ISTITUTO NAZIONALE DI FISICA NUCLEARE
Laboratori Nazionali di Frascati

LNF-83/67(R)
19 Ottobre 1983

O. Ciaffoni, M. Coli, M. L. Ferrer and L. Trasatti:

CANDI 2 - DESCRIPTION AND USER MANUAL

Servizio Documentazione
dei Laboratori Nazionali di Frascati
Cas. Postale 13 - Frascati (Roma)

INFN - Laboratori Nazionali di Frascati
Servizio Documentazione

LNF-83/67(R)
19 Ottobre 1983

CANDI 2 - DESCRIPTION and USER MANUAL

O. Ciaffoni, M. L. Ferrer, L. Trasatti
INFN, Laboratori Nazionali di Frascati

and

M. Coli
ENEA, Centro di Frascati

1.0 Introduction

CANDI2 (CAMAC Acquisition Node for Distributed Intelligence) is a powerful microcomputer system with specialized peripheral interfaces, designed in a physics laboratory to give rapid access to the most used kinds of peripherals.

The system has been designed at the LNF with finances from INFN and with the support of the Research Division and of the Computer Center.

The system is now being assembled and distributed by a commercial firm(1), which also offers maintenance.

The facilities implemented in CANDI2 are:

1. CAMAC interface: CAMAC is an accepted and widespread data acquisition system. Therefore CANDI implements the functions of a Crate Controller, and the whole computer is built on CAMAC boards. CANDI 2 is a 4-slot CAMAC module, which occupies the rightmost positions in the crate, substituting the normal controller.

2. PDP-11 and VAX interface: The use of computers of the Digital family as host computers has been decided considering their large and increasing number. Memory image interchange, ASCII file transfer and Transparent terminal utilities are available for the CANDI system.

3. Color graphics: the use of high resolution graphics for data display, event presentation and in general for a better man-machine interface is rapidly developing. A high resolution color display unit is built into CANDI2.

The main characteristics of the CANDI2 system are:

1. CPU: TMS 9995, 16 bit word, 3 MHZ clock.

(1) SEA s.r.l., via Tiburtina, 1020, 00156, ROMA.

2. RAM: 128 K resident on board; can be expanded off board up to 16 MB using a memory mapping technique implemented on the CPU board.

3. EPROM: 32 K maximum on board containing system firmware. Can be expanded off board.

4. I/O ports: 2 RS-232C or 20 mA current loop serial ports. Two parallel PIO's on front panel.

5. Printer port: serial, RS-232C.

6. CAMAC interface: replaces the normal Crate Controller.

7. Graphics interface: 8 colors, 2 memory pages, 512 x 512 pixels, TTL-RGB output with separate syncronism.

8. Capable of color hard-copy.

9. Optional alphanumeric keyboard with alphanumeric display superimposed on the graphic screen.

10. VAX - PDP11 Interface: RS232C or 20 mA current loop, up to 9600 baud.

11. BASIC Interpreter in firmware, 16 K.

12. Standard NIM-ESONE CAMAC subroutines implemented in firmware.

13. High level graphic subroutines, PLOT-10 compatible.

14. Tektronik 4006 emulation.

15. Several software utilities in firmware (assembler, disassembler, memory inspect/change/dump, memory mapping, etc).

2.0 Hardware configuration

The system consists of 3 CAMAC boards: CPU, CAMAC Controller and Graphics interface, plus the optional alphanumeric terminal interface.

2.1 CPU system board

The CPU board of CANDI2 is a complex master processing unit incorporating various I/O communication ports. The system can be devided in the following blocks:

1. CPU:Texas Instruments TMS 9995, 16 bit processor with related logic circuitry to generate and handle the system control signals (memory driving, interrupt handling, I/O signals including Reset, etc.).

2. EPROM memory: 32Kbytes containing the system firmware, on four 8Kbytes EPROMs with single +5V supply.

3. DYNAMIC RAM memory and driving circuitry, consisting of 128Kbytes (16 dynamic 64KByte RAM chips, featuring auto-refresh capability). The memory is driven multiplexing the 16 bits of the address-bus. The timing signals toghether with -CAS,-RAS are generated with appropriate delays, by a chain of Schmidt-triggers and flip-flops.

   Pulses for auto-refreshing are stolen during non memory addressing intervals.

   Access to the whole 160Kbytes is obtained by changing blocks or pages through appropriate settings of the MEMORY MAPPER registers.

4. The MEMORY MAPPER and its service latches are accessed directly as a memory position; the word transferred (16 bits)contains all of the necessary information for reading or writing the mapper registers.

   The MAPPER, contains 16 registers, 12 bits wide, containing the extended address(8+4 bits)selected by the most significant nibble of the CPU address-bus. The l.s. nibble of the 12 bits address contained in the mapper registers replaces the CPU m.s. nibble at the mapper output address-bus.

   The m.s.nibble of the CPU address-bus becomes the address of the mapper-register currently in use.

The resulting address for the system becomes:

1. 12 bits coming directly from the 12 l.s. bits of the CPU address-bus (decoding within 4Kbyte blocks in the memory)

2. 12 bits coming from the registers of the mapper that can be changed via software, giving the possibility of accessing a memory space of 16Mbytes(4K times 4kbytes memory blocks)

Summarizing, the current memory space accessed by the CPU address-bus(64Kbytes seen as 16 blocks of 4Kbytes each, or "pages") is defined within the physical memory space, that can extend up to 16Mbytes, by the contents of the 12 bits mapper registers.

Loading via software the map registers and subsequently changing the MAPPER control register bit 0 to 1 to enter memory mapping mode, the physical memory space(whose extension remains 16 pages of 4Kbytes)is changed to the one pointed to by the mapper registers.

5. The I/O communication system consists of two SERIAL ports and one PARALLEL (22 I/O bits+16 prioritized interrupt inputs.

1. SERIAL port B is an EIA Standard RS 232/C or TTY current-loop port defined as a DCE communication unit to be connected to the system I/O terminal . This can be any standard terminal or the I/O from the optional internal alphanumeric display board. In parallel to serial port B an output is provided on the front panel (PRINTER PORT) for a printer supporting EIA RS 232/C Standard signals. This port can be opened or closed via software writing a 1 (or 0) to the l.s. bit of memory location **7FC7H** (see also system routine "OUT").

2. Serial port A is again a standard RS232/C port defined as a DTE communication unit . It differs from Port B also in that the current-loop signals are optically coupled. Various functions on the I/O serial ports can be altered changing the position of appropriate jumpers. This port has been implemented as I/O port for the HOST COMPUTER (PDP 11, VAX, IBM or HP).

3. The PARALLEL PORT has been implemented with a TMS 9901 parallel port of the TMS 9900 family. It can be read/written to in CRU space and can provide:

1. 16 bits of I/O addressed one or more bits at a time.

2. 6 more bits on separate pins which can be used only as inputs.

3. Some of the same pins which are used for I/O can also be used to input 16 interrupt lines. These interrupts can be individually enabled by software, and they are automatically prioritized and encoded on four output wires.

Unlike the TMS 9900 the TMS 9995 CPU has only two input pins for interrupts, (-INT1,-INT4/EC). A special latch has been implemented to allow reading the interrupt code generated by the TMS 9901 on the m.s. nibble of memory location 7FC6H. The -INTREQ of the parallel port has been connected to the -INT4/EC of the CPU . This allows to mantain the same interrupt structure of CANDI 1, because the BASIC interrupt service routine has been modified to automatically read the interrupt code register.

2.2 CAMAC interface

The CAMAC interface is built on a board which sits in slot 25 of the CAMAC crate. The interface to slot 24 is supplied by the CPU board through flat cables. The system uses 5 PIO's (TMS 9901 Parallel Input Output Interfaces) to interface to the CAMAC highway, with the possibility to access the full 24 bit R/W bus.

