      AND    C1S         CLOCK 1 SEC.
01976      TRG    TRG03       TRIGGER
01977      EQL    CLP1S       1SEC CLOCK PULSE
*
* CLOCK PULSE 10 SEC. TEMP NOT USED BIT 2.2 USED INTERNAL CP25
* !!! SEE DATA SHEET FOR JUMPER SETTINGS
*
01978      NOP
*01975      AND    C10S        CLOCK 10 SEC.
*01976      TRG    TRG04       TRIGGER
*01977      EQL    CLP10S      10SEC CLOCK PULSE
*
* CLOCK PULSE 1 MIN. TEMP NOT USED BIT 2.3 USED INTERNAL CP25
* !!! SEE DATA SHEET FOR JUMPER SETTINGS
*
01979      NOP
*01978      AND    C60S        CLOCK 1 MIN.
*01979      TRG    TRG05       TRIGGER
*01980      EQL    CLP1MIN     60SEC CLOCK PULSE
*
* CLOCK PULSE 2 SEC.
*
01980      AND    CLP3S        3SEC CLOCK PULSE
01981      OR     BSYSRES      SYSTEM RESET (FIRST PLC CYCLE)
01982      FC     03
01983      SD     CLK3SEC      CLOCK 2 SEC DIGIT
01984      ST0    CLP3S        3SEC CLOCK PULSE
*
01985      AND    CLP1S        1SEC CLOCK PULSE
01986      CD     CLK3SEC      CLOCK 2 SEC DIGIT
01987      SB     CLP3S        3SEC CLOCK PULSE
*

```

COUNTER&TIMER

Because the PC20 instruction set the has no specific timer instruction like other PLC's normally have the timer has to be programmed in the same way as a counter is programmed. In the following example a timer (counter with 100mSec count pulse) that consists of three program parts a preset value (0) part followed by a compare the timer value and a counting up the timer (counter) on the 100mSec clock pulse.

The timer is started if P51RTM = "TRUE" (1)

The timer is elapsed if P51TIM P51 = "TRUE" (1)

A TRG function is used with DUM10_0 in order to reset the timer (counter) value on timer is elapsed.

REMARK: The CNU and the CMP instruction sequence is always performed with the LSD first and the MSD as last in the sequence.

*** PART 1 RESET STEP TIME OUT TIMER**

05700	ANN	P51TIM	P51 TIME OUT STEP ENDED
05701	TRG	DUM10_0	PS DUMMY TRIGGER
05702	FHC	0	
05703	STD	P51TO0	P51 STEP TIMEOUT 10^0
05704	STD	P51TO1	P51 STEP TIMEOUT 10^1
05705	STD	P51TO2	P51 STEP TIMEOUT 10^2

*

*** PART 2 TEST ON TIME OUT**

05706	AND	P51RTM	P51 TIME OUT STEP ON
05707	FHD	P51PT0	P51 PRESET STEP TIMEOUT 10^0
05708	FHD	P51PT1	P51 PRESET STEP TIMEOUT 10^1
05709	FHD	P51PT2	P51 PRESET STEP TIMEOUT 10^2
05710	CMP	P51TO0	P51 STEP TIMEOUT 10^0
05711	CMP	P51TO1	P51 STEP TIMEOUT 10^1
05712	CMP	P51TO2	P51 STEP TIMEOUT 10^2
05713	STB	P51TIM	P51 TIME OUT STEP ENDED

*

*** PART 3 TIME OUT TIMER**

05714	AND	P51RTM	P51 TIME OUT STEP ON
05715	ANN	P51TIM	P51 TIME OUT STEP ENDED
05716	AND	CLP01S	100mSEC CLOCK PULSE
05717	CNU	P51TO0	P51 STEP TIMEOUT 10^0
05718	CNU	P51TO1	P51 STEP TIMEOUT 10^1
05719	CNU	P51TO2	P51 STEP TIMEOUT 10^2

SUBROUTINES

Subroutines are "small PLC programs" that perform a specific function that is more often used in the PLC program or that contain complex calculations or actions that would only make the main program less readable and are placed in a subroutine for that reason.

In the program a call to the subroutine is placed. The program continues with the subroutine program and when ready the program continues on the program line following the line the subroutine call was on.

Example:

1000	AND	0.1	Begin Subroutine
..			
1010	RET		Return to Position Subroutine call was executed
..			
5001	AND	0.1	Condition to go to subroutine
5001	JSAT	1000	Subroutine Call

Often used subroutines are:

00157	SUB. COPY A-REGISTER TO MAIN PARAMETER NUMBER	A
00161	SUB. COPY A-REGISTER TO SUB PARAMETER NUMBER	A
00165	SUB. CALCULATE DM ADDRESS WITH PARAMETER NUMBER AS INPUT	000000 A
00227	SUB. CALCULATE DM ADDRESS OF THE SCREEN NO OFFSET	A
00274	SUB. WRITE DATA FROM A SPECIFIED SMA AREA TO THE DM	A
00298	SUB. READ PARAMETER FROM THE DM TO THE SPECIFIED SMA AREA	A

00307	SUB. DECO ROUTINE (1 DIGIT HEX --> 1 BIT OUT OF 16	S
00332	SUB. DECODE 2-DIGIT BCD INPUT NUMBER TO 1 OUT OF 64 BITS	S
00402	SUB. STEP UP STEPCOUNTER OR GO TO STEP	S
00525	SUB. INCREMENT THE CORRESPONDING PROGRAMSTEPS	S
01275	SUB. SUBROUTINE ERROR HANDLER MSH (16-BIT)	S
01315	SUB. SUBROUTINE ERROR HANDLER MSH1 (4-BIT)	S
01345	SUB. SUBROUTINE ERROR HANDLER MSH2 (1-BIT)	S
01431	SUB. CALCULATE DM-INDEX SHIFT	010694 S
01454	SUB. WRITE SCREEN-DATA SHIFT	150794 S
01478	SUB. READ SCREEN DATA SHIFT	010694 S
01503	SUB. WRITE SCREEN DATA FROM SHIFT TO RS20 NEXT SECTION	240694 S
01528	SUB. WRITE SCREEN DATA FROM RS20 PREV SECTION TO SHIFT	240694 S
01706	SUB. CALCULATE DM ADDRESS WITH DISPLAY INFO NUMBER AS INPUT	A
01730	SUB. READ DISPLAY INFO FROM DM INFO TABLE INTO SMA AREA	A
01734	SUB. WRITE DISPLAY INFO FROM SMA AREA TO THE DM INFO TABLE	A
01739	SUB. SET BIT 0 IN ERROR BIT TABLE	A
01759	SUB. RESET BIT 0 FROM DIGIT	A
01783	SUB. DETMOD DETERMINE MODE OF AUTOMATION MODULE	A
01834	SUB. COMPARE TWO TEMPERATURES	230794 A
01875	SUB. CHUCK DRIVER	140994 A
01904	SUB. LIFT DRIVER	050894 A

This list of subroutines is generated with the use of the PDS35 header function. Clear to see that the PC20 uses a lot of subroutines to build the required functionality. But important is that is made clear how the subroutines work otherwise the advantage of multiple use is lost in the extra time is put in to understand the functioning of the subroutines.

