CAMAC Interface PC16 / PC16-Turbo

••••••••••••••••••••••••••••••••••••

User Manual

CONTENT

17 January 1996

1. IBM PC/AT interface Card PC16

1.1. General description

The PC16 interface card connects the CC16 CAMAC crate controller with a standard personal computer AT or XT with 80X86 CPU. It is a 16-bit ISA-bus card of full length (AT-board). The PC16 can be used both in byte or word access mode to support XT or AT computers. The required mode is defined by jumper setting (jumpers 1-16). In case of XT - byte access mode two cycles for in- and for output are necessary.

For correct operation the interface cable has to be terminated inside the PC16 in the same way as done in the last CC16 of the chain (see 1.3). Two types of cable termination are provided:

type	resistor	cable length
short distance	DIL 220 Ohm	<30m
long distance*	2 x SIL 8+1 x 100 Ohm *	>30m

Table 1 Termination resistor types (* factory prepared)

The CC16 allows an interrupt based LAM servicing. Both IRQ-number as well as I/O address for data transfer can be chosen by DIP-switches or jumper settings (see Fig. 2),

IRQ	3	5	9	11	12
jumper	1	2	3	4	5

Table 2, IRQ setting - J19 (labelling of jumpers starts at the left side at RX3).

The enabling / disabling of interrupt based LAM servicing is done on the CC16 and has to be performed by software. Further the ALE switch has to be in ON position.

Up to 15 CC16 can be linked in parallel to one CC16 interface card. For control 15 geographical addresses (0-14) and one broadcast address (15) for parallel access is used.

1.2. PC16 Operation and I/O Address Range

The data in- and output is arranged via a 6 byte I/O-port range starting at the basic address **BADR**. The BADR value is defined by the binary decoded DIP-switch SW1 in the range between (hex) $0 and $3F8.

The first bits of BADR are 0. Switch 8 (ALE) has to be in position ON for CAMAC-operation,

Switch	1	2	3	4	5	6	7	8
Bit	A3	A4	A5	A6	A7	A8	A9	ALE
Function	bits of basic I/O address							IRQ

Table 3, BADR setting

The standard factory prepared address range starts at BADR=$380.

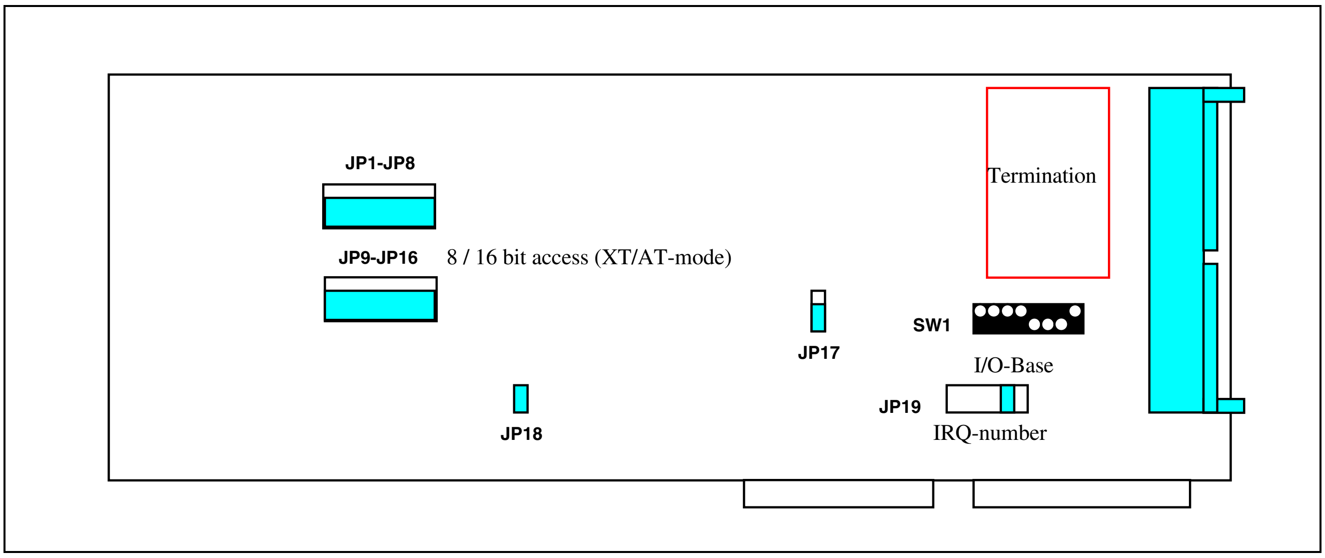

Fig. 1, PC 16 jumper and switch locations

All I/O operations for CC16 access are done within a 3 word range (3 x 16 bit, or 6 x 8 bit) starting at the basic I/O address defined by DIP switches setting as described above. The relevant operations are defined in the following table:

Address	Function	Used Bits	Operation
BADR	OSR - output status register	CSR0, CSR1	write
BADR	ISR - input status register	CSD0-CSD2	read
BADR+2	ODW - output data word	D0-D15	write
BADR+4	IDW - input data word	W0-W15	read

Table 4, I/O range of PC16

With the OSR output one of 4 four primary modes of the CC16 is defined by CSR0 and CSR1 (see 1.5).