The system only controls one crate, since we felt that intercrate communication and multicrate data acquisition would be much better performed by various CANDI units loosely interconnected at a higher level then the CAMAC Branch Highway. An HDLC link is being developed to communicate between CANDI units.

The fact that CANDI2 does not require a standard Crate Controller makes the system more compact and efficient and much less expensive.

The CAMAC Interface board also contains two PIO's and the associated logic to implement the parallel connection to a host computer or to a floppy disk system (under development).

2.3 Graphic system

Graphic display units are growing very fast in popularity. The immediate reason is the rapidly decreasing cost of hardware in general (and of memories in particular), while the real need they serve is to answer the classic question: "Whoever will read all of those numbers?", as referred to multi-kilogram computer printouts.

CANDI Graphics is a high resolution (512 x 512 pixel), 8 color, 2 page raster scan interlaced display unit . The system uses a high resolution raster-scan video terminal and supports a hard-copy unit.

Pixel-by-pixel reads and writes are possible, both for the hard copy unit and for cursor display. Furthermore, the unit is capable of understanding high level commands, that is, point draw/erase, vector draw/erase and alphanumeric character display in various sizes and orientations (normal or italic characters and horizontal or vertical axis). This last capability is particularly suited for hystogram axis identification and the like.

Two separate pages of memory are available, allowing independent preparation of two separate displays. Alternate presentation is also possible, with some amount of "flickering"; this could still be useful, for example for comparison of two different curves.

We have connected the unit to an optional alphanumeric display processor and to a keyboard. This allows presentation of both graphic displays (for example drawings of apparatuses, event display, plus miscellaneous hystograms and curves relating to experimental results) and alphanumeric information (program listings, etc.). It is important that the two things are handled by separate processors, to allow scrolling, erasing and editing of alphanumeric data without disturbing the somewhat more complicated graphics underlying it. The alpha display color is automatically adjusted for maximum readability over differently colored areas of the graphic display.

The unit may be divided into four logical sections:

1. Graphics Display Processor (Thomson-EFCIS) providing raster scan control, vector tracing, etc.

2. Graphic memory (24 x 64 Kbit Dinamic Ram's) for a

total of 192 Kbytes. This constitutes the necessary memory for two independent pages of three bit (8 color) times 512 by 512 pixels. This memory is completely independent of the CPU memory.

3. CPU interface, adapting the Thomson GDP processor to the Texas TMS 9995 CPU bus. The CPU sees the GDP processor as a set of 32 dedicated memory locations for maximum flexibility and speed of operation. Note that, although the video memory is handled directly by the GDP, and thus does not occupy CPU addressing space, the CPU can read or write the video memory through the dedicated register area.

4. Alphanumeric display board (optional). This board produces the normal terminal like alphanumeric display, which is mixed with the graphic display only at the level of TV interface. The unit has its own 4 Kbyte memory.

5. Hard copy unit. A software interface has been developed to implement a hard copy facility on a color dot printer (Integral Data Systems "PRISM" printer). The full resolution of the graphic display can therefore be transferred on paper at a very reasonable cost.

2.4 Memory configuration

The CANDIZ memory configuration is shown in fig. 1. Two and one half 64K pages are implemented on the CPU board. The Power On configuration is shown as PAGE 0 (32K EPROM + 32K RAM). The memory I/O space configuration is shown in Fig. 2. The CRU I/O space configuration is shown in Fig.3.

Note that, since the Memory Mapper is accessed via the memory I/O space, it is essential that this space be accessible even if the computer is not configured in Page 0, otherwise a "no return" situation would occurr. On the other hand, for some applications it could be useful to have an uninterrupted memory configuration (e.g. 64K contiguous RAM). This is possible in the CANDIZ system using a dedicated CRU location. Executing a CALL"WNDI" will make the I/O window disappear and be replaced by continuous memory, while executing a CALL"WNEN" will make it reappear always at locations 7F00H to 7FFFH, independently of the Memory Mapper configuration.

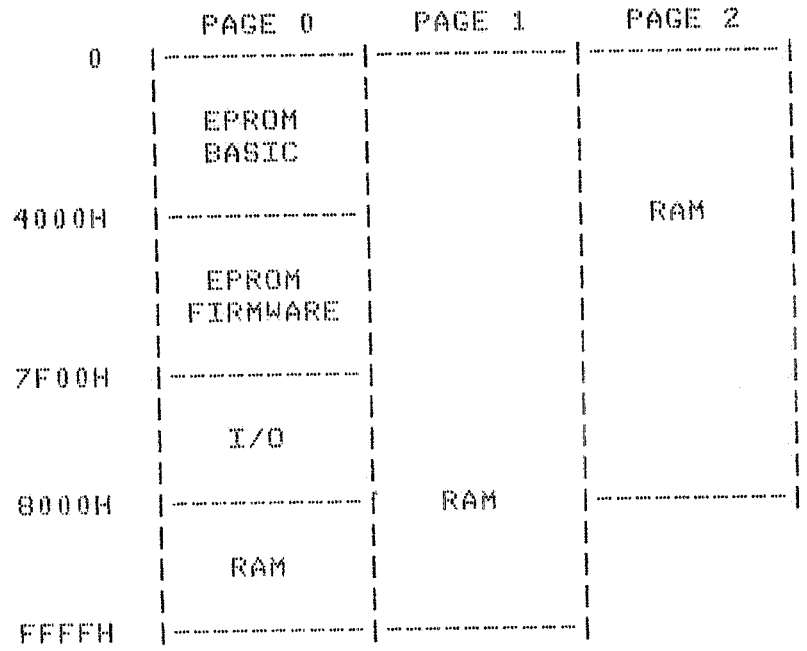

Fig. 1: CANDI2 memory configuration.

GRAPHIC REGISTERS

```
ADDRESS       REGISTER (bit 0 = LSB)

7F00          Control(WR) / Status(RD)
7F01          Control1: Bit 0: Penup(0)    / Pendown(1)
                        Bit 1: Eraser(0)   / Pen(1)
                        Bit 2: Display(0) / Nodisplay(1)
                        Bit 4,5,6 Interrupt Enable
7F02          Control2: Bit 0,1: vector characteristics:
                                 0=continuous
                                 1=dotted
                                 2=dashed
                                 3=dot-dash
                        Bit 2: Normal(0)/Italics char(1)
                        Bit 3: Horiz(0) /Vert char(1)
7F03          Char Size:Bit 0-3=Y scale; Bit 4-7=X Scale
7F05          Delta X
7F07          Delta Y
7F08          X (highest 4 bits)
7F09          X (lowest 8 bits)
7F0A          Y (highest 4 bits)
7F0B          Y (lowest 8 bits)
7F0C          XLP Light Pen X
7F0D          YLP Light Pen Y
7F10          Color and page reg  Bit 0-2: color
                                  Bit 4-6: color plane
                                  write enable
                                  Bit 7  : page
7F11          Graphic memory R/W (Red)
7F12          Graphic memory R/W (Green)
7F13          Graphic memory R/W (Blu)
```

CPU REGISTERS

```
7F99          Alphanumeric display enable (any bit)
7FC0          Mapper high byte
7FC1          Mapper low byte
7FC4          Mapper control: Bit0: Transparent(0)/
                                    Mapped(1)
                              Bit1: Nolatch    (0)/
                                    Latch(1)
7FC6          Interrupt code (Read only, bits 0-3)
7FC7          Out enable:  Bit0: Disable(0)  /Enable(1)
                           Printer Port
                           Bit1: Disable(0)  /Enable(1)
                           Terminal Output Port.
```

Fig. 2 Memory I/O configuration for CANDI2.