REMARK: The use of many subroutines causes a longer PLC Cycle time.

REMARK: The nesting of subroutines is only allowed 4 deep. That means that the calling of subroutines within subroutines before the previous subroutine RET function was encountered is only allowed 4 times.

CALCULATIONS

Calculating in the PC20 PLC instruction text is complex if larger numbers are used in the calculation. For calculation the PC20 PLC can be compared with a microprocessor, all the calculations are possible but all exceptional situations in the calculation need to be checked in the software and appropriate action needs to be taken to achieve the correct result. And all of this has to be done in the PC20 application software.

The calculations are often done in subroutines. To show the complexity of the calculations the following program is an example of a calculation of an index value for reading /writing data from a DM card ,that needs one multiply and one addition.

```

* *****
HD SUB. CALCULATE DM-INDEX SHIFT                                010694 S
* ++++++
*           IN : SHIFREGISTER NUMBER 2 DIGITS IN A-REGISTER (00-30)
*           : DM OFFSET 0009000 - 0009999
*           OUT: DM-ADDRESS IN DM-REGISTER DH EN DL
*
*           ADDRESS =(22*DISPLAY INFO NUMBER) +DM OFFSET(0009000)
*
*
01431  CALCIDAND    ALWAYS    ALWAYS ACTIVE=(1)

```

01432	MUL	N2	NUMBER 2
01433	MUL	N2	NUMBER 2
01434	STD	DDUM_00	DUMMY
01435	ADD	N0	NUMBER 0
01436	ADD	N0	NUMBER 0
01437	ADD	N0	NUMBER 0
01438	ADD	N9	NUMBER 9
01439	STD	DMCR14	DESTINATION ADDRESS 10^0
01440	STD	DMCR13	DESTINATION ADDRESS 10^1
01441	STD	DMCR12	DESTINATION ADDRESS 10^2
01442	STD	DMCR11	DESTINATION ADDRESS 10^3
01443	FHC	0	
*			
01444	AND	OVERFL	ARITHMATIC OVERFLOW
01445	FHC	1	
*			
01446	AND	ALWAYS	ALWAYS ACTIVE=(1)
01447	ADD	DM_OFF4	DM OFFSET 10^4
01448	ADD	DM_OFF5	DM OFFSET 10^5
01449	ADD	DM_OFF6	DM OFFSET 10^6
01450	STD	DMCR10	DESTINATION ADDRESS 10^4
01451	STD	DMCR09	DESTINATION ADDRESS 10^5
01452	STD	DMCR08	DESTINATION ADDRESS 10^6
01453	RET		

For explanation of the calculation possibilities check the PC20 user manual.

4.3The program layout

That PC20 PLC consists of one large list of instructions that are executed over and over again each PLC cycle. This is a not always a desirable situation. Often some parts of the software are activated only during start up and others are only activated if the start up routines were executed successfully.

For the explanation of the program structure the possibilities to use labels to jump to instead of absolute line numbers to jump to as used in PDS35 are used.

REMARK: The first Cycle of the PC20 program the RR line number 0 is true, all other plc cycles the RR will be false.

In the PC20 user manual these program setups are explained with more detailed information.

Standard program structure

Simple PLC program set up with a initialization program part.

0000	JFRF	MAIN	
			Initialization Program
0010	MAIN	AND	0.1
1000	LIO	123	
1001	END	100	

Only the first PLC scan the plc lines 1 to 9 are executed. On line 1001 following the END instruction the program counter will continue on line number 0000. But because RR of line 0000

is only true the first Plc cycle the program counter will continue on line number 10 were the MAIN program is located.

Often used program structure

This program layout uses the feature that if more then 5 subroutine calls are generated the first in is lost. What in this case does not matter because this call in on line number 0000 and does not need to be returned to.

Advantage of the structure is the possibility to execute reading inputs and writing outputs can be done whenever desired.

0000		JSAT	INIT	
0001		RET		
				Area with LIO/END instructions
0004	I_I	LIO	b_I	reading inputs
0005		END	e_I	
0006	O_O	LIO	b_O	writing outputs
0007		END	e_O	
0010	INIT	AND	0.1	
				Initialization Program
0099		JFRF	MAIN	
				Subroutine Area
1000	MAIN	AND	0.1	
				End of program
9000		AND	0.1	
9001		JSAT	I_I	
9002		JSAT	O_O	
9003		JSAT	MAIN	
9010		END	0	
9011		END	0	

The JSAT call to I_I (also O_O) will execute the LIO and END and will go to line number 0000 it will not execute the JSAT on line number 0000 but it will execute the RET on line 0001 and the line number following on which the JSAT I_I is located.

The JSAT call to MAIN will go to the line number the MAIN label is placed on and start the Main Program. The return line number will be placed in the return stack but it will never be retrieved from the stack because it is not needed for further continuing of the program.

In this way it is possible to read the inputs , execute the Main program and then write the outputs. Otherwise the dilemma can be first write outputs while the inputs are red but not used in the main program. (unknown). Or use unreliable inputs in the main program and write outputs on these unreliable inputs.

REMARK1: It is also possible to place a LIO *** on line 0000. The JSAT is then on line 0001. In this situation a identification in line 0000 is placed for a SCADA or an other master on the network. Or a version number of the program itself can be placed there. For the functionality of the program it does not matter.

REMARK2: The double END instruction is used if an online search is performed then the search stops at the double END. Otherwise the search will continue until the end of the 16K program.

