ISTITUTO NAZIONALE DI FISICA NUCLEARE
Laboratori Nazionali di Frascati

LNF-85/8(NT)
26 Marzo 1985

G. Di Pirro and L. Trasatti:

AN INTELLIGENT MASS STORAGE SYSTEM FOR CANDI 2

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

INFN - Laboratori Nazionali di Frascati
Servizio Documentazione

LNF-85/8(NT)
26 Marzo 1985

G. Di Pirro and L. Trasatti:

AN INTELLIGENT MASS STORAGE SYSTEM FOR CANDI 2

CANDI2 is an intelligent CAMAC crate controller[1-15] developed in the INFN Frascati National Laboratories, with high resolution graphics and the possibility to interact with host computers of the PDP/11-VAX family.

The system was originally designed to be an element of a data acquisition set up based on a host computer.

The necessity to use CANDI2 independently on a big computer forced us to design a mass storage system to store data and programs.

We chose floppy disks as mass storage media because of their large diffusion and low cost. Having made this choice we were faced with two alternatives: either build our own interface board (and write the necessary software), or use a commercially available one. Since the price on the market of a non intelligent floppy disk driver is not very different from the price of a complete microcomputer board, we decided to buy a complete system and to adapt it to our needs.

The problem was thus transformed to the implementation of a fast communication between two intelligent systems. Another advantage of this choice is the possibility to use all of the software available on the mass storage system. We selected a modular Z-80 system with 64 KBytes of memory, capable to implement the CP/M operating system. This operating system is easily adaptable to different hardware because the I/O primitives can be modified without changing the upper levels of software. In particular, once implemented the system for floppy disks, it is straightforward to extend it to other kinds of storage, e.g. Winchwester hard disk drives.

Once we decided on this approach, we started considering the advantages of turning the roles around, and making the CP/M system master of the configuration, with the possibility to use the vast selection of software (compilers and interpreters, in particular) available on the market, and to drive the dedicated CANDI2 peripherals (CAMAC, Graphics, VAX communication).

1.- PHYSICAL CONNECTION

The communication between the two systems was implemented using two parallel input output peripherals, to insure maximum speed and flexibility. Three 8-bit ports were used, one for byte read, one for byte write and one for control signals.

2.- SOFTWARE

The software developed is as follows:

a) A program on CANDI2 to talk directly to the CP/M system: in this situation CANDI2 is transparent.

b) Programs to transfer data (or program sources) from CANDI2 to CP/M and viceversa: in this situation CANDI2 is master.

c) Programs that transform CANDI2 into a slave: in this configuration we can use the CANDI2 resources from programs written on CP/M. Note that in all cases the terminal and keyboard are always connected to CANDI2.

2.1.- Low-level software

The communication is controlled by a handshake protocol. The I/O routines of the CP/M system were modified defining the parallel port as the system console. Primitives were written for CANDI2 to interact with the parallel ports (which already existed in the system on the CAMAC interface board).

3.- USER SOFTWARE

3.1.- Transparent terminal

This program makes CANDI2 transparent between the terminal and the CP/M system, redirecting all terminal I/O directly to the parallel ports. In this configuration the terminal appears as the CP/M system console.

To use this program we must write:

CALL "CPM" <CR>

we obtain the answer:

a) if it is the first call: the standard message from the CP/M system;

b) for successive calls: we return where we stopped the communication.

In all this cases we can use all the facilities of the CP/M.

To return to work on CANDI2 we must send a CTRL-A. This control character is interpreted as a command and not sent to the CP/M system. The answer is the message:

"""" CONTROL RETURNED TO CANDI2 FROM CP/M """" .

This program allows to work on the CP/M directly, but not to transfer data between the two systems. For this purpose we wrote two programs to transfer memory areas and ASCII programs.

3.2.- Memory image transfer

The program that allows to transfer CANDI2 memory images to the CP/M system uses as its counterpart a system program of the CP/M: the peripheral interchange program PIP.COM. The program can work both interactively and through a call from a CANDI2 BASIC program; the information is transferred as ASCII characters. We can work in two ways:

a) interactive:

CALL "PARCPM" <CR> .