SOFTWARE CRU ADDRESS	REGISTER
0000H	SEL1 : 9901 PIO for CAMAC CNAF
0040H	SEL2 : 9901 PIO for CAMAC R/W,S1,S2,X,Q
0080H	Serial port B (9902) : DCE, to terminal.
00C0H	SEL3 : 9901 PIO for CAMAC R/W
0100H	CPU on board 9901 PIO : for CAMAC LAM
0140H	SEL4 : 9901 PIO for parallel link to host
0180H	SEL5 : 9901 PIO for parallel link to host
01C0H	Serial port A (9902) : DTE, to computer.
0200H	Continuous memory or memory I/O enable
0240H	Reserved for expansion
0280H	" " "
02C0H	" " "
0300H	" " "
0340H	" " "

Fig. 3

3.0 Serial I/O jumper configuration

Several configurations are possible for the serial I/O ports, depending on the setting of various jumpers on the CPU board and on the pins used on the front panel connectors Port A,B and C.

A list of the possible settings is given in the following:

3.1 PORT A (Host Port) configurations.

3.1.1 20 mA current loop - Active -

In this configuration CANDI2 generates the necessary current and the host is passive.

Jumper setting

jumper	pins connected
JE1	2 to 3

Port A Connector

pin no.	signal
25	RX+ (Receiver positive)
13	RX- (Receiver negative)
12	TX+ (Transmitter positive)
24	TX- (Transmitter negative)

3.1.2 20 mA current loop - Passive -

This configuration may be used for a host capable only of active 20 mA current loop configuration:

Jumper setting

jumper	pins connected
JE1	1 to 2
JE2	1 to 2
JE3	1 to 2

Port A Connector

pin no.	signal
18	RX+ (Receiver positive)
23	RX- (Receiver negative)
25	TX+ (Transmitter positive)
24	TX- (Transmitter negative)

3.1.3 Standard EIA RS232/C configuration -

This configuration is the normal configuration for connection to a host in the standard RS 232/C mode.

Jumper setting

jumper	pins connected	
JE1	2 to 3	
J5	2 to 3	if the host has the "Reverse channel" on Pin 11 of the Port A connector (Pin 2 of the jumper is the CTS of the UART)
	2 to 4	if the terminal has CTS on Pin 20 of Port A.

Port A Connector

pin no.	signal
1	Ground
2	TX (Transmitter)
3	RX (Receiver)
5	+12 V
6	+12 V DSR
7	Ground
8	RTS
11	Reverse channel
13	-12 V
19	Ground
20	CTS

3.2 PORT B (Terminal Port) configurations.

3.2.1 20 mA current loop - Active -

This configuration, where CANDI2 generates the necessary current and the terminal is passive, is the normal configuration for connection to a terminal in current loop mode.

Jumper setting

jumper	pins connected
JE1	2 to 3

PORT B Connector

pin no.	signal
25	RX+ (Receiver positive)
13	RX- (Receiver negative)
12	TX+ (Transmitter positive)
24	TX- (Transmitter negative)

3.2.2 Standard EIA RS232/C configuration -

This configuration is the normal configuration for connection to a terminal in the standard RS 232/C mode.

Jumper setting

jumper	pins connected	
JE1	1 to 2	
	2 to 3	if the terminal has the "Reverse channel" on Pin 11 of the Port B connector (Pin 2 of the jumper is the CTS of the UART)
J3	2 to 4	if the terminal has CTS on Pin 20 of Port B.

Port B Connector

pin no.	signal
1	Ground
2	RX (Receiver)
3	TX (Transmitter)
5	+12 V CTS (out)
6	+12 V DSR
7	Ground
8	RTS
11	Reverse channel
19	Ground
20	CTS

3.3 PORT P (Printer Port) configurations.

3.3.1 Standard EIA RS232/C configuration -

This configuration is the normal configuration for connection to a printer in the standard RS 232/C mode.

Jumper setting

jumper	pins connected	
J4	2 to 1	if the printer has the "Reverse channel" on Pin 11 of connector SCP (Pin 2 of the jumper is the CTS of the UART)
	2 to 3	if the printer has CTS on Pin 20 of SCP.

Printer Port Connector

pin no.	signal
1	Ground
3	TX (Transmitter)
6	+12 V DSR
7	Ground
8	RTS
11	Reverse channel
20	CTS

3.4 RGB Port configuration

The front panel RGB Port contains the signals for the color graphic display, as well as the signals for the optional alphanumeric keyboard. The pinout of the connector is as follows:

RGB Port Connector

pin no.	signal
3	RED
4	GREEN
5	Horizontal sync
6	Keyboard reset
7	Ground
8	BLU

10	Keyboard IN
11	+ 5 V supply
12	Sync
14	Audio out
16	14 MHz clock

4.0 Software Configuration

The CANDI2 system is based on a 16 KByte BASIC interpreter (see manual), plus 16 more KBytes of utilities (CANDI firmware).

CANDI firmware is burned into 2 64 kbit EPROM's.

It is organized as a set of machine language subroutines which can be recalled from BASIC using the statement

[sn] CALL"name" [,p1 [,p2 [,p3 [,p4]]]] <CR>

(<CR>=carriage return)

If a CALL statement is inserted in a program, it is decoded and listed as follows:

[sn] CALL"[name]",address [,p1 [,p2 [,p3 [,p4]]]] <CR>

where: sn is the statement number (not used if the subroutine is recalled in immediate mode)
name is the name of the subroutine;
address is the absolute address in memory of the subroutine;
p1... are optional input and output parameters.

Table 1 is a list of the available subroutines; a detailed description can be found in the following paragraphs.

The CALL statement can also be used to recall user assembly language subroutines: in this case the format is:

[sn] CALL"[name]",address [,p1 [,p2 [,p3 [,p4]]]] <CR>

where the parameters have the same meaning as the

TABLE 1: CANDI2 UTILITIES

NIM-ESONE CAMAC SUBROUTINES

CALL"CDREG"	Compute CAMAC address
CALL"CFSA"	Perform single CAMAC action (24 bits)
CALL"CSSA"	Perform single CAMAC action (16 bit)
CALL"CFGA"	Perform a series of CAMAC actions
CALL"CARPT"	Repeat n times the last CAMAC action
CALL"CDZER"	Clear CAMAC address buffer
CALL"CCCZ"	Perform a CAMAC Z function (initialize)
CALL"CCCC"	Perform a CAMAC C function (clear)
CALL"CCCI"	Set or reset CAMAC I (inhibit)
CALL"CTCI"	Test CAMAC I
CALL"CCCD"	Enable or disable Crate Demand
CALL"CTGL"	Test Crate Demand
CALL"CLDEC"	Decode Interrupt LAM's
CALL"CCLWT"	Wait for LAM
CALL"CDLAM"	Encode a LAM identifier
CALL"CDZEL"	Clear LAM identifier
CALL"CCLM"	Enable LAM
CALL"CCLC"	Clear LAM
CALL"CTLM"	Test LAM
CALL"CGREG"	Decode address buffer
CALL"CGLAM"	Decode LAM identifier
CALL"CFUBC"	Perform multiple CAMAC action (Q stop, 24 bit)
CALL"CSUBC"	Perform multiple CAMAC action (Q stop, 16 bit)
CALL"CFUBR"	Perform multiple CAMAC action (Q repeat, 24 bit)
CALL"CSUBR"	Perform multiple CAMAC action (Q repeat, 16 bit)
CALL"CFUBL"	Perform multiple CAMAC action (L sync, 24 bit)
CALL"CSUBL"	Perform multiple CAMAC action (L sync, 16 bit)
CALL"CFMAD"	Perform multiple CAMAC action (address scan mode, 24 bit)
CALL"CSMAD"	Perform multiple CAMAC action (address scan mode, 16 bit)
CALL"CENAB"	Enable part of a crate for LAM
CALL"CDISAB"	Disable part of a crate for LAM

HOST COMPUTER INTERACTION SUBROUTINES

CALL"SER"	Memory image transfer (RS/232 or 20 mA current loop line)
LOAD2	Ascii Basic source transfer (RS/232 or 20 mA
SAVE2	" " "
CALL"TRASP"	Transparent terminal emulation (including Tektronix 4006)

