O. Ciaffoni, M. Coli, M. L. Ferrer and L. Trasatti:
CANDI USER'S GUIDE.
# Index

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assembler language I/O routines</td>
<td>11</td>
</tr>
<tr>
<td>CAMAC Inhibit</td>
<td>7</td>
</tr>
<tr>
<td>CAMAC interface</td>
<td>3</td>
</tr>
<tr>
<td>CAMAC interface routines</td>
<td>5</td>
</tr>
<tr>
<td>CANDI 2</td>
<td>13</td>
</tr>
<tr>
<td>Communication with the host</td>
<td>8</td>
</tr>
<tr>
<td>Connection to the host</td>
<td>3</td>
</tr>
<tr>
<td>Disassembler</td>
<td>13</td>
</tr>
<tr>
<td>Error messages</td>
<td>10</td>
</tr>
<tr>
<td>Hardware configuration</td>
<td>1</td>
</tr>
<tr>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>LAM Handling</td>
<td>7</td>
</tr>
<tr>
<td>Memory Inspect/Change</td>
<td>10</td>
</tr>
<tr>
<td>Microcomputer</td>
<td>1</td>
</tr>
<tr>
<td>Number format</td>
<td>6</td>
</tr>
<tr>
<td>Parallel link</td>
<td>3</td>
</tr>
<tr>
<td>Parallel link file exchange</td>
<td>10</td>
</tr>
<tr>
<td>Routine Parameters</td>
<td>5</td>
</tr>
<tr>
<td>Serial link</td>
<td>4</td>
</tr>
<tr>
<td>Serial link file exchange</td>
<td>8</td>
</tr>
<tr>
<td>Software Configuration</td>
<td>4</td>
</tr>
<tr>
<td>Subroutine description</td>
<td>5</td>
</tr>
<tr>
<td>Transparent Terminal</td>
<td>8</td>
</tr>
<tr>
<td>User assembler language subroutines</td>
<td>11</td>
</tr>
</tbody>
</table>
1. - INTRODUCTION.

CANDI is a CAMAC data acquisition system\(^{(1,2,3,4)}\) based on the TMS9900 microprocessor, developed in the Frascati National Laboratories.

The system uses commercial Texas Instruments boards for the microcomputer and a CAMAC module, developed by the authors, which acts as a regular crate controller.

In addition to the usual CAMAC functions, CANDI is capable to communicate with a PDP-11 or VAX computer as an intelligent terminal. Moreover, using a parallel or serial link, the CANDI user can exchange programs and data with the host computer, thus accessing all of its facilities.

CANDI uses as high level language the Texas TM 990 Power Basic, EPROM resident, with the possibility to access assembler language routines.

2. - HARDWARE CONFIGURATION.

2.1. - MICROCOMPUTER.

The system is based on commercial Texas Instruments TM 990 system boards, and can be implemented in two different configurations:

i) Two board system. This system consists of:

- A TM 990/100 or TM 990/101 CPU board, containing the TMS 9900 CPU, I/O decoding, two serial RS 232 ports (for terminal and host computer), 4 EPROM's containing one half of the BASIC interpreter, and 1 Kbyte RAM (4 Kbytes for the TM 990/101 board).
- TM 990/201 memory expansion board, containing the remaining half of the BASIC software on EPROM and 16 Kbytes static RAM memory, plus space for 20 more Kbytes EPROM.

ii) Three board system. This system is the same as above, with the addition of
- TM 990/302 Software Development Board, containing 4 additional Kbytes of RAM memory and the possibility of recording programs and data on audio cassettes and on EPROM's.

The system memory map is shown in Table I for both configurations. Table II shows the setting of the memory map switches on the boards in the two configurations. Table III shows the jumper positions for the three boards.