Start up delay

If a start up delay is required during switching on of the 24VDC switched to sensors and actuators, a delay timer can be placed in the initialization program. See following example.

```
*
*
* -----
* DELAY AFTER START OF PC20
* -----
*
00091      AND    ALWAYS          ALWAYS ACTIVE=(1)
00092      FHC     5
00093      STD     POWONT          POWER ON TIMER
00094      ST0      S1OPOT         OUTPUT POWER ON TIMER
00095 JB1   AND     C100MS        CLOCK 100 MSEC.
00096      TRG      TRG13         TRIGGER
00097      CND      POWONT         POWER ON TIMER
00098      STB      S1OPOT         OUTPUT POWER ON TIMER
*
00099      AND     S1OPOT         OUTPUT POWER ON TIMER
00100      JBF      JB1
```

The program will be locked in this loop until the time is elapsed. During this time other parts of the equipment can start up, and when they are ready the PC20 will start.

4.4PDS35

TeHa developed the PDS35 software support program for generating a source codes that could be converted to object programs and could be downloaded and debugged by one program. Also many functions like Cross references, program contents generation, program comparison and the generation of documentation of program files are available in PDS35. Also making online changing is possible in PDS35 but .

In the PDS35 program it is possible to change the setting for all kind of types of the pc20 PLC family.(e.g. PC20, MC30, MC41 in ram and rom settings)

4.5 Developed Software tools by third parties

VHE developed many software tools during the period that many PC20 PLC projects were executed by VHE engineers. The following programs can be very helpful during creating and debugging an automation project. However most of the VHE programs are quicker then the PDS35 programs they do not always give the correct error special during communication , and mostly they only work for the PC20 rack version.

By VHE developed PC20 Support programs:

Qload = Uploading object from PC20

Qdump = Downloading object into PC20

Qcompare = Comparing object programs

A complete development system with a cross reference more complete the TeHa crossreference.

Other support programs are Fusion general development system used for PC20 and S5 (Pulse).

And PDS5 a graphic development system used by some Philips departments.

TopPromisys was developed by TeHa and had extra features like a oscilloscope function and the possibility to online monitor with use of the symbolic names from the source program.

Moni.exe was developed by I.Verijdt to be able to monitor data and change data simultaneously. This is only possible with the other tools from VHE and TopPromisys. The advantage of Moni is that it is easy to use and has the possibility to store monitor configuration settings that can be used again a next time (Like VAT table from S7)

5. Converting PC20 to S7 (or other PLC,s)

At the moment there is no conversion tool for converting PC20 to S7. The best way to convert PC20 source code to a S7 program, is to copy the PC20 source to S7 source and compile the source to a S7 object program.

The following differences have to be taken in account:

The AND ,OR logical structure differs and if complex structures are used they have to be checked.

The PC20 is a fast PLC and the S7 you choose to replace the PC20 has to be able to be just as fast. Therefore it is important that the other activities like the data exchange with a HMI or Scada do not take too much of the processor time of the S7 CPU. This can be configured in the S7 system configuration.

The FHD is a conditional instruction while the comparable S7 instruction LOAD is a non conditional instruction. This will ask for a specific solution for each PC20 program part that uses FHD instructions or replace the FHD by a combined JUMP and LOAD instruction over the non conditional S7 program lines.

Conversion is therefore a time consuming activity that needs knowledge of the old PC20 programming system and knowledge of the new programming system. This often doubles the work load because two programs have to be studied and checked carefully that the functionality of the systems are the same.

6.Working online with the PC20

The working online with the PC20 is complex and changes online should only be done as you fully understand the instructions you want to change. If you are able to make small changes online then that is a large contribution of the possibilities you have in working with the PC20 PLC.

The original way to work online with the PC20 is with the PU30 via the PU21 card or a TTY terminal monitor that was connected to the PC20 via the CI20 card. Because these working methods had no possibilities to work symbolic and had no good backup system.

Therefore development tools were created to be able to monitor and debug the PC20 programs online. Some of the programs are:

PDS5	Development tool
PDS35	For documentation and debugging using Kermit communication program.
TTY	VHE developed tool
MONI	Data Monitor program with terminal mode with Kermit
TopPromisys	With Scope option and Symbolic monitoring option

Because the Kermit like solutions are most like the TTY monitor presentation. The explaining of working online is based on these programs

The following online actions are interesting to use for debugging when working directly online. These are not all online commands because the commands can be different depending on the medium (PDS35, Moni , Kermit) that is used.

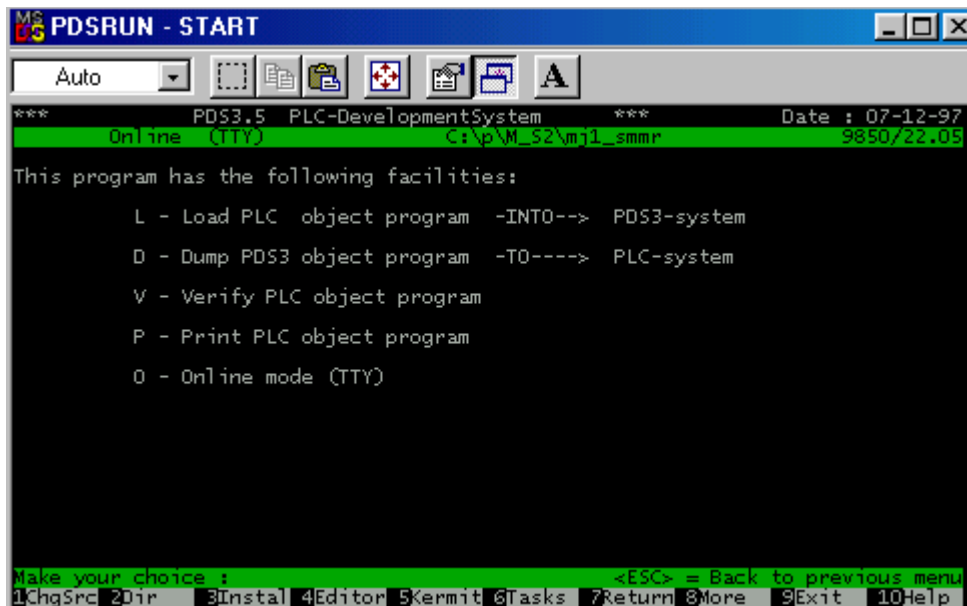
Online commands for MC30/31 and MC40/41 can be different and may have restrictions.