Reading the status register (**ISR**) one can get the Q and X response of the previous CAMAC operation. Further any pending LAM is given if the CC16 is enabled for LAM servicing. The following table gives the ISR bit assignment

Bit	Name	Function
0	CSD0	Q-response
1	CSD1	X-response
2	CSD2	LAM

Table 5, Bit assignment **ISR**

1.3. High Speed Interface PC 16 - Turbo

The PC16 - TURBO was designed to accelerate the CAMAC control via IBM-PC. This new interface card for the CC16 fits into a standard AT-long slot (word access only).
The PC16 Turbo card contains a high integrated PLD circuit which allows to reduce the CAMAC controller access time by using a wider I/O port range. Thus this new interface has a read / write rate between computer and CAMAC controller of up to 6µs/CNAF or up to nearly 1.5µs in block mode (measured with 486DX2-66).
In case of cable length up to 15 metres one may use the internal 24MHz oscillator, in case of longer distances and / or a noisy environment it may be necessary to run the interface card with the standard 8MHz AT ISA bus clock (jumper selectable, see figure 2).

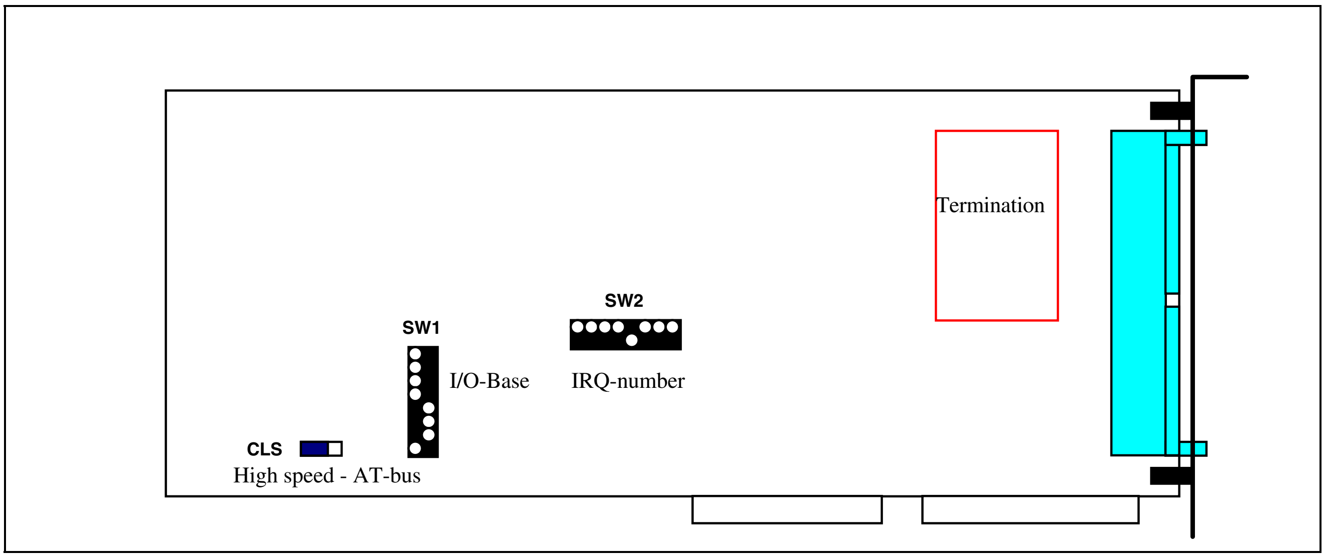

Fig. 2, PC16 - Turbo jumper and switch locations

1.4. PC16 - Turbo Operation and Address Range

The PC16 - Turbo data in- and output is arranged via an enhanced I/O-port range of 26 bytes starting at the basic address **BADR**. The BADR value is defined by the binary decoded DIP-switch SW1 in the range between (hex) $000 and $3E0.

Switch	1	2	3	4	5	6	7	8
Bit	-	-	A6	A5	A8	A7	A9	-
Function			Base Address setting					

Table 6, PC16 - Turbo BADR switch setting

The standard factory prepared address range starts at BADR=$380.

All I/O operations for CC16 access are done within a 26 byte (13 word) range starting at the basic I/O address defined by DIP switches setting as described above. The relevant registers and operations are defined within the following table:

Address	Name	Function	Operation
BADR	GEO_port	crate address and cable delay	write
BADR+$02	FNA_port	function, subaddress, station number	write
BADR+$04	HB_W	write high byte	write
BADR+$06	LW_W	write low word	write
BADR+$08	HB_R	read high byte	read
BADR+$0A	LW_R	read low word	read
BADR+$0C	BR_R	block read with NAF repeat	read
BADR+$0E	S_mode_W	set CAMAC I, C, Z, LAM-enable	write
BADR+$10	S_mode_R	read CAMAC I, LAM-enable, on/off	read
BADR+$12	R_S	read X, Q, LAM	read
BADR+$14	Crate_C	release CAMAC C cycle	write
BADR+$16	Crate_Z	release CAMAC Z cycle	write
BADR+$18	Crate_on	set broadcast bus call on	write
BADR+$1A	Crate_off	set broadcast bus call off	write

Table 7, PC16-Turbo I/O address layout (address offset in hex)

Note, that all the primary and submode operations described within the CC16 manual as well as in the PC16 chapter are done automatically by the high integrated PLD of the PC16-Turbo interface.