### TABLE I - CANDI memory map.

<table>
<thead>
<tr>
<th>Address</th>
<th>With 302 board</th>
<th>Without 302 board</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000H</td>
<td>Basic</td>
<td>EPROM 100/101</td>
</tr>
<tr>
<td>2000H</td>
<td>Basic</td>
<td>EPROM 201</td>
</tr>
<tr>
<td>4000H</td>
<td>CANDI utilities</td>
<td></td>
</tr>
<tr>
<td>5000H</td>
<td>Free</td>
<td>RAM 301</td>
</tr>
<tr>
<td>A000H</td>
<td></td>
<td>RAM 302</td>
</tr>
<tr>
<td>B000H</td>
<td></td>
<td>RAM 101</td>
</tr>
<tr>
<td>E000H</td>
<td></td>
<td>RAM 100/101</td>
</tr>
<tr>
<td>F000H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FC00H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FFFFH</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### TABLE II - Memory map switches.

<table>
<thead>
<tr>
<th>With 302 board</th>
<th>Without 302 board</th>
</tr>
</thead>
<tbody>
<tr>
<td>201 Switches</td>
<td>302 Switches</td>
</tr>
<tr>
<td>1 OFF</td>
<td>1 OFF</td>
</tr>
<tr>
<td>2 ON</td>
<td>2 ON</td>
</tr>
<tr>
<td>3 ON</td>
<td>3 ON</td>
</tr>
<tr>
<td>4 ON</td>
<td>4 ON</td>
</tr>
<tr>
<td>5 OFF</td>
<td>5 ON</td>
</tr>
<tr>
<td>6 ON</td>
<td>6 ON</td>
</tr>
<tr>
<td>7 ON</td>
<td>7 ON</td>
</tr>
<tr>
<td>8 ON</td>
<td>8 ON</td>
</tr>
<tr>
<td>Board TM 990/101</td>
<td>Board TM 990/100</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>E1-E2 connected</td>
<td>J1 P1-18 to INT4</td>
</tr>
<tr>
<td>E3 open</td>
<td>J2-J3-J4 2716</td>
</tr>
<tr>
<td>E4-E5 connected</td>
<td>J11 installed for 20 mA current loop terminal (Main port), removed for RS232</td>
</tr>
<tr>
<td>E6 open</td>
<td>J13-J14-J15 removed</td>
</tr>
<tr>
<td>E7-E8 connected</td>
<td>J5-J6-J8 removed</td>
</tr>
<tr>
<td>E53 open</td>
<td>J9-J10-J12 removed</td>
</tr>
<tr>
<td>E9-E10 connected</td>
<td></td>
</tr>
<tr>
<td>E11-E12 open</td>
<td></td>
</tr>
<tr>
<td>E13-E14 connected</td>
<td></td>
</tr>
<tr>
<td>E15 open</td>
<td></td>
</tr>
<tr>
<td>E16-E17 connected</td>
<td></td>
</tr>
<tr>
<td>E18-E19 connected</td>
<td></td>
</tr>
<tr>
<td>E20-E21-E22 open</td>
<td></td>
</tr>
<tr>
<td>E23-E24-E25 open</td>
<td></td>
</tr>
<tr>
<td>E26-E27 connected</td>
<td></td>
</tr>
<tr>
<td>E28-E29 connected</td>
<td></td>
</tr>
<tr>
<td>E30 open</td>
<td></td>
</tr>
<tr>
<td>E31-E32 connected</td>
<td></td>
</tr>
<tr>
<td>E33-E34 connected</td>
<td></td>
</tr>
<tr>
<td>E35 open</td>
<td></td>
</tr>
<tr>
<td>E36-E37-E38 open</td>
<td></td>
</tr>
<tr>
<td>E39-E40 connected</td>
<td></td>
</tr>
<tr>
<td>E41 to E52 open</td>
<td></td>
</tr>
<tr>
<td>E54-E55 connected</td>
<td></td>
</tr>
<tr>
<td>E56 open</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Board TM 990/201</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>J1 SLOW</td>
<td></td>
</tr>
<tr>
<td>J2 SLOW</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Board TM 990/302</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>E1-E2 connected</td>
<td></td>
</tr>
<tr>
<td>E3-E4 open</td>
<td></td>
</tr>
<tr>
<td>E5-E6 connected</td>
<td></td>
</tr>
</tbody>
</table>

2.2. - CAMAC INTERFACE.