In this case the parameters are requested interactively as follows:

QUESTION	ANSWER
READ (R) OR WRITE (W)?	FC
FILENAME (6 CHARACTERS)?	FILENAME
START ADDRESS (4 DIGITS)?	STADD
END ADDRESS (4 DIGITS)?	EADD

where:

FC : is the parameter to distinguish the read or write operation. The answer must be R for read and W for write.

FILENAME: the name of the file we want to read or write from/to CP/M. The qualifier (.MIC) is inserted automatically from the program.

STADD : start address in CANDI2 memory of the block to be transferred. This parameter is requested for both write and read operations.

EADD : end address in CANDI2 memory of the block to be transferred. This parameter is requested only to write data to the CP/M system.

b) Program calls:

This is an example of a program using this routine:

```
10 DIM FN(2)
20 $FN=<FILENAME>
30 CALL "PARCPM",FC,(FN(0)),STADD,EADD
```

the meaning of the parameters is the same as before except for FC that must be 1 for a read or 2 for a write and for the variable $FN which must be 6 characters long.

3.3.- BASIC program source transfer

Using the PARCPM program we can also transfer a BASIC program by saving the whole RAM workarea. However, this is very slow, especially for short programs, and moreover to reload the program to CANDI it is necessary to know the amount of memory which was originally allocated to the BASIC system. It is much more efficient to transfer a program as a list of the source statement, and a program to do this has been implemented.

To use this program we write the command:

a) to read a file from floppy:

```
LOAD1
```

b) to write a file to floppy:

```
SAVE1
```

In both cases we obtain on the terminal:

```
DISK FILE NAME?
```

In this case the file name qualifier must be specified. Like to PARCPM, the program PIP.COM must be present on the disk.

The program is saved in the form of source instructions, and during the load process a syntactical analysis is performed.

4.- THE CP/M AS MASTER

The programs described up to now use the CP/M system only as mass storage; this is a limitation, considering the vast possiblities of the system (e.g. compilers and interpreters).

To be able to use these facilities it is necessary to write a program to transform CANDI2 into a slave respect to CP/M.

Assume a situation where we are working on the CP/M system (for example using the CBASIC compiler) using the terminal connected to CANDI2 (which is working in transparent mode).

If we want the BASIC program on CP/M to have access, for example, to the CAMAC crate containing CANDI2, we must signal to CANDI2 that some string of commands coming from CP/M is not directed to the terminal in transparent mode, but must be executed by the system. To do this we use a control character to instruct CANDI2 to accept input from the CP/M, to execute the relative command, and to return to the transparent mode after execution. We have modified the transparent program so that it can perform this task. It is necessary to start from the transparent program because the terminal is always connected to CANDI2, therefore every action performed on CP/M has to pass through it.

The programs we developed allow:

a) sending commands as ASCII strings (CANDI2 BASIC immediate commands);
b) calling from CP/M the CANDI2 assembler routines;
c) transferring data between memory areas of the two systems.

4.1.- Sending direct commands to CANDI2

a) On CANDI2 we have modified the I/O routines to switch from the serial to the parallel port when the CP/M sends the command.
b) On CP/M we have written two programs: one to send the master command and the other to wait until the command has been executed.

As an example, to send the command "command line" to CANDI2, we must write a BASIC program on the CP/M system (using the transparent connection):

```
10 CALL OPC2
20 PRINT "command line"
30 CALL CLC2.
```

OPC2 puts CANDI2 in the slave condition while CLC2 waits for the completetion of the command.

4.2.- Calls to CANDI2 system routines

Using this method we can execute all CANDI2 commands. However, if we want to work only with the CANDI2 routines this program is slow because CANDI2 must interpret every line of code. For this reason we have written a program that can call the CANDI2 system routines, thus using all of its facilities. The names of the routines are the same as in the CANDI2 system, with the same syntax. The only limitation is that the parameters must be defined as integers (% variables).