- **set crate address and delay, GEO_port Bit assignment (Address: BADR):**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A0	A2	A4	A8	-	-	-	-	D0	D2	D4	D8	-	-	-	-
Crate number								Delay							

Table 8, GEO_port bit assignment

The address of the crate which has to be accessed corresponds to the value selected on the CC16 front panel. For broadcast calls the address 15(hexF) has to be used.

The delay value can be neglected (set to 0) for cable lengths of up to 10 ... 30 metres. In case of longer distances each bit corresponds to an additional delay of about 500ns.

- **set NFA, NFA_port Bit assignment (Address: BADR+$02):**

The NAF-code includes the station number N of the addressed module, the sub-address A and the function number F. It is coded according to the following bit assignment:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
N1	N2	N4	N8	N16	F1	F2	F4	F8	F16	A1	A2	A4	A8	--	S
station N					function F					sub-address A				--	S

Table 9, NAF_port coding

The last bit S corresponds to an immediate CAMAC cycle on a NAF-load. If this bit is 0 the NAF load in the CC16 will not cause a prompt CAMAC cycle. S=0 is necessary write data to the dataway. For read-operations one has to set S=1.

- **set special modes, S_mode_W Bit assignment (Address: BADR+$0E):**

Value	Reference
10 ($A)	set CAMAC I (Inhibit line)
11 ($B)	no CAMAC I (Inhibit line)
12 ($C)	enable LAM detect
13 ($D)	disable LAM detect

Table 10, S_mode_W bit mapping

- **Read special status, S_mode_R Bit assignment (Address: BADR+$10):**

Bit	Reference
0	read CAMAC I (Inhibit line)
1	read LAM enable/disable
2	bus on / off (broadcast calls)

Table 11, S_mode_R bit mapping

- **Read Controller status, R_S Bit assignment (Address: BADR+$12):**

Bit	Reference
0	read Q-Response
1	read X-Response
2	-
3	LAM pending and enabled

Table 12, R_S bit mapping

The controller status register R_S can be used for getting as well as the Q and X response of the CAMAC calls as well as a pending LAM. The LAM information is only available if the CC16 crate controller is enabled for LAM servicing (test by reading S_mode_R register bit 1). Normally the CAMAC module which has to generate the LAM has to be enabled too by a N-F(26)-A(0) command.

1.5. Using Interrupts

The LAM request from any CAMAC station can be used for interrupting the computer to execute a service routine immediately and return then to what is was doing. However, MS-DOS is not a real-time system and has only rudimentary features for interrupt handling.

The PC16 card has the feature to connect the LAM line with an interrupt line of the IBM-PC/AT (IRQ: 3, 5, 9, 11, 12) selecteable by jumper or DIP-switches (see figures 1 and 2). In case of CC16 - Turbo DIP-switch setting only **one** switch has to be in ON position!

For interrupt based LAM servicing the LAM detect / IRQ feature has to be enabled within the CC16. Further in case of the PC16 standard card the ALE bit of the DIP switch has to be in position on.

From the software standpoint the interrupt handling is highly language dependent. Thus consult the language description for correct programming.

2. Installation

2.1. Interface - installation

Prepare and insert the interface into the computer system. Check the I/O address and IRQ-number for validance and insert the board into a free IBM-PC slot of full size (16-bit slot for 16-bit operations).

By the help of switch 1 the basic I/O address is fixed. The standard value for I/O base address is hex380. Note that installed devices or PC-cards (mouse, network cards, printer, ...) often use this I/O range, too. In this case one has to look for another free range. If the CC16 can not be accessed change the used I/O address on the PC16 card and in the used CAMAC software.

To get a free IRQ inspect the computer with the help of a hardware test code like the Microsoft® MSD program. Be shure that the IRQ number choosen for the PC16 card is not used by any other hardware component (lpt, com, net, ...).

Connect the PC 16 to the CC16 in parallel with the enclosed 50-pin flat cable. In case of longer distances (>30 metres) the use of a twisted pair cable is recommended.

CC16 standard interface only: If only a 8-bit slot or XT-computer is available change the jumper setting for 8/16 bit access (jumper 1-17). The jumper 18 has to be installed for operation with CC16.

2.3. Initialisation

For correct operation the CC16-interface system has to be initialised first by setting the geographical address of the CC16 controller which has to be accessed. Further the initial conditions for inhibit line (I - set off) and Detect/IRQ on LAM should be defined (see next chapters). At the end of initialisation a CAMAC initialise (Z) should be performed (see next chapter).

3. PC16 Programming and CAMAC operation

The CC16 operation is performed on the basis of calling the several primary and sub-modes and writing / reading data via the I/O ports of the interface card / IBM-PC system according to the CP-Bus protocol.