GRAPHIC SUBROUTINES

CALL"INIT" Graphic system initialization
CALL"PAGE",P Change page
CALL"BKG",BK,COL Change background color
CALL"COL",COL Change writing color
CALL"CUR" Display graphic cursor
CALL"VECA",X,Y Draw absolute vector
CALL"VECR",DX,DY Draw relative vector
CALL"VECDA",X,Y,AT Draw dotted absolute vector
CALL"VECDR",DX,DY,AT Draw dotted relative vector
CALL"MOVA",X,Y Move position absolute
CALL"MOVR",DX,DY Move position relative
CALL"PNTA",X,Y Plot point absolute
CALL"PNTR",DX,DY Plot point relative
CALL"RECT",X,Y Draw rectangle
CALL"FRECT",X,Y Draw full rectangle
CALL"CRCMF",R,X,Y Draw circumference
CALL"CIRCLE",R,X,Y Draw circle
CALL"CSIZ",SX,SY Set size for hardware character generator
CALL"STRING",HV,AT,(STRING) Draw alpha string
CALL"RGMEM",(RED),(GREEN),(BLU) Read graphic memory
CALL"WGMEM",RED,GREEN,BLU Write graphic memory
CALL"HARD" Print hardcopy
CALL"OUT",DISP,PRINT CRT display and Printer on or off
CALL"ACRT",EN CRT alpha enable

ASSEMBLY LANGUAGE PROGRAMMING SUBROUTINES

CALL"ASSEMB" TM9995 assembler
CALL"DISAS",STADD,EADD TM9995 disassembler
CALL"MEMY" Memory inspect/change/dump
CALL"TRUNC",IN,OUT Floating point to integer conversion

MEMORY MAPPING

CALL"WRMA",STADD0,NBYTES,PAGE,STADDN Write from Page 0 to Page N
CALL"RDMA",STADD0,NBYTES,PAGE,STADDN Read from Page N to Page 0
CALL"MAPPIN,PAGE,NBLOCKST,NBLOCKE Allows to use two pages contemporarily.
CALL"MAPONE",NREG,PAGEN,NBLOCK Write one register of the mapper.
CALL"WNEN" Allow addressing the I/O window (7F00H to 7FFFH).
CALL"WNDI" Disable the I/O window replacing it with continuous memory.
CALL"FILMA",DATA,NBYTES,PAGEN,STADDN Fill NBYTES of page PAGEN starting from address STADDN with the number DATA.

firmware CALL, but:
name is optional
address must be specified

4.0.1 Parameters - Input parameters to the subroutines may be passed as absolute values, variables or addresses, allowing vector transfer. Thus

```
10 DIM V(100)
20 A=1
30 CALL"SUBR",5000H,3,A,(V(0))
40 ....
```

will recall subroutine SUBR at address 5000 hexadecimal (EPROM) transferring a value of 3 into the first parameter (R4) , a value of 1 into the second (R5),and the address of the first component of vector V in R6.

Output parameters may only be transferred as addresses.

4.0.2 Number format -

Due to the difference in number formats between CAMAC and BASIC, problems may arise if floating point numbers are transferred from BASIC to CAMAC or if numbers longer than 15 bits are exchanged. A BASIC variable consists of 3 16-bit words. An integer number ranging from -32768 to +32767 is located in the second word, in two's complement.The 24 bit CAMAC word is located into memory in the last 8 bits of the first word and in the 16 bits of the second. Therefore, BASIC will recognize a number output by a CAMAC module as a regular positive integer number only if it is contained in the 15 least significant bits of the CAMAC word. If bit 16 is equal to 1, the number will be interpreted as a negative number. Also, if bits higher than the 16th are not equal to zero, the number will be erroneously interpreted as a floating point (or absurd) number. In these cases it is always possible to decode the input data using the BIT instruction of the BASIC interpreter: For example

```
A=BIT(BUF(3),9)
```

will put into A the value of the 24th(most significant) bit of the CAMAC word read into the fourth component of vector V. Instead,

```
A=BIT(BUF(3),32)
```

will do the same to the least significant bit of the input data.

When numbers are transferred from BASIC to CAMAC using a write function, the user must be sure that the number being sent is in the right format. For example,

```
A=256
```

will deliver the right number, while

```
A=2^8
```

will send a number in floating point format generating an absurd transfer. Note that a number loaded as hexadecimal will always fill only the appropriate 16 bits in the correct order. A define procedure such as

```
A=8000H
```

is strongly recommended. Loading a number as a decimal integer is only allowed for N<=32677. Moreover, the BASIC BIT function may be used to write single bits. For example,

```
BIT(BUF(1),9)=1
```

will set to one the most significant bit of the word to be transferred to CAMAC.

It is possible to transform a floating point number into a number in integer format using the subroutine TRUNC (See utilities). The subroutine will truncate the number and restitute it in integer format suitable for input output if the input is in the range

```
-32768<IN<32768
```

The calling sequence is:

```
CALL"TRUNCT",IN,(IN)
```

If the number is out of the range specified, a "FIX ERROR" will result.

4.1 Subroutine description

Available routines can be divided in five groups (see Table 1 for complete list):

1. CAMAC interface
2. Communication with the host computer
3. Graphics

4. Assembly language programming routines

5. Memory mapping

4.2 CAMAC Subroutines

The NIM-ESONE set of CAMAC interaction subroutines is becoming an accepted standard. The whole set is implemented in firmware,with the addition of a few more routines specifically written to take advantage of the peculiar features of CANDI. A list of the subroutines is shown in Table 1. We give now a short description.

1. CALL"CDREG",(EXT),N,A

   Compute and store in (EXT) the formatted CAMAC address of a register.

   (buffer EXT)=Address in memory of the formatted CAMAC address EXT.

   N=Station Number A=Subaddress

2. CALL"CFSA",F,EXT,(INT),(Q)

   Perform CAMAC action specified by F (24 bit transfer).

   F=CAMAC function Number

   (INT)=Address in memory of the input or output word

   Q=CAMAC Q response

3. CALL"CSSA",F,EXT,(INT),(Q)

   Same as CFSA but 16 bit transfer.

4. CALL"CFGA",(FA(0)),(EXA(0)),(INT(0)),(CB(0))

   Perform a sequence of CAMAC functions at a corresponding sequence of CAMAC addresses and return a sequence of Q-responses.

   (FA(0))=Address in memory of the first component of the vector containing the sequence of CAMAC functions.

EXA(I)=Vector containing the sequence of CAMAC addresses.

INT(I)=Vector containing the sequence of input or output data.

CB(0)=Number of CAMAC actions to be performed.

CB(1)=Number of CAMAC actions actually performed.

CB(2)/CB(I+2)=Q response of I th CAMAC action.

5. CALL"CARPT",N

Repeat N times the last CAMAC action performed by CFSA, where INT must be a vector if the action is a read.

6. CALL"CDZER",(EXT)

Clear the CAMAC address buffer EXT.

7. CALL"CCCZ"

Generate a Dataway Initialize (Z) and set Inhibit (I).

8. CALL"CCCC"

Generate a Dataway Clear (C).

9. CALL"CCCI",EXT,L

Set Dataway Inhibit (I) if L=1 and reset it if L=0.

10. CALL"CTCI",EXT,L

Test Dataway Inhibit and return its value in L.

11. CALL"CCCD",EXT,L

Enable Crate Demand if L=1 or disable it if L=0

12. CALL"CTGL",EXT,L

Set L=1 if Crate Demand is present or L=0 if not.

13. CALL"CLDEC",(NLM(0)),N

Decode interrupt LAM's, i.e. identify and store in vector NLM(I) the station number(s) of the modules which sent a LAM, for one of the three groups of modules. The group is specified by N.