In addition to these boards, which are housed in a standard Texas crate, a 40 lead cable plugs onto connector P4 of the CPU board to connect the microcomputer to the CAMAC CAMAC interface. This unit is housed in a double width CAMAC module and plugs directly into stations 24 and 25 of the CAMAC crate, in the standard crate controller position. The front panel of the CAMAC interface houses a 40 pin connector for the microcomputer cable, and two 40 pin card edge connector sockets for the parallel link to the host computer.

2.3. - CONNECTION TO THE HOST.

Connection to the host computer is possible in one of two ways:

2.3.1. - Parallel link.

The parallel link connects via two 40 lead cables the CAMAC interface to a DR-11A board resident inside a PDP/11 or VAX computer. It is fast (12 Kbit/sec with an interrupt handling program and 200 Kbit/sec with a dedicated host on a PDP 11/34), but requires to be physically close to the host (≤ 30 m).
2.3.2. - Serial link.

The serial link uses a 4-lead cable to connect the second RS-232 port (P3) on the right hand side of the CPU board to a DZ-11 port on the host computer. If no EIA RS-232 DZ-11/A is available, a simple interface permits to use a 20 mA current-loop port on a DZ-11/C.

3. - SOFTWARE CONFIGURATION.

CANDI firmware is burned into two TMS 2716 (Texas Instruments) EPROM's plugged into the TM 990/201 board.

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

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

(<CR> = carriage return)

where: name is optional and is the name of the subroutine;
address is the absolute address in memory of the subroutine (see Table IV);
p1... are input and output parameters (see Table IV).

<table>
<thead>
<tr>
<th>Routine name</th>
<th>Address</th>
<th>Parameters and descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERIAL</td>
<td>4000H</td>
<td>p1 = 0 other parameters will be input from keyboard</td>
</tr>
<tr>
<td></td>
<td></td>
<td>p1 = 1 read from host</td>
</tr>
<tr>
<td></td>
<td></td>
<td>p1 = 2 write into host</td>
</tr>
<tr>
<td></td>
<td></td>
<td>p2 = Filename address</td>
</tr>
<tr>
<td></td>
<td></td>
<td>p3 = Memory start adress</td>
</tr>
<tr>
<td></td>
<td></td>
<td>p4 = Memory stop address</td>
</tr>
<tr>
<td>PARALL</td>
<td>4006H</td>
<td>same as SERIAL</td>
</tr>
<tr>
<td>TRASP</td>
<td>400CH</td>
<td>no parameters</td>
</tr>
<tr>
<td>MEMISC</td>
<td>4012H</td>
<td>no parameters</td>
</tr>
<tr>
<td>ZCAM</td>
<td>4018H</td>
<td>no parameters</td>
</tr>
<tr>
<td>CCAM</td>
<td>401EH</td>
<td>no parameters</td>
</tr>
<tr>
<td>RDCAM</td>
<td>4024H</td>
<td>p1 = Station number N</td>
</tr>
<tr>
<td></td>
<td></td>
<td>p2 = Number of subaddresses NA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>p3 = Function F</td>
</tr>
<tr>
<td></td>
<td></td>
<td>p4 = Data array address</td>
</tr>
<tr>
<td>WRCAM</td>
<td>402AH</td>
<td>p1 = Station number N</td>
</tr>
<tr>
<td></td>
<td></td>
<td>p2 = Address number A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>p3 = Function F</td>
</tr>
<tr>
<td></td>
<td></td>
<td>p4 = Data array address</td>
</tr>
</tbody>
</table>
3.1. - 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.

3.2. - SUBROUTINE DESCRIPTION.

Available routines can be divided in three groups (see Table IV for complete list):

1. CAMAC interface;
2. Communication with the host computer;
3. Memory inspect/change.

3.2.1. - CAMAC interface routines.