The following instruction schemes for the PC16 standard interface are given in terms of setting these modes and writing / reading data. In general the several calls for I/O operations are depending on the programming language. Therefore in this chapter only the main principles of programming are given. For description of modes and registers see chapter 1.5.

Software examples for several languages (TURBO PASCAL, BASIC) are given in the next chapters .

3.1. Call and set crate on/off

select crate, write geographical address

```
set PM0     - set crate / branch
write GAW   - write crate / branch
```

Set crate on / off

```
set PM1     - set sub-mode
set SM7     - set SM to CC16 status register
set PM2     - set write data
write SCB   - write scb with:     bit2=0 switch crate on
                                  bit2=1 switch crate off
```

to test the setting read the SCB with

```
set PM3     - set read data
read SCB    - read SCB with:      bit2=0 crate is on
                                  bit2=1 crate is off
```

The crate ON / OFF is displayed by a status LED at the front panel of the CC16. These ON / OFF modes are only relevant for broadcast operations.

3.2. General CAMAC commands Z, C, I

CAMAC INITIALISE - Z

```
set PM1     - set sub-mode
set SM9     - set SM to enable Z
set SM6     - set SM to perform Z cycle
```

Q and X response will be set to 1 as response of CC16 to the Z-cycle.

CAMAC CLEAR - C

```
set PM1     - set sub-mode
set SM8     - set SM to enable C
set SM6     - set SM to perform C cycle
```

Q and X response will be set to 1 as response of CC16 to the C-cycle.

SET CAMAC INHIBIT - I

```
set PM1    - set sub-mode
set SM10   - set SM to enable I
```

SET BACK CAMAC INHIBIT - I

```
set PM1    - set sub-mode
set SM11   - set SM to disable I
```

The status of the inhibit line (I) is displayed at the CC16 front panel by the Inhibit LED.

3.3. LAM servicing

ENABLE SERVICING/IRQ ON LAM REQUEST

```
set PM1    - set sub-mode
set SM12   - set SM to enable IRQ on LAM request
```

DISABLE SERVICING/IRQ ON LAM REQUEST

```
set PM1    - set sub-mode
set SM13   - set SM to disable IRQ on LAM request
```

GET PENDING LAM (if IRQ enabled)

```
read ISR   - read ISR pending LAM = bit 2
```

READ STATUS REGISTER SCB

```
set PM1    - set sub-mode
set SM7    -  set SM to CC16 status register
set PM3    - set read data
read SCB   - read SCB (bit mapping see 1.5)
```

CC16 INITIALISATION

The complete CC16 initialisation should include setting of crate / branch and SCB as well as of giving Z-cycle and checking for no setting of I-line,

```
set PM0    - set crate / branch
write GAW  - write crate / branch
set PM1    - set sub-mode
set SM7    - set SM to CC16 status register
set PM2    - set write data
write SCB  - write SCB =0  bit2=0 set crate on
set PM1    - set sub-mode
set SM11   - set SM to disable I
set SM13   - set SM to disable IRQ on LAM request
```

3.4. Read and Write Data

ACCESS AND DATA TRANSFER TO CAMAC MODULES

The data transfer to or from modules of the CC16 controlled crate has to be done in two steps due to the 24-bit word length of the CAMAC bus. First read or write the low and middle byte (in AT-mode as one 16-bit word). The second cycle is for reading / writing the high byte if required. However a lot of CAMAC modules use only word length up to 16 bit. In this case 16-bit data transfer is faster and more useful.

WRITE 24-BIT WORD

```
set PM1     - set sub-mode
set SM0     - set SM to write NAF-word
set PM2     - set write data
write NAF   - NAF with S-bit=0
set PM1     - set sub-mode
set SM1     - set SM to write high byte
set PM2     - set write data
write W17-24     - write high data byte
set PM1     - set sub-mode
set SM2     - set SM to write low word
set PM2     - set write data
write W1-16      - write low data word and start CAMAC cycle
```

WRITE 16-BIT WORD

```
set PM1     - set sub-mode
set SM0     - set SM to write NAF-word
set PM2     - set write data
write NAF   - NAF with S-bit=0
set PM1     - set sub-mode
set SM2     - set SM to write low word
set PM2     - set write data
write W1-16      - write low data word and start CAMAC cycle
```

16-BIT FAST BLOCK TRANSFER (WRITE)

```
set PM1     - set sub-mode
set SM0     - set SM to write NAF-word
set PM2     - set write data
write NAF   - NAF with S-bit=0
set PM1     - set sub-mode
set SM2     - write low word
set PM2     - set write data
write W1-16      - write low data word and start CAMAC cycle
write W1-16      - write low data word and start CAMAC cycle
write W1-16      - write low data word and start CAMAC cycle
write W1-16      - write low data word and start CAMAC cycle
write W1-16      - write low data word and start CAMAC cycle
  ....
```