If the PU20 is used a step by step mode is also possible.

Command	Description	action
DL <pma>	DUMP LINE from 0 (RETURN then next 16)	Present program from line 0 -16
ML <pma num cond>	MONITORLINE	
ML 0 16 2	MONITOR LINE from 0-16 condition always	Monitors lines 0 -16
ML 0 16 1	MONITOR LINE from 0-16 condition true	Monitors lines 0 -16 If RR True
MD <SMA num>	MONITOR DATA	
MD 0 24	MONITOR DATA from Nibble 0 until 24	Monitors data 0 -24
RN	RUN	
RS	STOP	
ED	EDIT	Stop PLC and to edit mode
INS <pma inst SMA>	INSERT LINE	Insert line PLC in stop
DEL <pma>	DELETE LINE	Delete line PLC in stop
WL <pma inst SMA>	WRITE LINE	
DL <pma>	DUMP LINE	
WD <SMA value>	WRITE DATA	Write SMA-data

6.1PDS 35 online monitoring

In PDS35 the online TTY mode has the following online options that are shown. A disadvantage is that the communication is slow because the software refreshes the screen after each communication action.

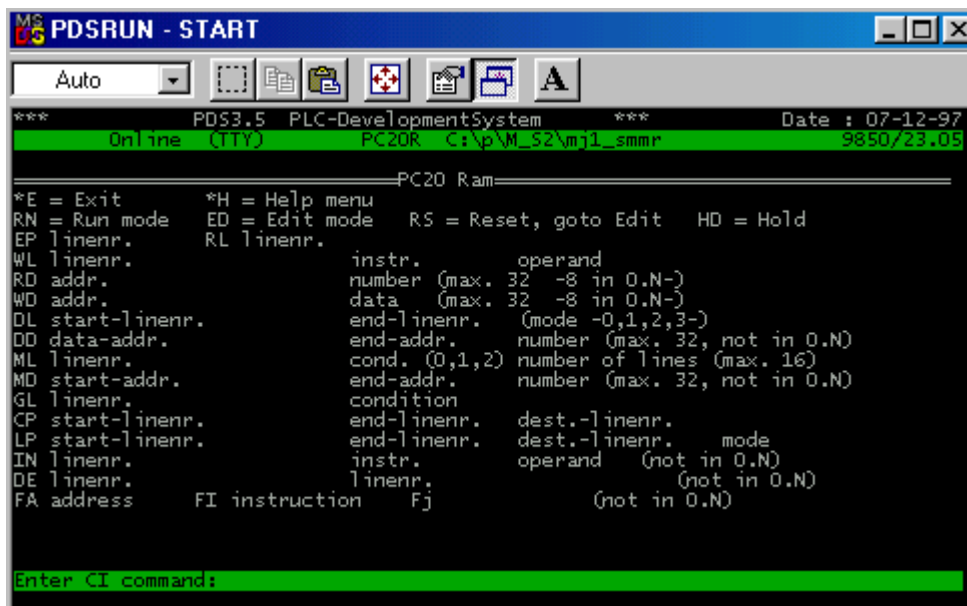


```

MS-DOS PDSRUN - START
Auto
*** PDS3.5 PLC-DevelopmentSystem *** Date : 07-12-97
Online (TTY) C:\p\M_S2\mj1_smmr 9850/22.05
This program has the following facilities:
  L - Load PLC object program -INT0--> PDS3-system
  D - Dump PDS3 object program -T0----> PLC-system
  V - Verify PLC object program
  P - Print PLC object program
  O - Online mode (TTY)
Make your choice : <ESC> = Back to previous menu
1ChgSrc 2Dir 3Instal 4Editor 5Kermit 6Tasks 7Return 8More 9Exit 10Help

```

In the PDS35 program the online function has a help option in which all possible commands are presented.



```

MS-DOS PDSRUN - START
Auto
*** PDS3.5 PLC-DevelopmentSystem *** Date : 07-12-97
Online (TTY) PC20R C:\p\M_S2\mj1_smmr 9850/23.05
-----PC20 Ram-----
*E = Exit      *H = Help menu
RN = Run mode  ED = Edit mode  RS = Reset, goto Edit  HD = Hold
EP linenr.    RL linenr.
WL linenr.    instr.      operand
RD addr.      number (max. 32 -8 in 0.N-)
WD addr.      data (max. 32 -8 in 0.N-)
DL start-lin. end-linr.   (mode -0,1,2,3-)
DD data-addr. number (max. 32, not in 0.N)
ML linenr.    cond. (0,1,2) number of lines (max. 16)
MD start-addr. end-addr.  number (max. 32, not in 0.N)
GL linenr.    condition
CP start-lin. end-linr.   dest.-linr.
LP start-lin. end-linr.   dest.-linr.   mode
IN linenr.    instr.      operand (not in 0.N)
DE linenr.    linenr.     (not in 0.N)
FA address    FI instruction Fj (not in 0.N)
Enter CI command:

```

For example command "DL 0" presents the first 0-16 program lines. Each return then presents the next 16 program lines

Command "ML 0 16 2" monitors the lines 0-16.

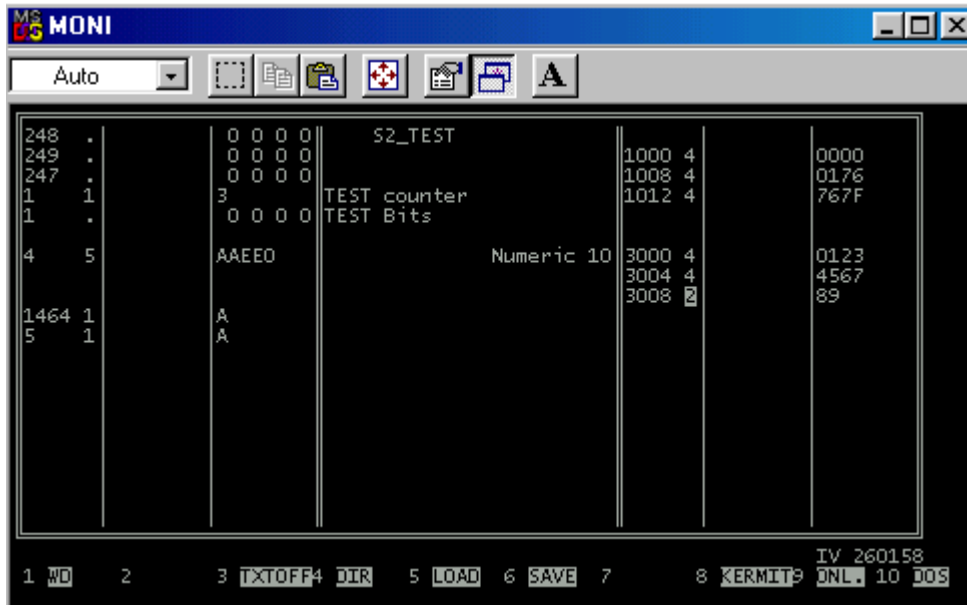
Back to PDS35 is with the keys <ESC> <E> <RETURN>

This differs from the normal terminal instructions

6.2MONI online monitoring

Moni is a standalone program that is used for monitoring and debugging. Moni is no development tool and has no symbolic representation and documentation possibilities.