Four CAMAC interface routines are available:

1. CALL"ZCAM", 4018H
   Perform a CAMAC initialize (Z);
2. CALL"CCAM", 401EH
   Perform a CAMAC clear (C);
3. CALL"RDCAM", 4024H, N, NA, F, (BUF(0))
   Perform a 24 bit CAMAC read function, where:
   N = CAMAC station number,
   NA = Number of subaddresses to be read, always starting from A = 0,
   F = CAMAC Function,
   BUF(0) = Q-response,
   BUF(1) ... BUF(NA) = Data from subaddresses A = 0 ... A = NA-1 respectively;
4. CALL"WRCAM", 402AH, N, A, F, (BUF(0))
   Perform a 24 bit CAMAC write operation, where:
   N = Station number,
   A = Subaddress to be written (single operation only),
   F = CAMAC Function,
   BUF(0) = Q-response,
   BUF(1) = Number to be written.

Note that the Read routine performs a multiple address read operation, to save time for calls from BASIC in an operation which is quite usual.
3.2.2. - 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 -32678 to +32677 is located in the second word, with the most significant bit representing the sign bit. 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

\[ A = \text{BIT}(\text{BUF}(3), 9) \]

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,

\[ A = \text{BIT}(\text{BUF}(3), 32) \]

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

\[ A = 100H \]

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

\[ \text{BIT}(\text{BUF}(1), 9) = 1 \]

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

A new set of CAMAC interaction subroutines is being implemented; these routines will conform to the standard NIM/ESONE for CAMAC subroutines"
3. 2. 3. - CAMAC Inhibit (I).

CAMAC Inhibit may be set using the sequence:

```
BASE 120H
CRB(13) = 0
```

and reset by

```
BASE 120H
CRB(13) = 1
```

3. 2. 4. - LAM Handling.

LAM requests from CAMAC modules are routed to different interrupt levels of the TM 9900
CPU. Interrupt 12 is triggered by any LAM in the crate, while interrupts 5, 6, 7 are triggered
by groups of 8 stations of numbers:

```
N 1 - N 8     INT 5
N 9 - N 16    INT 6
N 17 - N 23   INT 7
N 1 - N 23    INT 12
```

The interrupt level can be decided in the TM 9900 by the instruction

```
IMASK n,
```

which enables all interrupt levels from 0 to n.

For example, the instruction

```
IMASK 6
```

will enable both INT 5 and INT 6.

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.

This is an example of a complete interrupt handling program:

```
10 TRAP 6 to 1000 ! Define interrupt subroutine
20 IMASK 6      ! Enable up to interrupt level 6 on the 9900 CPU
30 BASE 100H    ! Enable only INT 6 on the 9901
40 CRB(0) = 0
50 CRB(6) = 1
60 BASE 120H    ! Reset CAMAC Inhibit
70 CRB(13) = 1
80 GOTO 80      ! Wait loop.
1000 BASE 120H   ! Set CAMAC Inhibit
1010 CRB(13) = 0
1020 ...        ! Interrupt service
1080 BASE 120M   ! Reset Inhibit
1090 CRB(13) = 1 ! Reset Inhibit
1100 IRTN        ! Return from interrupt subroutine.
```

The single LAM address is read and can be accessed through the on board 9901(1).
3.3. 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 file exchange,
3. Parallel link file exchange.

3.3.1. 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 two routines (and a logoff afterwards).

The routine needs the serial link (RS-232 on port P3) 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 (P2) is set during the initialization of the microcomputer, while the speed of the second port (P3) is set automatically by the program (generating a bell on the terminal).

Calling sequence is:

CALL"TRASP", 400CH <CR>.

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

3.3.2. Serial link file exchange.

This routine exchanges a block of data with the host computer using the RS-232 serial link (Port P3). It can work up to 9600 baud.

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.

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"SERIAL", 4000H, 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:

```
PDP answers <CR> VAX answers
> > ... (Login procedure) $ ...
   <CTRL>A (To exit the transparent mode)
   Control returned to CANDI
   CALL:"SERIAL", 4000H, 0
```

The micro answers with the questions:

read or write ? 2 (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 done.