14. CALL"CCLWT",LAM

Enter wait loop for a LAM from the address specified by `LAM`. The user must enable the LAM of the module calling the routine CCLM.

15. CALL"CDLAM",(LAM),N,M

Encode all necessary values concerning a LAM and store an identifier in LAM.

M=Subaddress if M>=0; M=negative of a bit position if M<0 (LAM accessed via Group 2 Registers).

16. CALL"CDZEL",(LAM)

Clear the identifier stored in LAM.

17. CALL"CCLM",LAM,L,(INT),(Q)

Enable if L=1 or disable if L=0 the LAM specified by LAM.

INT=Input or Output Graded LAM number

18. CALL"CCLC",LAM,(INT),(Q)

Clear LAM specified by LAM. Same parameters as CCLM.

19. CALL"CTLM",LAM,(L),(INT)

Test LAM specified by LAM and set L=1 if LAM set; L=0 if LAM absent.

20. CALL"CGREG",EXT,(N),(A)

Decode EXT and restitute N and A. Inverse of CDREG.

21. CALL"CGLAM",LAM,(N),(M)

Decode LAM identifier and restitute N and M. Inverse of CDLAM.

22. CALL"CFUBC",F,EXT,(INT(0)),(CB(0))

Block transfer. Repeat action specified by F at address EXT. 24 bits.

INT(I)=Input or output array containing CAMAC words.

CB(0)=Number of words to be transferred.

CB(1)=Number of transferred words.

23. CALL"CSUBC",F,EXT,(INT(0)),(CB(0))

Same as CFUBC but for 16 bit transfers.

24. CALL"CFUBR",F,EXT,(INT(0)),(CB(0))

Repeat action F at address EXT using Q-repeat mode and LAM stop. Syncronized by module. Same parameters as CFUBC. 24 bits.

25. CALL"CSUBR",F,EXT,(INT(0)),CB(0))

Same as CFUBR but 16 bit transfers.

26. CALL"CFUBL",F,EXT,(INT(0)),(CB(0))

Block transfer. Repeat action F at address EXT syncronized by module LAM and terminated by Q=0 or by exceeding word count (CB(0)). Same parameters as CFUBC. 24 bit.

27. CALL"CSUBL",F,EXT,(INT(0)),(CB(0))

Same as CFUBL but 16 bit transfers.

28. CALL"CFMAD",F,(EXB(0)),(INT(0)),(CB(0))

Repeat action F at a series of addresses calculated using address scan algorithm. If Q=1, increment A. If Q=0 or X=0 clear A and increment N. Terminate when end address is exceeded or word count reached.

EXB(0)=Start address

EXB(1)=End address

Other parameters as in CFUBR. 24 bits.

29. CALL"CSMAD",F,(EXB(0)),(INT(0)),(CB(0))

Same as CFMAD but 16 bit transfer.

30. CALL"CENAB",N

Enable part of a crate to generate interrupts. The crate is devided in three sections: N=1-7, N=8-15, N=16-24.

N=1,2,3 for the three sections of the crate.

31. CALL"CDISAB",N

Same as CENAB but for disabling.

4.2.1 LAM Handling -

LAM requests from CAMAC modules are routed to different interrupt levels of the system. Interrupt 12 is triggered by any LAM in the crate, while interrupts 5,6,7 are triggered by groups of 8 stations as follows:

N 1 - N 8	INT 5
N 9 - N 16	INT 6
N 17 - N 24	INT 7
N 1 - N 24	INT 12

The interrupt can be enabled in the TM 9995 by the instruction

IMASK n

where n=0 disables all interrupts, while n=4 enables them all. Note that the BASIC real time clock uses INT 3, so that the instruction

IMASK 3

will disable all external interrupts while leaving the clock running.

Moreover, since the interrupts are handled by a Parallel Input Output Interface, TM 9901, it is possible to enable or disable any single interrupt line from reaching the CPU.

The sequence is as follows:

```
10 BASE 100H    !Address on-board TM9901
20 CRB(0)=0     !Set interrupt mode
30 CRB(6)=1     !Enable only interrupt 6
40 CRB(7)=1     !Enable only interrupt 7
.............
```

The single LAM address is read and can be accessed through the on board 9901.

4.3 Communication with the host computer

Three routines are available for communication with a PDP/11 or with a VAX computer:

1. Transparent Terminal
2. Serial link for memory image file exchange
3. Serial link for source Basic file exchange and Basic printout into the host computer

4.3.1 Transparent Terminal -

This routine makes the microcomputer transparent, thus allowing use of the keyboard to talk directly to the host. All the facilities of the host are thus made available to the CANDI user. In particular, this routine can be used to complete a login procedure before a file transfer with the other three routines (and a logoff afterwards).

The routine needs the serial link (RS-232 or 20 mA current loop on port A) to be installed and can work up to 9600 baud. No software is required on the host computer, but the transfer speed of the terminal attached to the microcomputer must be the same as that of the DZ-11 channel. The speed of the first RS-232 port (B) is set during the initialization of the microcomputer, while the speed of the second port (A) is set automatically by the program (generating a bell on the terminal).

Calling sequence is:

CALL"TRASP" <CR>

To exit the transparent mode it is sufficient to type <CTRL> A.

The routine also performs, if a CTRL B character is received, a complete Tektronix 4006 emulation (see Graphic Routines).

4.3.2 Serial link file exchange -

This routine exchanges a block of data with the host computer using the RS-232 serial link (Port A). It can work up to 9600 bauds.

The files created on the host are binary files with 512 bytes block size, reproducing the microcomputer memory image. The transfer, however, occurs in ASCII mode for compatibility with the terminal driver.

A communication program (CLERKS) must be present in the host computer, and is automatically recalled by the routine. The routine can be used interactively but can also be recalled by program, which is especially useful for data transfer. (See par. 4.3.4 for the procedure to install CLERKS).

The terminal buffer size must be set to 128 bytes (when working with a PDP-11 host). Ask your system manager to initialize it this way. Calling sequence in the two cases is:

CALL"SER",RW [,(FILENAME),STARTADDR [,STOPADDR]]

If RW=0, the operator must supply the input parameters from terminal, following prompts by the program; otherwise, the parameters are supplied directly by the calling sequence. The meaning of the parameters is:

RW=1 for input from the host

=2 for output to the host

(FILENAME)=address of the vector containing the name of the file to be transferred. It is a 6-character identifier to which the program appends the necessary identifications (default UIC, Device, and last version number (last+1 for write); qualifier=(MIC).

STARTADDR=start address (hexadecimal) in the memory of the microcomputer of the file to be transferred

STOPADDR= end address (hexadecimal) in the memory of the microcomputer of the file to be transferred (only for output to the host).

The program can only transmit complete blocks, therefore a multiple of 512 bytes is always transferred. Note that, if the parameters are supplied by terminal, after program prompts, the "H" suffix to an hexadecimal number need not be used .

To use the program it is necessary to be logged onto the host, which can be done using the transparent terminal routine. Thus, a complete file exchange between MICRO and HOST would require the following sequence:

```
                CALL"TRASP"      <CR>
PDP answers                                  VAX answers
-----------                                  -----------
>                                            Username:

>...        (Login procedure)                 $...

                <CTRL> A     ( To exit the transparent
mode )

                Control returned to CANDI

                CALL"SER",0
```

The micro answers with the questions:

```
                read or write ? W          ( For write )
                filename ? PROVAA
                start address (4 hex) ? C000
                stop  address (4 hex) ? CFFF
```

File transfer is now in progress:

At the end the micro prints:

```
                0008 hex blocks transmitted
```

At this point a logoff procedure must be executed.

If the parameter RW is different from 0, the parameters are supplied by the program. For example

```
10 DIM A(1)
20 $A(0)="FILENM"
30 CALL "SER",1,(A(0)),0C000H
40 ...
```

will read from the host the file FILENM.MIC and load it into the memory of CANDI starting from C000 hex (the end address is not specified because it is defined by the length of the input file). Possible error messages are:

1. HOST NOT READY. -Fatal error-

   Problems on host computer link. CLERKS is not running.