READ 24-BIT WORD

```
set PM1     - set sub-mode
set SM0     - set SM to write NAF-word
set PM2     - set write data
write NAF   - NAF with S-bit=1 and start CAMAC cycle
set PM1     - set sub-mode
set SM4     - set SM to read high byte
set PM3     - set read data
read W17-24      - read high data byte
set PM1     - set sub-mode
set SM3     - set SM to read low word
set PM3     - set read data
read W1-16 - read low data word and start CAMAC cycle
```

READ 16-BIT WORD

```
set PM1    - set sub-mode
set SM0    - set SM to write NAF-word
set PM2    - set write data
write NAF  - NAF with S-bit=1 and start CAMAC cycle
set PM1    - set sub-mode
set SM3    - set SM to read low word
set PM3    - set read data
read W1-16 - read low data word and start CAMAC cycle
```

16-BIT FAST BLOCK TRANSFER (READ)

```
set PM1    - set sub-mode
set SM0    - set SM to write NAF-word
set PM2    - set write data
write NAF  - NAF with S-bit=1 and start CAMAC cycle
set PM1    - set sub-mode
set SM5    - set SM to read low word
set PM3    - set read data
read W1-16 - read low data word and start CAMAC cycle
read W1-16 - read low data word and start CAMAC cycle
read W1-16 - read low data word and start CAMAC cycle
read W1-16 - read low data word and start CAMAC cycle
  ...
```

the last word has to be read again by the full sequence :

```
set PM1    - set sub-mode
set SM3    - set SM to read low word
set PM3    - set read data
read W1-16 - read low data word and start CAMAC cycle
```

3.5. CAMAC status bits Q and X

The status bits Q and X are stored in the status register of the interface (**ISR**) in the first bits CSD0 and CSD1. The Q- and X- bits will be stable within 1.4µs.

```
read ISR   - read ISR Q = bit 0  X = bit 1
```

4. PC16 - Software Examples

4.1. TURBO PASCAL

Within TURBO PASCAL the I/O operations are performed via ***port*** (for byte in- and output) or ***portw*** (for word in- and output) commands. BADR is the start address of the I/O range (normally hex380).

The following procedure examples describe:
- set crate number
- switch selected crate bus on
- get status (crate on/off)
- CAMAC clear (C)
- set CAMAC inhibit (I)
- get Q- and X-response
- call NAF and write/read data (16bit)

```
PROCEDURE SetCrate(crate : byte);
var i : word;
begin
      i:=1;
      i:= 16 or crate;
      portw[BADR]:=0;
      portw[BADR+2]:=i;
end;

PROCEDURE CratOn(crate : byte);
var i : word;
begin
      i:=1;
      i:=16 or crate;
      portw[BADR]:=0;
      portw[BADR+2]:=i;
      portw[BADR]:=1;
      portw[BADR+2]:=7;
      portw[BADR]:=2;
      portw[BADR+2]:=0;
end;

Function getstatus : word;
var j: byte;
begin
  j:=0;
  portw[BADR]:=1;
  portw[BADR2]:=7;
  portw[BADR]:=3;
  J:=port[BADR4];
  getstatus:=j and 7;
end;

procedure CAM_C;
var i: byte;
begin
  portw[BADR]:=1;
  portw[BADR2]:=$B;
  portw[BADR2]:=8;
  portw[BADR2]:=6;
end;

procedure CAM_I;
begin
```

```
  portw[BADR]:=1;
  portw[BADR2]:=$A;
end;
```

PROCEDURE CAM_QX(var x, q : boolean);

```
var i : word;
begin
      i:=portw[BADR];
      if (i and 1) = 0 then q:=false
      else q:=true;
      if (i and 2) = 0 then x:=false
      else x:=true;
end;
```

PROCEDURE CAM_NAF(n,a,f: byte; var data: word);

```
var i,j : word;
begin
      i:=(a and $0F) shl 5;
      i:=(i or (f and $1F)) shl 5;
      i:=i or (n and $1F);
      PORTW[BADR]:=1;
      PORTW[BADR+2]:=0;
      PORTW[BADR]:=2;
      if f<15 then begin
            PORTW[BADR+2]:=i or $8000;
            PORTW[BADR]:=1;
            PORTW[BADR+2]:=3;
            PORTW[BADR]:=3;
            data:=PORTW[BADR+4];
      end else begin
            PORTW[BADR+2]:=i;
            PORTW[BADR]:=1;
            PORTW[BADR+2]:=1;
            PORTW[BADR]:=2;
            PORTW[BADR+2]:=$0000;
            PORTW[BADR]:=1;
            PORTW[BADR+2]:=2;
            PORTW[BADR]:=2;
            PORTW[BADR+2]:=data;
      end;
end;
```

4.2. C/C++

Within C++ the I/O operations are performed via ***outport*** (for integer output) or ***inport*** (for integer input) commands. BADR is the start address of the I/O range (hex = 0x380).