But Moni is a helpful tool for setting up a testing configuration that can be stored and reused. The start up screen tries to connect directly to the PC20 and if this is the case a screen like seen in the following picture is presented



It is also possible that the middle colon presents no text but also online data. This is an option that can be selected with the F3 Function key. Normally the screen is divided in 3 identical colons. Each colon consists of a SMA address, write SMA data area, actual SMA value. With a arrow keys the Cursor can be placed on that field we want update or force.

If the cursor is in the correct positions. The function keys have the following functions:

- F1 Write Data
- F3 Text on off toggle for the middle colon.
- F4 Directory where MONI configuration file can be stored.
- F5 Load a MONI configuration file
- F6 Save a MONI configuration file
- F8 Goto TTY terminal mode
- F9 Switch online on / off toggle
- F10 Stop MONIi Program back to DOS
- <TAB> Reset input field

Presentation:

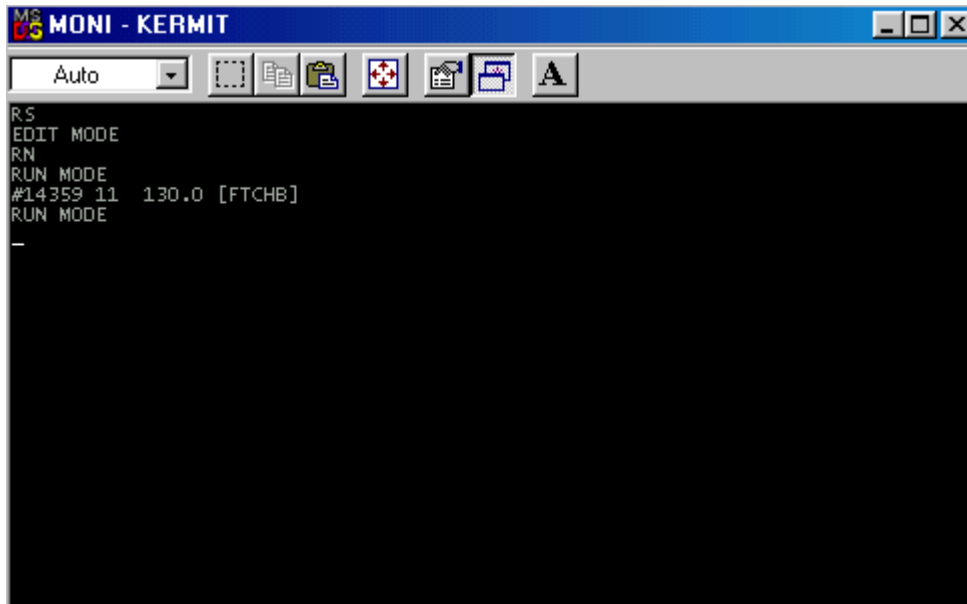
"1-8" Number of nibbles to present as Digit

". " Bit presentation of one Nibble

Write value's:

0-9,A-F Hex value's

In the following picture The TTY monitor is presented following on the <F8> key.



```

MS MONI - KERMIT
Auto
RS
EDIT MODE
RN
RUN MODE
#14359 11 130.0 [FTCHB]
RUN MODE
-

```

On this screen the following commands were executed.

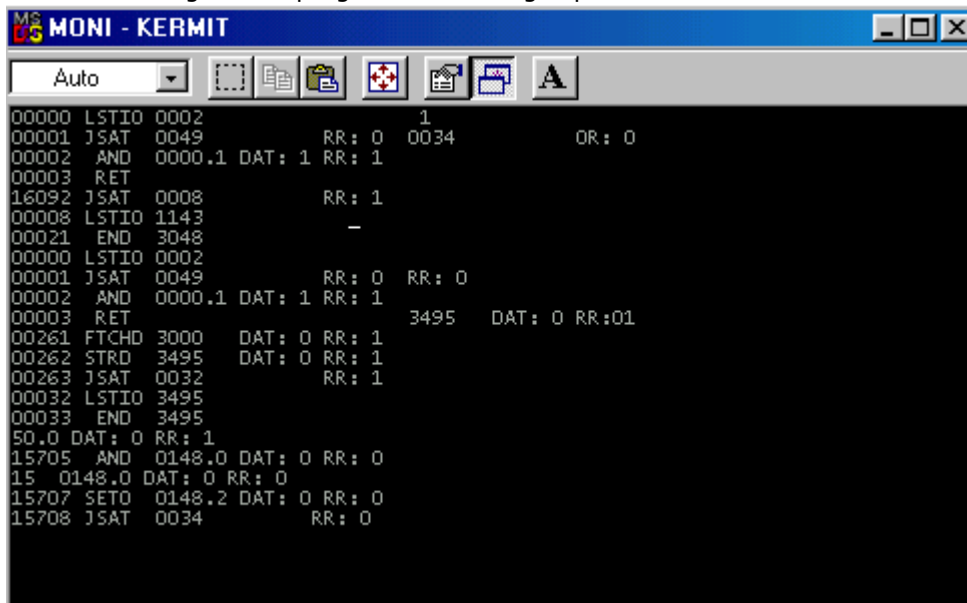
RS RESET stops the PLC in edit mode

RN Places the PLC back in RUN

FA 130.0 FIND ADDRESS if SMA 130.0 (command overwritten by result)

When search for SMA 130.0 is ended the RUN MODE is presented in the screen.