2. INVALID ANSWER FROM HOST. -Fatal error-

   Abort CLERKS using TRASP and restart.

3. INVALID ID FROM HOST. -Fatal error-

   Abort CLERKS and restart.

4. HOST OVERFLOW. -Fatal error-

   No more free space on your directory. Purge your files and restart.

5. INVALID ECHO RETURNED FROM HOST. -Fatal error-

   Abort CLERKS and restart.

6. FILE NOT OPENED. -Fatal error-

   Reading from host. Wrong file name. Restart "SER" giving the correct name.

7. OVERFLOW. -Fatal error-

   Writing on micro. No more space in RAM memory.

4.3.3 ASCII files transfer -

A separate file interchange routine allows to transfer files in ASCII code to/from the host. This is particularly useful for source BASIC programs, where the ASCII code is generally much shorter than the corresponding memory image. Two more advantages are gained by using this type of storage: a) the BASIC code can be edited directly on the host computer; b) the code is relocatable, i.e. it can be saved and reloaded after changing the "NEW" parameter in the BASIC operating system.

The calling sequence is:

SAVE2,"filename" <CR>

to save all the Basic program source into the host computer. Filename is the name on the host to be created. The qualifier must be specified. A communication program (PAP) must be present in the host computer and is automatically recalled by the routine. (A login on the host computer must have been performed). See par. 4.3.4 for instructions on installing PAp.

The Filename can be given separately. The calling sequence is, in this case:

SAVE2 <CR>

The routine asks for the Filename.

Similarly, the calling sequence:

LOAD2,"filename" <CR>

allows to copy from the host a file containing Basic source code. Each line is sintactically analyzed and interpreted as a source from terminal. Lines with already existing line numbers are replaced.

The task PAP could be used to produce output from a Basic program into a file on the host. This is an example.

```
      10      DIM NAM(7)
      20      PRINT "HOST FILE NAME"  ! Ask for file
name
      30      INPUT $NAM
      40      BAUD 1,2                ! Set host port
at 4800 bauds
      45      UNIT 2                  ! Output to host
only
      50      PRINT "PAP"             ! Run PAP on
host
      60      PRINT "S";$NAM          ! Save operation
and file
                                      ! name
      70      UNIT 3                  ! Output to
terminal also
      80      $CD=%0DH%0              ! Define
Carriage return
      90      $CZ=%01AH%0             ! Define
Control-Z to finish
                                      ! PAP
      ........................
      200     PRINT ......;$CD        ! Only Carriage
return at the
                                      ! end of record.
PAP

Line feed
      ........................
      300     PRINT $CZ               ! Stop PAP
      400     UNIT 1                  ! Output to
terminal only
      ........................
```

The different communication programs must be installed with the names:

...CLS

...PAP

on a PDP-11

For the same purpose, the following symbols

CLS:=="RUN dev:[directory]CLERKS"

PAP:=="RUN dev:[directory]PAP"

must be defined on the VAX-11.

4.4 Graphics routines

Twentyfive graphic routines have been implemented:

1. CALL"INIT"

   Initialize graphic processor, setting white writing color on black background, X=Y=0, page 0.

2. CALL"OUT",DISP,PRINT

   Enable/disable output to the alphanumeric CRT and to the printer.

   DISP=1 (0) CRT output enable (disable)

   PRINT=1 (0) Printer output enable (disable)

3. CALL"ACRT",DISEN

   Enable (disable) alphanumeric CRT visualization when DISEN=1 (0).

4. CALL"PAGE",P

   Change page, where:

   P=0 or P=1 is the page number.

5. CALL"BKG",BKG,COLOR

   Clear screen and select background and display colors, where:

   BKG=Background color

   COLOR=Writing color

6. CALL"COL",COLOR

   Change writing color,where:

COLOR=Writing color

Note: Both BKG and COLOR in the last two subroutines can be used at various levels. A number from 0 to 7 will set the GDP to write (or erase) all the three color planes, thus destroying the old information, following the color code in table 6. If, on the contrary, it is desired to modify only one or two of the color memory planes, it is possible to do so by setting to one the 5th(red), 6th(blu) and 7th(green) bit (counting from the LSB) of BKG or COLOR. These bits disable modification of the corresponding memory planes, thus allowing simultaneous display of up to three different drawings, which can be modified without interfering with the others. A good example of such a possibility could be drawing a double sided PC board, where every color plane represents one face of the board.

For example, CALL"COL",061H will only write the red color plane.

7. CALL"CURS"

Display graphic cursor at current coordinates. To erase the cursor, call again the subroutine without changing position.

8. CALL"VECA",IX,IY

Draw line from current coordinates to (IX,IY),where:

IX=End x coordinate of the vector (can be out of the screen)

IY=End y coordinate of the vector (can be out of the screen)

9. CALL"VECR",IDX,IDY

Draw vextor from current coordinate (X,Y) to (X+IDX,Y+IDY),where:

IDX=X displacement from current position

IDY=Y displacement from current position.

NOTE: Neither VECA nor VECR change the current position.

10. CALL"VECDA",IX,IY,TYPE

    Draw vector from current coordinate to (IX,IY) according to TYPE,where:

    TYPE=0 is a solid line; =1 is a dotted line, =2 is a dashed line, =3 is a dash-dot line.

11. CALL"VECDR",IDX,IDY,TYPE

    Same as VECR with:

    TYPE same as in VECDA.

12. CALL"MOVA",IX,IY

    Update the current coordinates to (IX,IY)

13. CALL"MOVR",IDX,IDY

    Update the current coordinates (X,Y) to (X+IDX,Y+IDY)

14. CALL"PNTA",IX,IY

    Plot a point at screen address (IX,IY)

15. CALL"PNTR",IDX,IDY

    Plot a point displaced (IDX,IDY) from the current coordinates

16. CALL"RECT",DX,DY

    Draw a rectangle with sides DX,DY. Current position is bottom left corner of the rectangle.

17. CALL"FRECT",DX,DY

    Draw a full rectangle. Same as RECT for the rest.

18. CALL"CRCMF",R,X,Y

    Draw circumference with radius R and center (X,Y)

19. CALL"CIRCLE",R,X,Y

    Draw full circle with radius R and center (X,Y)

20. CALL"CSIZ",SX,SY

    Set x and y scales for hardware graphic character generator;

SX,SY=1 to 16 different sizes (1=min, 16=MAX).

21. CALL"STRING",DIR,TY,(A(0))

Write string contained in $A(N),where:

DIR=0 Horizontal string; DIR=1 Vertical string

TY=0 Normal characters; TY=1 Italics

22. CALL"RGMEM",(RED),(GREEN),(BLU)

Read one byte of the graphic memory in the current position. Place contents in RED,GREEN,BLU. The returned byte corresponds to 8 consecutive pixels lying on a horizontal line, containing the current address pixel and starting from an integer multiple of 8. The most significant bit corresponds to the leftmost pixel.

23. CALL"WGMEM",RED,GREEN,BLU

Write to one byte of the graphic memory the contents of RED,GREEN,BLU. The transferred byte corresponds to 8 consecutive pixels lying on a horizontal line, containing the current address pixel and starting from an integer multiple of 8.

24. CALL"HARD"

Print color hard copy of video display (for "PRISM" Printer).

25. CALL"TRASP"

Set up the CANDI system for serial terminal emulation with the host. The system will clear the graphic screen, and, upon receiving a CTRL-B character, will start Tektronix 4006 emulation.

The program allows, through a control character which can be sent both by the host and by the keyboard, to enter the "Tektronix emulation mode", where all the commands recognised by a Tektronix 4006 are interpreted and translated to CANDI graphics language.

On top of this, a series of new commands which allow to take advantage of the additional capabilities of the system, has been implemented. This commands are shown in tables 2 to 5.