The following procedure examples describe:

- set base address
- set crate number
- get status (crate on/off)
- CAMAC initialize (Z)
- set CAMAC inhibit (I)
- get Q- and X-response
- call NAF and write/read data (16bit)
- get LAM

void cam_adr(int bad)

```
{
  badr=bad;
  badr2=badr+2;
  badr4=badr+4;
};

void   set_crate (int crate)
{
  outport(badr,0);
  outport(badr2,(crate & 16));
};

int getstatus(void)
{
  unsigned int j;
  j=0;
  outport(badr ,1);
  outport(badr2,7);
  outport(badr ,3);
  j=inport(badr4);
  return(j & 7);
};

void cam_z(void)
{
  outport(badr,1);
  outport(badr2,11);
  outport(badr2,9);
  outport(badr2,6);
};

void cam_i(void)
{
  outport(badr,1);
  outport(badr2,10);
};

int cam_q(void)
{
  char j;
  j=inport(badr);
  return ( j & 1);
};

int cam_x(void)
{
  char j;
  j=inport(badr);
  return ( j & 2) > 1;
};

void cam_nfa_write(int n, int f, int a, int data) //16 bit data
{ int nfa;
  nfa= a << 5;
  nfa= (nfa | f) << 5;
  nfa= nfa | n;
  outport(badr,1);
  outport(badr2,0);
  outport(badr,2);
  outport(badr2,nfa);
  outport(badr,1);
  outport(badr2,2);
  outport(badr,2);
  outport(badr2,data);
};

int cam_nfa_read(int n, int f, int a) //16 bit data
{ int nfa;
  nfa= a << 5;
  nfa= (nfa | f) << 5;
  nfa= nfa | (n+0x8000);
  outport(badr,1);
  outport(badr2,0);
  outport(badr,2);
  outport(badr2,nfa);
  outport(badr,1);
  outport(badr2,3);
  outport(badr,3);
  return (inport(badr4));
};

int cam_lamf(void)
{
  int j;
  j=inport(badr);
  return((j & 4) > 2);
};
```

4.3. BASIC

Within BASIC the I/O operations are perfomed via an ***inp***-command for getting the value of an I/O register and an ***out***-command for setting an I/O register. Due to the large differences between the BASIC dialects this examples demonstrate only the general principle.
BADR is the start address of the I/O range (normally hex380).

The following examples describe:

- set crate number
- switch selected crate bus on
- get status (crate on/off)
- CAMAC clear (C)
- set CAMAC inhibit (I)
- get Q- and X-response
- call NAF and write/read data (16bit)

SetCrate crate% = crate number

```
10   l=1
11   l= 16 or crate%
12   OUT BADR%, 0
13   OUT BADR%+2, l
```

CratOn crate% = crate number

```
20   l=1
21   l=16 or crate%
22   OUT BADR%, 0
23   OUT BADR%+2, l
24   OUT BADR%, 1
25   OUT BADR%+2, 7
```

```
26    OUT BADR%, 2
27    OUT BADR%+2, 0
```

getstatus

```
31    j=0
32    OUT BADR%, 1
33    OUT BADR%+2, 7
34    OUT BADR%, 3
35    J=INP(BADR%)
36    getstatus=j and 7
```

CAM_C

```
41    OUT BADR%, 1
42    OUT BADR%2, $B
43    OUT BADR%2, 8
44    OUT BADR%2, 6
```

CAM_I

```
51    OUT BADR%, 1
52    OUT BADR%2, $A
```

CAM_NAF   n%,a%,f%; data%

```
61    l%=(a% and $0F  )*2^5
62    l%=(l% or (f% and $1F))*2^5
63    l%=l% or (n% and $1F)
64    OUT BADR%, 1
65    OUT BADR%+2, 0
66    OUT BADR%, 2
67    if f%<15 then goto 73
68         OUT BADR%+2, l% or $8000
69         OUT BADR%, 1
70         OUT BADR%+2, 3
71         OUT BADR%, 3
72         data=OUT BADR%+4
73    if f%>15 then goto 84
75         OUT BADR%+2, l%
76         OUT BADR%, 1
77         OUT BADR%+2, 1
78         OUT BADR%, 2
79         OUT BADR%+2, $0000
80         OUT BADR%, 1
81         OUT BADR%+2, 2
82         OUT BADR%, 2
83         OUT BADR%+2, data
84    return
```

5. PC16 - TURBO Programming and CAMAC operation

Compared with the PC16 standard interface the CC16 TURBO operation and programming has to be done in a more simple and compressed form without considering the several primary modes and and submodes of the CP-bus protocol.

The following instruction schemes describes programming language independant the CAMAC access types.

5.1. Call and set crate on/off

select crate, write geographical address

```
write crate address (+ opt. delay) to GEO_port (BADR+$0)
```

Set crate on / off

```
write dummy data to Crate_on (BADR+$18)
```

to test the setting:

```
read S_mode_R (BADR+$10)        bit 3: on=0, off=1
```

The crate ON / OFF is displayed by a status LED at the front panel of the CC16. These ON / OFF modes are only relevant for broadcast operations.

5.2. General CAMAC commands Z, C, I

CAMAC INITIALISE - Z

```
write dummy data to Crate_Z (BADR+$16)
```

Q and X response will be set to 1 as response of CC16 to the Z-cycle.

CAMAC CLEAR - C

```
write dummy data to Crate_C (BADR+$14)
```

Q and X response will be set to 1 as response of CC16 to the C-cycle.