In the Following screen program monitoring is presented.



```

MS MONI - KERMIT
Auto
00000 LSTIO 0002 1
00001 JSAT 0049 RR: 0 0034 OR: 0
00002 AND 0000.1 DAT: 1 RR: 1
00003 RET
16092 JSAT 0008 RR: 1
00008 LSTIO 1143 -
00021 END 3048
00000 LSTIO 0002
00001 JSAT 0049 RR: 0 RR: 0
00002 AND 0000.1 DAT: 1 RR: 1
00003 RET 3495 DAT: 0 RR: 01
00261 FTCHD 3000 DAT: 0 RR: 1
00262 STRD 3495 DAT: 0 RR: 1
00263 JSAT 0032 RR: 1
00032 LSTIO 3495
00033 END 3495
50.0 DAT: 0 RR: 1
15705 AND 0148.0 DAT: 0 RR: 0
15 0148.0 DAT: 0 RR: 0
15707 SETO 0148.2 DAT: 0 RR: 0
15708 JSAT 0034 RR: 0

```

The "ML 0 16 2" is presented in the previous screen dump.

Back to MONI with the following key combination.

<Ctrl><]> , <C> , <Q> , <RETURN>

This key combination is specific for the Kermit version used with MONI.

6.3PC20 online tools for Windows XP and W2000

Because the problems that existing program development tools do not work any more, development of new tools are under construction. A monitor program and a download and upload program are being developed for Windows XP and windows 2000 professional, but they are still under construction.

An other option is using a virtual PC program and Run Windows 95/98 under Windows XP or windows 2000.

7.APPENDICES

7.1Appendix A PLC Hardware environment in Source code

```
GP *****
GP * PLC HARDWARE ENVIRONMENT INFO:
GP *****
GP  PC20 CARD RACK(1)      ADRESS  SETTINGS  REMARKS:
GP   1 CI21                9600/E/2/1
GP   2 MM26 16K EEPROM      RAM=ON
GP   3 CP25 SMA2K4          16K
GP   4 IM20 16IN            004.0 - 007.3  NOT SWITCHED
GP   5 OM22 32OUT           008.0 - 015.3  "
GP   6 IM22 32IN            016.0 - 023.3  "
GP   7 OM22 32OUT           024.0 - 031.3  "
GP   8 IM22 32IN            032.0 - 039.3  "
GP   9 OM23 16OUT           040.0 - 043.3  "
GP  10 OM22 32OUT           048.0 - 055.3  "
GP  11 OM22 32OUT           056.0 - 063.3  "
GP  12
GP  13
GP  14 AD20 PS01  32D I    1320 - 1351  MID 1001  TEMP MEASUREM.
GP  15 VI21/1    NETWORK  2256 - 2319  148      CAM
GP  16 VI21/1    NETWORK  2320 - 2383  152      INFO SYSTEM
GP  17 RS20 SECT 1 64D I/O 1000 - 1063  MID 1001  PREV. SECTION
GP  18 RS20 SECT.3 64D I/O 1064 - 1127  MID 1001  NEXT. SECTION
GP  19 VI21/1    ECU      148.0 - 151.3  ECU
GP  20 DM20              3480 - 3495
GP  21 DM21 128BYTES      (IN PROGRAM)  BAT=ON    64KRAM/64KEEPROM
GP
GP
GP  PC20 CARD RACK(2)      ADRESS  SETTINGS  REMARKS:
GP   1 IM22 32IN           064.0 - 071.3  NOT SWITCHED
GP   2 IM22 32IN           072.0 - 079.3  "
GP   3 IM22 32IN           080.0 - 087.3  SWITCHED
GP   4 IM22 32IN           088.0 - 096.3  "
GP   5 IM22 32IN           096.0 - 103.3  "
GP   6 IM22 32IN           104.0 - 111.3  "
GP   7 IM22 32IN           112.0 - 115.3  "
GP   8
GP   9
GP  10
GP  11 OM23 16OUT          120.0 - 123.3  NOT SWITCHED
GP  12 OM23 16OUT          124.0 - 127.3  "
GP  13 OM23 16OUT          128.0 - 131.3  "
GP  14 OM23 16OUT          132.0 - 135.3  "
GP  15 OM23 16OUT          136.0 - 139.3  SWITCHED
GP  16 OM23 16OUT          140.0 - 143.3  "
GP  17 OM23 16OUT          144.0 - 147.3  NOT SWITCHED
GP  18
GP  19
GP  20
GP  21
```

```

GP
GP
GP
NP
GP *****
GP * PROGRAM INFO:
GP *****
GP ADDRESSING THE DM20
GP
GP   DM20 CONTROL REGISTER
GP   =====
GP   3480 MSD LOW SMA ADDRESS
GP   3481
GP   3482
GP   3483 LSD LOW SMA ADDRESS
GP   3484 MSD HIGH SMA ADDRESS
GP   3485
GP   3486
GP   3487 LSD HIGH SMA ADDRESS
GP   3488 MSD ADDRESS POINTER IN DM
GP   3489
GP   3490
GP   3491
GP   3492
GP   3493
GP   3494 LSD ADDRESS POINTER IN DM
GP   3495 I/O CONTROL (0 =DISABLE, 1 =ENABLE, 2 =WRITE, 3 =READ)
GP

```

7.2Appendix B Abbreviations and name explanations

AD20	PC20 card analog inputs 8 , 0..10 V , 0..20 mA , 4..20 mA , 4 digits per input 1000bit range, each channel scantime 0,1 ms
AD21	PC20 card analog inputs 8, 0..10 V , 0..20 mA , 4..20 mA , 4 digits per input 1000bit range, each channel scantime 0,1 ms, R.C. time 0,5 ms
AI20	PC20 card analog input 1 ,0..10 V 0..20 mA 4..20 mA, digits per input adjustable 1000bit range/ scantime 0,04 ms
AM30	MC30/31 card analog 8in /4out ,0..1V ,0..10V, -10..10V, 0..20mA
AO20	PC20 card analog outputs 1, 0..10 V, 0..20 mA, 4..20 mA, 3 digits per output , conversiontime 500 nS
BCD	Binary Coded Decimal. The data presentation the PC20 uses for calculations.