Table 6 shows the color codes of the CANDI system.

TABLE2: SINGLE-CONTROL CHARACTER COMMANDS (FROM KEYBOARD)

ASCII Code			Keyb.Key	CANDI FUNCTION
Abbr.	Dec.	Hex.		
STX	2	2	CTRL B	ENTER TEK4006 EMULATION
EOT	4	4	CTRL D	EXIT TEK4006 EMULATION
ENQ	5	5	CTRL E	CLEAR PAGE INITIALIZE
ACK	6	6	CTRL F	BACKGROUND
DC4	20	14	CTRL T	CLEAR PAGE
ETB	23	17	CTRL W	HARDCOPY

TABLE 3 : MULTIPLE-CONTROL CHARACTER COMMANDS (FROM KEYBOARD)

ASCII Code			Keyb.Key	CANDI FUNCTION
Abbr.	Dec.	Hex.		
SYN	22	16	CTRL V	SET CHAR SIZE 1 (min)
	34	22	"	
SYN	22	16	CTRL V	SET CHAR SIZE 2 (med)
	35	23	#	
SYN	22	16	CTRL V	SET CHAR SIZE 3 (lar)
	36	24	$	
SYN	22	16	CTRL V	SET CHAR SIZE 4 (max)
	37	25	%	
CAN	24	18	CTRL X	SET COLOR
	n			

TABLE 4 : SINGLE-CONTROL CHARACTER COMMANDS
(FROM HOST)

ASCII Code			Keyb.Key	CANDI FUNCTION
Abbr.	Dec.	Hex.		
ACK	6	6	CTRL F	BACKGROUND

TABLE 5: MULTIPLE-CONTROL CHARACTER COMMANDS
(FROM HOST)

ASCII Code			Keyb.Key	CANDI FUNCTION
Abbr.	Dec.	Hex.		
ESC	27	1B	ESCAPE or CTRL [	SET CHARACTER SIZE 1
	34	22	"	
ESC	27	1B	ESCAPE or CTRL [	SET CHARACTER SIZE 2
	35	23	#	
ESC	27	1B	ESCAPE or CTRL [	SET CHARACTER SIZE 3
	36	24	$	
ESC	27	1B	ESCAPE or CTRL [	SET CHARACTER SIZE 4
	37	25	%	
CAN	24	18	CTRL X	SET COLOR
	n			SEE TABLE 6 FOR THE VALUES OF n

TABLE 6: CANDI COLOR CODES (n)

ASCII Code		Keyb.Key	CANDI COLOR
Dec.	Hex.		
48	30	0	BLACK
49	31	1	RED
50	32	2	BLUE
51	33	3	MAGENTA
52	34	4	GREEN
53	35	5	YELLOW
54	36	6	CYAN
55	37	7	WHITE

We give now a short description of the new commands.

CTRL B: This command, sent by the keyboard, directs all the following input stream from the host to the Tektronix emulation program. From this moment and until a CTRL D command is recognized, the system acts as a Tektronix 4006 terminal.

CTRL D: Exit Tek 4006 emulation mode. See CTRL B.

CTRL E: Clear page, setting black background and white writing color. Set terminal to alpha mode (Tektronix).

CTRL F (from keyboard or from host): Erase and set background using the present writing color (see CTRL X).

CTRL T: Erase without changing neither background color nor writing color.

CTRL W: Make color hardcopy of present display using Integral Data System printer type "PRISM".

CTRL V (followed by ",#,$,%): Set hardware character size 1,2,3,4. Size 1 corresponds to the hardware

character size of the Tektronix 4006; The others are bigger.

CTRL X (from keyboard or from host, followed by a number n): Set writing color according to table 6.

Four new ESCape sequences have been implemented:

ESC followed by ",#,$,%: Set character size: same as CTRL V from keyboard.

4.4.1 NEW VERSION OF GD3. -

To take advantage of the new capabilities offered by CANDI, a new version of the GD3 library has been implemented on the VAX 11/780 of the Frascati National Laboratories. A list of the new features introduced follows.

1. In the standard call CALL TVBGN(<logic unit>,<terminal>) a new terminal type has been added: <terminal>=9900. If the program is initialized with this value, the host sends a CTRL B at the beginning of the execution and a CTRL D at the end. If, on the contrary, the program is initialized with <terminal>=4006, the CANDI system must be set to Tektronix emulation mode (CTRL B) from the keyboard. Moreover, for <terminal>=9900 the addressable range is 1024 x 1024, instead of 1024 x 780, since the CANDI graphics window is a square instead of a rectangle like the Tektronix 4006. Since the CANDI resolution is 512 x 512 pixel, these addressable ranges are converted to a phisical range of, respectively, 512 x 512 pixel and 512 x 390 pixel.

2. CALL TVSET(OPT), which was a dummy for Tek 4006, is now implemented for 4 possible hardware character sizes in CANDI. OPT=large,medium, small or minimum. Note that minimum corresponds to the Tektronix size.

3. CALL TVCOLO, which was a dummy for <terminal>=4006, is now operative and performs the same function as CTRL X with a single parameter (n2).

4. CALL TVBACK is a new subroutine, which performs the same function as CTRL F.

5. CALL TVCOL99(i1,i2,...,i6) is a new subroutine which performs the same function as CTRL X; i1 - i3 are

the enable bits (i1 is the most significant=green: 0=enable); i4 - i6 are the write/erase bits (i4=green: 1 =write).

4.5 Assembly language programming subroutines

4.5.1 Memory Inspect/Change -

This routine is supplied for ease of operation to the machine language user: it dumps portions of memory giving also (when possible) ,the ASCII equivalent of each byte, or presents a location at a time allowing to modify it.

Calling sequence is:

CALL"MEMY" <CR>

4.5.2 - Number truncation

This routine truncates a number and restitutes it in integer format (the INP routine only truncates, but the number stays in the original format, which is a problem for interaction with peripheral equipment). Calling sequence is:

CALL"TRUNCT",IN,(IN)

If the number IN is not in the range -32768<IN<32767 a FIX ERROR will result

4.5.3 User assembler language subroutines - -

It is possible for the CANDI user to write his own assembler language subroutines. This can be very useful when execution speed is important, since assembler language programs execute much faster than the corresponding BASIC programs (but are more difficult to write).

To do this it is first necessary to allocate a portion of memory to the subroutines, so that the BASIC interpreter cannot destroy it. BASIC uses RAM memory

from the end down, so the command

NEW 09000H

will restrict the BASIC RAM memory to the space 9000H-FFFFH. Assembler routines can now be loaded in the space 8000H-8FFFH.

Routines can now be inserted into memory in three ways:

1. using the MEMY CANDI routine and inserting the routines directly in machine language;

2. using a cross-assembler program resident on the LNF VAX. This program interprets the Texas assembler language and produces object code, which can be directly down-loaded to CANDI using the SERIAL CANDI routine and a simple conversion program (PRINT). This program is available on request.

3. Using the CANDI2 resident assembler (see later).

4.5.4 - Assembler language I/O routines -

Assembler language terminal I/O can be accomplished by a set of 7 routines wich can be recalled with the Branch and Link instruction.

4.5.4.1 - OUTCHAR BL @>43A6

This routine writes on terminal the ASCII character found in the left byte of register 10 (R10)

No register is modified

4.5.4.2 - INCHAR BL @>43AA

This routine inputs from terminal an ASCII character and stores it in the left byte of R10.

R10 is modified.

4.5.4.3 - ECHO BL @>43AE

This routine inputs from terminal an ASCII character and echoes it to the terminal. The character is stored in the left byte of R10.

R5 and R10 are modified.

4.5.4.4 - SECHO BL @>43C2

This routine sends to the host the character stored in the left byte of R10, waiting for the echo. After sending a carriage return, waits the neccesary time to avoid writing to the host buffer before it is free.