The syntax is the same as for calls to CP/M routines, because the CP/M is modified to recognize the CANDI2 routine names. For example, the program

```
10 A% = 100
20 B% = 200
30 CALL VECA (A%, B%)
```

will draw a vector to the point (100,200) from the current graphic position.

4.3.- Memory transfer

When the parameters to be transferred are vectors, or in any case in which the CANDI2 subroutines use CANDI2 memory areas, it is necessary to transfer memory areas between the two systems. This program is called CANMEN. The call is as follows:

```
CALL CANMEN (DS%,RW%,ADD1%,ADD2%,ADD3%)
```

The meaning of the parameters is:

a) DS% specifies ASCII characters (0) or data (1).
b) RW% specifies if we want to read (1) or to write (2).
c) ADD1%,ADD2%,ADD3% give the initial and final addresses as follows:
   i) for ASCII character strings ADD1 is the start address on CP/M; ADD2 is the start address on CANDI2.
   ii) for numeric data:
      - in a write operation ADD1 is the start address on CP/M; ADD2 is the end address on CP/M; ADD3 is the start address on CANDI2;
      - in a read operation ADD1 is the start address on CP/M; ADD2 is the start address on CANDI2; ADD3 is the end address on CANDI2.

The addresses relative to the CP/M system are passed indirectly, while the addresses relative to CANDI2 must be passed as integer numbers (absolute). For example, the following program will transfer from CP/M to CANDI2 one thousand numbers:

```
10 DIM A%(1000)
20 DS% = 1
30 RW% = 2
40 D% = 33000
50 CALL CANMEN (DS%,RW%,A%(0),A%(1000),D%).
```

The addition of the CP/M coprocessor to the CANDI2 system has resulted not only in the addition of a mass storage system, but in a much more powerful and flexible data acquisition system. We feel that intelligent peripherals are an economic and powerful way to build up a system.

REFERENCES

(1) O.Ciaffoni, M.Coli, M.L.Ferrer, S.Li Causi and A.Villalba, 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, M.Coli, M.L.Ferrer and L.Trasatti, Il nodo intelligente di acqusizione dati da CAMAC "CANDI", Frascati Report LNF-81/25 (1981).

(3) L.Trasatti, A disassembler for Texas 9900 microcomputers, Frascati Report LNF-81/31 (1981).

(4) O.Ciaffoni, M.L.Ferrer and L.Trasatti, Data acquisition system for cosmic ray muon background tests under the Gran Sasso tunnel, Frascati Report LNF-81/36 (1981).

(5) O.Ciaffoni, M.Coli, M.L.Ferrer and L.Trasatti, CANDI, a microprocessor based CAMAC acquisition system with distributed intelligence features, From "Data acquisition in High Energy Physics", 1983, LXXXIV Corso, Società Italiana di Fisica, Bologna.

(6) O.Ciaffoni, M.Coli, M.L.Ferrer and L.Trasatti, 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 (1981).

(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 (1982).

(10) R.Bertoldi, O.Ciaffoni, M.Coli and L.Trasatti, A color graphic display for CANDI, Frascati Report LNF-82/48 (1982).

(11) R.Bertoldi, O.Ciaffoni, M.Coli, M.L.Ferrer, A.Martini and L.Trasatti, CANDI2, un sistema di acquisizione dati CAMAC a microprocessore con interfacce verso calcolatori DEC e grafica a colori, Frascati Report LNF-82/83 (1982).

(12) M.Pistoni and L.Trasatti, CANDI as a Tektronix 4006 emulator, Frascati Report LNF--82/85 (1982).

(13) O.Ciaffoni, M.Coli, M.L.Ferrer and L.Trasatti, CANDI2: Description and User Manual, Frascati Report LNF-83/67 (1983); to be published in Nuclear Instr. and Meth.

(14) O.Ciaffoni, M.Coli, M.L.Ferrer and L.Trasatti, CANDI: acquisizione dati mediante microprocessori, Presented at the LXIX Congresso Nazionale della Società Italiana di Fisica, Messina, October 1983, Bollettino della SIF (1983), p. 130.

(15) L.Trasatti, A graphic histogram subroutine for CANDI2, Frascati Report LNF-84/18 (1984).