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:"SERIAL", 4000H, 1,(A(0)), 0C000H
40 ... 
```
will read from the host the file FILEMN, MIC and load it into the memory of CANDI starting from C000 (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 CLERKS.

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 "SERIAL" giving the correct name.

7. OVERFLOW. - Fatal error -
   Writing on micro. No more space in RAM memory.

8. ATTEMPT TO WRITE ON PROTECTED AREA. - Warning -
   Writing on micro, attempt to write on workspace area. The procedure continues without overwriting this area. (A000H-A01FH for systems with 302 board and B000H-B01FH for systems without 302 board).

3.3.1. Parallel link file exchange.

This routine exchanges a block of data with the host computer using a parallel link (two 40-lead cables) to a DR-11 interface placed inside the host.

The exchange speed is much higher than for the serial link, but the cables cannot be longer than 30 m.

A communication program (CLERKP) must be present in the host computer, and is automatically recalled by the program.

Using a PDP-11 host, the serial link must, however, be installed if the routine must be recalled by a program without operator intervention.

The calling sequence is identical to that for the serial link, except for the address:

CALL"PARALL", 4006H, RW [(FILENAME), STARTADDR [, STOPADDR]].

3.3.4. 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"MEMISC", 4012H <CR>.

3.3.5. - 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 will 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 0C000H
```

will restrict the BASIC RAM memory to the space C000-FFFF. Assembler routines can now be loaded in the space B000-BFFF (without the TM 990/302 board) or in the space A000-BFFF for the configuration with TM 990/302.

Routines can now be inserted into memory in two ways:
1. Using the MEMISC CANDI routine and inserting the routines directly in machine language.
2. Using a cross-assembler program resident on the CINECA CDC 6600 (TMSASM). This program interprets the Texas assembler language and produces object code, which can be directly down-loaded to CANDI using the SERIAL or PARALLEL CANDI routines and a simple conversion program (PRINT for the PDP 11/34). This program is available on request. The intercomputer network communication required is shown in Fig. 1.

![Diagram](image)

**FIG. 1** - Configuration for direct down-loading.

3.3.6. - Assembler language I/O routines.

Assembler language terminal I/O can be accomplished by a set of 7 routines which can be recalled with the Branch and Link instruction.
3.3.6.1. - **OUTCHAR** BL C>4030.
This routine writes on terminal the ASCII character found in the left byte of register 10 (R10).

No register is modified.

3.3.6.2. - **INCHAR** BL C>4034.
This routine inputs from terminal an ASCII character and stores it in the left byte of R10.
R10 is modified.

3.3.6.3. - **ECHO** BL C>4038.
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.

3.3.6.4. - **MOUT** BL C>403C.
This routine writes to terminal the ASCII string pointed to by R10. The string is ended when a null is encountered.
R4, R5 and R10 are modified.

3.3.6.5. - **HDOUT** BL C>4040.
This routine writes to terminal the rightmost hexadecimal digit of R10.
R4, R5, R9 and R10 are modified.

3.3.6.6. - **HOUT** BL C>4044.
This routine writes to terminal the binary content of R10 as 4 hexadecimal digits.
R4, R5, R6 and R10 are modified.

3.3.6.7. - **RHEX** BL C>4048.
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, they 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.
3.3.7. Disassembler.

A disassembler program\(^{(6)}\) is available to restitute standard TM 990 assembler language from object code. The program is written in BASIC.

3.4. CANDI 2.

A new version of CANDI is in advanced state of realization. The entire microcomputer will be assembled on a CAMAC board which will be housed inside the CAMAC interface. A new, faster CPU will be used (TMS 9995), and the computer will support floppy disks, memory mapping and colour display graphic capability for 512 x 512 pixels. HDLC communication line protocol for serial data links between CANDIs at a rate of 1 MBaud will also be available. The languages available will include UCSD Pascal, Basic and Fortran. The new unit, however, will be made compatible with CANDI 1, so that software developed for CANDI 1 will be immediately transportable to the new system.

REFERENCES.


(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).