BIDM21	DM20 backpanel cable for use in a PC20 rack configuration
BIDM23	DM20 backpanel for use in a PC20 rack configuration
CI20	PC20 card serial bidirectional programming interface, RS232/422
CI21	PC20 card serial bidirectional programming/networking interface, RS232 /442
CI30	MC30 card serial bidirectional programming networking interface,RS232/422
CP20	PC20 card CPU 2k16 2 x EPROM 0.25k4 battery ext
CP21	PC20 card CPU 1k16 RAM 0.25k4 battery int
CP22	PC20 card CPU extern,max 8k4 2k4 battery ext. use MM20, MM21, MM22
CP22[16k]	PC20 card CPU extern,max 16k4 2k4 battery ext. MM23 ,MM25
CP24	PC20 card CPU 2k16 RAM 0.25k4 battery ext
CP25	PC20 card CPU 2k16 RAM 2k4 battery ext
DA20	PC20 card analog outputs 8, 0..10 V, 0..20 mA , 4..20 mA, 4 digits per output 1000bit range
DA21	PC20 card analog outputs 4, 0..10 V, -10V..10V ,max 20 mA , 3 digits per

	output
DM20	PC20 card Data memory controller 1Mb Max
DM21	PC20 card Data memory storage 64Kram , 64Keprom. Bat backup
DM22	PC20 card Data memory storage 128Kram .Bat backup
DM23	PC20 card Data memory storage 128Keprom
EC20	High speed counter card
Fusion	PC20/S5 Development software (Pulse Venlo Netherlands)
HMI	Human Machine Interface
IM20	PC20 card inputs 16, 24 VDC, led's off option
IM22	PC20 card inputs 32, 24 VDC, led's upper or lower, switch selectable
IM23	PC20 card inputs 16, 48 VDC, led's
IO30/1	MC30 card Input/output 24in/16out
IO31	MC30 card Input/output 12in/8out, 1A, short circuit proof
Label	Symbolic equivalent of a line number
LSB	Least Significant Bit
LSB	Least Significant Bit
LSD	Least Significant Digit
MC20	Predecessor of MC30/MC31 with 2K16 eprom 0,5K4 SMA 32in/20out
MC30/1	PC20 based microcontroller 2K4, 2K16, 24in ,16out expandable
MC31	PC20 based microcontroller 2K4, 8K16, 24in ,16out, RS232,RS485 PPCCOM onboard, expandable
MC40	PC20 like stand alone microcontroller
MC41	PC20 like stand alone microcontroller
MI20	PC20 card interface between PU20 en MC20/PLC773
MID	Module Identification pads for card addressing
MM20	PC20 card memory 8k16,8 x 2k8 EPROM , EPROM's type 2716.
MM21	PC20 card memory 8k16 RAM, batt. int max 30 hours
MM22	PC20 card memory 4k16 RAM, batt. int max 30 hours
MM23	PC20 card memory 16k16 RAM, batt. int kan maximaal 300 uur
MM25	PC20 card memory 16k16,4 x 8k8 EPROM
MM26	PC20 card memory 16k16,4 x 16k8 EEPROM
Modbus	Modicon PLC network protocol
MONI.EXE	DOS PC20 Monitor Program written by I.Verijdt
MSD	Most Significant Digit
NIBBLE	4 Bit "word" size used by the PC20
OM20	PC20 card outputs 16, 24 V, 0.5 A , max. 6A per module
OM21	PC20 card outputs 8, 24 V, 2 A, max. 8A per module
OM22	PC20 card outputs 32,30 V,100 mA ,Floating (gnd) outputs
OM23	PC20 card outputs 16, 60 V, 0.5 A,max. 6A per module
PDS35	PC20 Development software (TeHa Netherlands)
PENI.EXE	DOS PC20 Monitor Program for "fast" PC's (Pentium) written by I.Verijdt
PLC	Programmable Logic Controller
PLC773	Rack card version MC20 with 2K16 eprom 0,25K4 SMA 32in/20out
PPCCOM	Is a data communication protocol used for data exchange between networked PLC's in a master / slave configuration . PPCCOM supports RS232 and RS485 (point to point and multi drop) asynchronous data communication.
PPCCOM	Is a data communication protocol used for data exchange between networked PLC's in a master / slave configuration . PPCCOM supports RS232 and RS485 (point to point and multi drop) asynchronous data communication.
PU20/2	MC30/PC20 Desk top programming unit
PU21	PC20 card interface between PU20 and PC20.
PU23	PC20 card interface between PU20 and PC20/MC20/PLC773.
PU30	MC30/PC20 Handheld programming unit
RP20	PC20 card parallel bidirectional interface, for displays en thumbwheel switches

RP30	MC30 card bidirectional Parallel I/O bus
RR	Result register of logic program result. Used to store result and enable following instructions.
RS20	PC20 card serial bi-directional interface, for communication between PC20's, 3 modes (master, slave, passive slave)
RSE	Reset Central Processor (If activated while RSME enable then reset SMA)
RSME	Reset Scratchpad Memory Enable
SCADA	Supervisory Control And Data Acquisition "Data presentation and handling"
SCIO	Separate configuring code for Inputs and Outputs for card addressing
SFC	Sequence Flow Control "Step program"
SM	Scratchpad Memory
SMA	Scratchpad Memory Address
SMA	Scratchpad Memory Address
SO20	PC20 card power+outputs 8, 24 V, 0.5 A ,max. 1A per module
TeHa	A software company that build many software tools for PC20 programming. The company does no longer exist.
VI20	PC20 card serial bi-directional interface, prog. mem. 2k16,scratchpad mem. 0.25k4 RS232
VI21	PC20 card serial bi-directional interface, prog. mem. 2k16,scratchpad mem. 0.25k4 RS232/442 PPCCOM
VI22	PC20 card serial bi-directional interface, for MODBUS interface RS485

7.3 Appendix C Short Instruction Set PC20

Short Description Instruction set PC20 PLC

INSTRUCTION	NO :	DESCRIPTION	RR	ACTIONS IN CPU PC20:	USED:
NOP	00	NO OPERATION	-	RESET RESULT OF LOGICAL RUNG	-
TRG SMA.0	01	EDGE PULSE	/	CYCLE HIGH EDGE PULSE(USED WITH EQL/N)	BEFORE EQL
EQL SMA.0	02	EQUAL	-	RR-> SMA. (COPY RESULT TO SMA)	TRG/RESULT
EQN SMA.0	03	NOT EQUAL	-	NOT RR-> SMA. (COPY INVERSE RESULT TO SMA)	TRG/RESULT
				TRG SMA.0 SMA.0 = RR	
				EQL SMA.1 1CYCLE HIGH ALS RR IS HIGH	
SFL SMA	04	1 BIT LEFT	1	SHIFT 1 BIT LEFT IN SMA SMA.3->OVF	GENERAL
SFR SMA	05	1 BIT RIGHT	1	SHIFT 1 BIT RIGHT IN SMA SMA.0->OVF	GENERAL
				SFL SMA1 1ST RESET OVF REG.(0.0)	
				SFL SMA2 OVF SMA1 -> SMA2.0 (SFR.3)	
				STB SMA.0 OVF SMA2 -> SMA.0	
CND SMA	06	COUNT DOWN	1	SMA=SMA-1 SMA->RE.M/Q-1 9=CARRY->SMA	BEFORE STR.