SET CAMAC INHIBIT - I

```
write $A to S_mod_W (BADR+$0E)
```

SET BACK CAMAC INHIBIT - I

```
write $B to S_mod_W (BADR+$0E)
```

The status of the inhibit line (I) is displayed at the CC16 front panel by the Inhibit LED.

5.3. LAM servicing

ENABLE SERVICING/IRQ ON LAM REQUEST

```
write $C to S_mod_W (BADR+$0E)
```

DISABLE SERVICING/IRQ ON LAM REQUEST

```
      write $D to S_mod_W(BADR+$0E)
```

GET PENDING LAM (if IRQ enabled)

```
      read R_S(BADR+$12)     - read ISR pending LAM = bit 2
```

READ STATUS REGISTER SCB

```
      read S_mod_R(BADR+$10)
```

CC16 INITIALISATION

The complete CC16 initialisation should include setting of crate / branch and SCB as well as of giving Z-cycle and checking for no setting of I-line,

```
      write crate address (+ opt. delay) to GEO_port (BADR+$0)
      write dummy data to Crate_Z (BADR+$16)
      write $B to S_mod_W (BADR+$0E)
      write $C to S_mod_W (BADR+$0E)
```

5.4. Read and Write Data

ACCESS AND DATA TRANSFER TO CAMAC MODULES

The data transfer to or from modules of the CC16 controlled crate has to be done in two steps due to the 24-bit word length of the CAMAC bus. First read or write the low 16-bit word. The second cycle is for reading / writing the high byte if required. However a lot of CAMAC modules use only word length up to 16 bit. In this case 16-bit data transfer is faster and more useful.

WRITE 16-BIT or 24-BIT WORD

```
      write NAF to FNA_port (BADR+$02) (FNA with S=0!)
(     write high byte to HB_W (BADR+$04)     )
      write low word to LW_W (BADR+$06)
```

The S-bit of the NAF-word has to be 0 to avoid the CAMAC cycle ddirectly after setting NAF. The CAMAC cycle will be realized in this case after writing the low word.

READ 16-BIT or 24-BIT WORD

```
      write NAF to FNA_port (BADR+$02) (FNA with S=1!)
(     read high byte from HB_R (BADR+$08)    )
      read low word from LW_R (BADR+$0A)
```

To realise a prompt CAMAC cycle after NAF setting the S-bit has to be 1.

16-BIT FAST BLOCK TRANSFER (READ)

```
      write NAF to FNA_port (BADR+$02) (FNA with S=0!)
      read low word from BR_R (BADR+$0C)
      read low word from BR_R (BADR+$0C)
        ...
```

the last word has to be read again by the full sequence :

```
      read low word from LW_R (BADR+$0A)
```

5.5. CAMAC status bits Q and X

The status bits Q and X are stored in the status register of the interface PC16-TURBO in the first bits. The Q- and X- bits will be stable within 1.4µs.

```
      read status from R_S (BADR+$12)
```

6. PC16 - TURBO Software Examples

6.1. TURBO PASCAL

Within TURBO PASCAL the I/O operations are performed via ***port*** (for byte in- and output) or ***portw*** (for word in- and output) commands. BADR is the start address of the I/O range (normally hex380).

The following procedure examples describe:

- set crate number
- get status (crate on/off)
- CAMAC clear (C)
- set CAMAC inhibit (I)
- get Q- response
- call NAF and write/read data (16bit)

```
Procedure Set_Crate(CNr : word);
begin
  portw[BAdr00]:=(16 or cnr);
end;

function getstatus : word;
var i: byte;
    j : integer;
begin
  j:=0;
  J:=portw[badr10];
  getstatus:=j and 7;
end;

procedure CAM_C;
begin
  portw[badr14]:=0;
end;

procedure CAM_I;
begin
  portw[badr0E]:=10;
end;

function CAM_Q: Boolean;
var i:byte;
begin
  i:=port[badr12] and 1;
  if i=1 then CAM_Q:=true else CAM_Q:=false;
end;

procedure CAM_NFA_Write(N,F,A,Data : integer);
var naf : integer;
begin
```

```
  naf:=(A and $0F) shl 5;
  naf:=(naf or (F and $1F)) shl 5;
  naf:=naf or (N and $1F);
  portw[badr02]:=naf;
  portw[badr06]:=data;
end;

function CAM_NFA_Read24(N,F,A : integer) : longint;
var li : longint;
    naf : integer;
begin
  naf:=(A and $0F) shl 5;
  naf:=(naf or (F and $1F)) shl 5;
  naf:=naf or (N and $1F);
  portw[badr02]:=naf or $8000;
  li:=portw[badr08];
  li:=(li and $ff) shl 16;
  CAM_NFA_Read24:=li + portw[badr0A];
end;
```

6.2. C / C++

Within C++ the I/O operations are performed via ***outport*** (for integer output) or ***inport*** (for integer input) commands. BADR is the start address of the I/O range (hex = 0x380).