R0, R5 and R10 are modified.

4.5.4.5 - MOUT BL @>43B2

This routine writes to terminal the ASCII string pointed to by R10. The string is ended when a null is encountered.

R4,R5,R10 are modified.

4.5.4.6 - SEND BL @>43C6

This routine sends to the host the Ascii string pointed to by R10. The string is ended when a null is encountered.

R4,R5,R10 are modified.

4.5.4.7 - HDOUT BL @>43B6

This routine writes to terminal the rightmost hexadecimal digit of R10.

R4,R5,R9,R10 are modified.

4.5.4.8 - HOUT BL @>43BA

This routine writes to terminal the binary content of R10 as 4 hexadecimal digits.

R4,R5,R6,R10 are modified.

4.5.4.9 - RHEX BL @>43BE

This routine inputs a maximum of 4 hexadecimal digits and converts them to a binary number which is stored into R10.

If <ESC> is typed the control returns to Power Basic.

If an "H" is typed, it is ignored.

If an illegal hex digit is given, an "?" is written and all preceding inputs are ignored.

If more than 4 hexadecimal digits are typed, the last ones are ignored.

The input is terminated by a <CR>, a "minus" or a "blank".

The termination character is stored in the most significant byte of R9.

R8 is loaded with 1 if at least one hexadecimal digit is given and with zero if no hex digit is input.

R0,R4,R8,R9 and R10 are modified.

4.5.5 Disassembler -

Two disassembler programs are available to restitute standard TM 990 assembler language from object code. One version is written in BASIC and is available on request. It requires interactively the start and stop absolute memory position on the micro to be disassembled. Output on host can be obtained using the procedure indicated above.

Another version is written in Assembler language and is part of the CANDI firmware.

The calling sequence is:

CALL"DISAS",start,stop,TH

where start and stop are the absolute memory addresses and TH represents the output mode, (TH=0,output to terminal; TH=1, output to host). In the second case, the program asks HOST FILE NAME?

4.5.6 Assembler - An assembler program to produce object code from source given by terminal is available. The calling sequence is:

CALL"ASSEMB",startaddress

In the following, an example is given.

The assembler works in a single pass, allowing the use of labels two characters long (the first must be alphabetic except "R", and the second must be alphanumeric).

The start address indicates where the machine language will be placed.

Statement example:

```
CALL "ASSEMB",0C000H
C000 Cxxx label MOV R0,R1        <CR>
```

The statement is constructed in the following steps:

1. The program writes C000, skips the code position and awaits for input.

2. The user writes the Label (or a space) and the instruction.

3. The program writes the machine code and the next memory address in the following line. In case of error the same address will be written again.

Before giving <CR>, the instruction can be deleted giving ESC. The program stops with the line: <space>END.

4.5.7 - Memory Mapping Routines - -

Seven routines are available to use a memory address space bigger than allowed by the CPU address bus (64 Kbytes):

1. CALL"WRMA",STADD0,NBYTES,PAGEN,STADDN

   Copy NBYTES from page 0 (Starting address STADD0) to page PAGEN (Starting from Address STADDN). At the end of this operation the mapper is not activated and the I/O window is addressable. A copy of 0 bytes can be used to disactivate any mapping.

2. CALL"RDMA",STADD0,NBYTES,PAGEN,STADDN

   As WRMA, but copying from page PAGEN to page 0.

3. CALL"MAPPIN",PAGEN,NKST,NKEND

   Activate the memory mapper. Allows to use the window NKST*1000H to (NKEND+1)*1000H-1 from PAGEN as page 0 addressable by BASIC.

   As an example,

   CALL"MAPPIN",1,7,7

   maps the CPU address window 7000H to 7FFFH to the physical memory space 17000H to 17FFFH.

   to use different non consecutive windows, call MAPPIN repeatedly.

   To return to page 0 (transparent mode: CPU address bus = memory address bus) call WRMA with NBYTES=0 (CALL"WRMA",0,0).

   At the end of the MAPPIN call, the mapper is active and the I/O window (7F**H) is unaddressable. It will be impossible, for example, to use the graphic routines.

4. CALL"MAPONE",NREG,PAGEN,NBLOCK

   Write into mapper register NREG the value R=PAGEN*10H+NBLOCK; that is, map block NREG of the CPU address bus (NREG*1000H to NREG*1000H+FFFH) to block NBLOCK of page PAGEN (16 blocks per page). Leaves the mapper activated.

5. CALL"WNEN"

   Allow addressing the I/O window (7F00H to 7FFFH).

   This routine is equivalent to the sequence:

```
10 BASE 200H
20 CRB(0)=0
```

6. CALL"WNDI"

   Disable the I/O window replacing it with continuous memory.

7. CALL"FILMA",DATA,NBYTES,PAGEN,STADDN

   fill NBYTES of page PAGEN starting from address

   STADDN with the number DATA.

Using the above routines, all of the CANDI2 memory can be used as follows:

1. To memorize data.

2. To have different BASIC programs that the CPU can use alternatively

We give an example of two separate programs that run alternatively, one from page 0 and one from page 1:

Program in page 1:

In page 0 (Mapper in transparent state)

```
NEW 0D000H
10 PRINT "PAGE ONE"
20 CALL"WRMA",0,0      !To pass to page zero
30 GOTO10              !to execute again when coming back
 from page 1
```

```
CALL"WRMA",0D000H,3000H,1,0D000H    !To copy all of the
 BASIC work area
```

```
NEW 0D000H                          !To clear page 0
10 PRINT "PAGE ZERO"
20 CALL"MAPPIN",1,0DH,0FH  !To use BASIC with Page 1
 work-area
30 GOTO10                           !To continue execution
```

```
RUN
```

The output obtained will be:

```
PAGE ZERO
PAGE ONE
PAGE ZERO
....
```

It is very important to remember that the window 4000H to 4FFFH, which contains the mapper utilities, must be always addressable in each page, in order to allow return to page 0 using the utilities above described.

REFERENCES

1. O.Ciaffoni et al.: A CAMAC system controller using the TEXAS TMS9900 microprocessor as stand-alone and PDP 11 connected unit. Frascati Report LNF - 80/27 (1980)

2. O.Ciaffoni et al.: Il nodo intelligente di acquisizione dati da CAMAC "CANDI". Frascati Report LNF - 81/25 (1981)

3. O.Ciaffoni et al.: Data acquisition system for cosmic ray muon background tests under the Gran Sasso tunnel. Frascati Report LNF - 81/36 (1981)

4. O.Ciaffoni et al.: CANDI, a microprocessor based CAMAC acquisition system with distributed intelligence features. Proceedings of the Summer School on "Data Acquisition for High Energy Physics" held in Varenna (Italy) , 1981.

5. L. Trasatti: A PROM programmer for CANDI. Int Report LNF-82/15 (1982)

6. O. Ciaffoni et al.: CANDI USER'S GUIDE, Frascati Report LNF-81/65 (1981)

7. O. Ciaffoni et al.:CANDI, a microprocessor based CAMAC acquisition system with distributed intelligence features, EURATOM-ENEA Report 81.55 (81)

8. L. Trasatti: A PROM PROGRAMMER FOR CANDI, Frascati Report LNF-82/15 (1982)

9. M. L. Ferrer: CROSS-ASSEMBLER for TEXAS TMS9900 microprocessors, Frascati Report LNF-82/28(NT) (1982)

10. R. Bertoldi et al.: A Color Graphic Display for CANDI, Frascati Report LNF-82/48(NT) (1982)

11. M. Pistoni et al.: CANDI as a Tektronix 4006 emulator, Frascati Report LNF-82/83(NT) (1982)

12. R. Bertoldi et al.: CANDI2, un sistema di acquisizione dati CAMAC a microprocessore con interfacce verso calcolatori DEC e grafica a colori, Frascati Report LNF-82/85(NT) (1982).

INDEX