CNU SMA	07	COUNT UP	1	SMA=SMA+1 SMA->RE.M/Q+1 0=CARRY->SMA	BEFORE STR.
				CNU SMA1 RR=1 SMA1+1 OVF-> STATE	
				CNU SMA2 STATE=1 SMA2+1 OVF->STATE	
				STB SMA.0 STATE->SMA.0	
ST0 SMA.0	08	SET 0	1	SMA.= 0	OUT
ST1 SMA.0	09	SET 1	1	SMA.= 1	OUT
STB SMA.0	10	STORE BIT	1	REG A->SMA.0 NA FHD FHB FHC 1ST START	TEL/COMP
				OVF-> SMA.0 NA SFL SFR CALCULATION	

INSTRUCTION	NO :	DESCRIPTION	RR	ACTIONS IN CPU PC20:	USED:
				STATE-> SMA.0 NA CND CNU	
				COMP.0->SMA.0 NA CMP	
FTB SMA.0	11	FETCH BIT	1	(16)SMA.->REG A&B 1ST CLEARS A&B	TEL/COMP
FHC CONST	12	FETCH CONST	1	(4)CONST ->REG A&B 1ST CLEARS A&B	TEL/COMP
FTD SMA	13	FETCH DIGIT	1	(4)SMA ->REG A&B 1ST CLEARS A&B	TEL/COMP
STD SMA	14	STORE DIGIT	1	(4)REG A-> SMA. 1ST START BEREK.	TEL/COMP
				COMP->SMA NA CMP	
CMP SMA	15	COMP.IN NIBLE	1	COMP REG A WITH SMA RES.IN SMA	BEFORE STR.
				FHD SMA1 SMA1->REGISTER A&B	
				CMP SMA2 COMP REG A/SMA2 RES->COMP	
				STD SMA3 COMP->SMA3 .0= .1< .2> .3#	
				STB SMA3.0 COMP.0 ->SMA3.0 .0=	
AND SMA	16	LOGIC. AND	-	RR AND SMA.->RR {AND A example }	RESULT
ANN SMA	17	LOGIC.NAND	-	RR NAND SMA.->RR {ORI B }	RESULT
ORI SMA	18	LOGIC. OR	-	RR OR SMA.->RR {AND C }	RESULT
ORN SMA	19	LOGIC.NOR	-	RR NOR SMA.->RR {ORI D RR=A+(BC)+D }	RESULT
ADD SMA	20	ADD	1	(4)SMA->B(A+B =A->SMA) 0.0=B=0	CALCULATION
SUB SMA	21	SUBTRACT	1	(4)SMA->B(A-B =A->SMA) 0.0=B=0	CALCULATION
				FHD SMA1 SMA1->REGISTER A&B	
				ADD SMA2 SMA2->REGISTER B (SUB)	
				STD SMA3 REG.A=REG.A+REG.B -> SMA3	
				(ADD)ALS REG.A>9999 OVF-> SMA 0.0 = 1	
				(SUB)ALS REG.A<0 OVF-> SMA 0.0 = 1	
MUL SMA	22	MULTIPLY.	1	(4)SMA->M&Q(M&Q*B=A->SMA) 0.0=B=0	CALCULATION
				FHD SMA1 SMA1->REGISTER A&B	
				MUL SMA2 SMA2->REG.M/Q REG.A & OVF=0	
				STD SMA3 REG.A=REG.B*REG.M/Q ->SMA3	
				ALS REG.A>9999 OVF-> SMA 0.0 = 1	
				EERSTE STD(STB) START BEREKENING	
DIV SMA	23	DIVIDE	1	(4)SMA->B(A/B=M&Q->SMA REST=A 0.0=0	CALCULATION
				FHD SMA1 SMA1->REGISTER A&B	
				DIV SMA2 SMA2->REG.B OVF=0	
				STD SMA3 REG.M/Q=REG.A/REG.B ->SMA3	
				ALS REG.B=0 OVF(SMA 0.0) = 1	
				REST VALUE REMAINS IN REG.A	
JSF PMA	24	FALSE->SUBR	0	PMA = SUBR.(PMA) PMA+1 IN STACK	FUNCTIONS
JST PMA	25	TRUE->SUBR.	1	PMA = SUBR.(PMA) PMA+1 IN STACK	FUNCTIONS
RET	26	END SUB	-	PMA = STACK PMA RR = RR VAN PMA-1	FUNCTIONS
LIO SMA	27	LAST IO	-	SMA = LAST I/O ADRES(END =1ST I/O)	I/O SELECT
	28				
JBF N	29	JUMP BACK	0	PMA=PMA-N PMA+1 IN STACK	JUMP RELATIV
JFF N	30	JUMP FORW.	0	PMA=PMA+N PMA+1 IN STACK	JUMP RELATIV
END SMA	31	GA REGEL 0	-	SMA = FIRST I/O ADRES UPDATE PMA = 0	I/O SELECT

7.4Apendix D Wiring diagram Programming cable PC20

WIRING PC RS232 TO CI21.

FEMALE (DTE) 9-POLIG PC	Wiring		MALE (DTE) 25-POLIG CI
1 RLSD	in	out	8 RLSD
2 RXD?	in	out	3 RXD
3 TXD?	out	in	2 TXD
4 DTR	out	in	20 DTR
5 GND	---	---	7 GND
6 DSR	in	out	6 DSR
7 RTS	out	in	4 RTS
8 CTS	in	out	5 CTS
9 RI			22 RI