The following procedure examples describe:

- set crate number
- get status (crate on/off)
- CAMAC clear (C)
- set CAMAC inhibit (I)
- get Q- response
- call NAF and write/read data (16bit)

```
void   set_crate (int crate)
{
  outport(badr00,(crate & 16));
};

int getstatus(void)
{
  char i;
  char j;
  j=0;
  j=inport(badr10);
  return(j & 7);
};

void cam_c(void)
{
  outport(badr14,0);
};

void cam_i(void)
{
  outport(badr0e,10);
};
```

```
int cam_q(void)
{
  char j;
  j=inport(badr12);
  return ( j & 1);
};

void cam_nfaqx_write(int n, int f, int a, int data,int *q, int *x)
{ int nfa;
  char j;
  nfa= a << 5;
  nfa= (nfa | f) << 5;
  nfa= nfa | n;
  outport(badr02,nfa);
  outport(badr06,data);
  j=inport(badr12);
  *q=( j & 1);
  *x=( j & 2) > 1;
};

long int cam_nfaqx_read24(int n, int f, int a, int *q, int *x)
{ int nfa;
  char j;
  long int lh;
  nfa= a << 5;
  nfa= (nfa | f) << 5;
  nfa= nfa | (n+0x8000);
  outport(badr02,nfa);
  lh=inport(badr08) & 255;
  lh=lh <<16;
  lh += (unsigned int) inport(badr0a);
  j=inport(badr12);
  *q=( j & 1);
  *x=( j & 2) > 1;
  return (lh);
};
```

A P P E N D I X

1. Technical Documentation PC16

1. Technical Documentation PC16 - TURBO

TURBO PASCAL CAMAC - Routines for CC16 (TP 5.0 - 7.0)

Unit CAM_CC16.PAS comprises :

CAM_ADR	- Set I/O Address (starting address=BADR)
Getstatus	- get CC16 Status
CAM_CrateCheck	- check crate available
Set_Crate	- set geographical crate Address
CAM_Controller_Ini	- CC16 Initialization
CAM_0	- CAMAC Command register=0 (no C, Z, I)
CAM_Z	- CAMAC Initialize (Z)
CAM_C	- CAMAC Clear (C)
CAM_I	- CAMAC Inhibit (I)
CAM_CI	- set CAMAC Clear + Inhibit (I)
CAM_IRQ	- CAMAC LAM-Interrupt enabled
CAM_NoIRQ	- CAMAC LAM-Interrupt disabled
CAM_Q	- get CAMAC Q-response
CAM_X	- get CAMAC X-Response
CAM_NFA	- set N,F,A and start CAMAC cycle
CAM_NFA_Read	- set N,F,A and read data (2byte)
CAM_NFA_Write	- set N,F,A and write data (2byte)
CAM_NFAQX_Write	- set N,F,A write data (2byte) and get Q and X
CAM_NFAQX_Read	- set N,F,A read data (2byte) and get Q and X
CAM_NFAQX_Write24	- set N,F,A write data (3byte) and get Q and X
CAM_NFAQX_Read24	- set N,F,A read data (3byte) and get Q and X
CAM_LAM	- LAM request from station
CAM_LAMF	- any LAM request
CAM_NFA_Blockread	- set NFA and fast blockread

BADR := basic I/O address has to be defined in CAM_Controller_Ini (standard $380)

BORLAND C++ CAMAC - Routines for CC16

Unit cam_cc16.h comprises :

cam_adr	- Set I/O Address (starting address=BADR)
getstatus	- get CC16 Status
cam_cratecheck	- check crate available
set_crate	- set geographical crate Address
cam_controller_ini	- CC16 Initialization
cam_0	- camAC Command register=0 (no C, Z, I)
cam_z	- camAC Initialize (Z)
cam_c	- camAC Clear (C)
cam_i	- camAC Inhibit (I)
cam_ci	- set camAC Clear + Inhibit (I)
cam_irq	- camAC LAM-Interrupt enabled
cam_noirq	- camAC LAM-Interrupt disabled
cam_q	- get camAC Q-response
cam_x	- get camAC X-Response
cam_nfa	- set N,F,A and start camAC cycle
cam_nfa_read	- set N,F,A and read data (2byte)
cam_nfa_write	- set N,F,A and write data (2byte)
cam_nfaqx_write	- set N,F,A write data (2byte) and get Q and X
cam_nfaqx_read	- set N,F,A read data (2byte) and get Q and X
cam_nfaqx_write24	- set N,F,A write data (3byte) and get Q and X
cam_nfaqx_read24	- set N,F,A read data (3byte) and get Q and X
cam_lam	- LAM request from station
cam_lamf	- any LAM request

badr := basic I/O address has to be defined in cam_controller_ini (standard 0x380)

W-IE-NE-R, Plein & Baus GmbH
PC16 - TURBO (1)

W-IE-NE-R, Plein & Baus GmbH

PC16 - TURBO (1)

W-IE-NE-R, Plein & Baus GmbH

PC16 - TURBO (2)

W-IE-NE-R, Plein & Baus GmbH

PC16 - TURBO (3)

W-IE-NE-R, Plein & Baus GmbH

PC16 - TURBO (4)

W-IE-NE-R, Plein & Baus GmbH

PC16 - TURBO (5)

Changes

26.09.2001 Page 6: Bit -> Value (Kö)