SPS/ACC/PvdS/Report 80-27

14 October 1980

TRANSPARENT ACCESS TO MICROPROCESSORS

P.D.V. van der Stok

Contents

Paper presented at the NOCUS meeting, Helsingör,
15-17 October, 1980

Introduction

The SPS Division has developed a CAMAC module which incorporates the the TMS 9900[1] microprocessor into the Auxiliary Crate Controller (ACC)[2] to permit autonomous data processing in a CAMAC crate attached to a host computer. The arrangement is shown in fig. 1. A NORD-10 or NORD-100 computer is linked through a NORD Crate Controller to the CAMAC crate which contains the ACC.

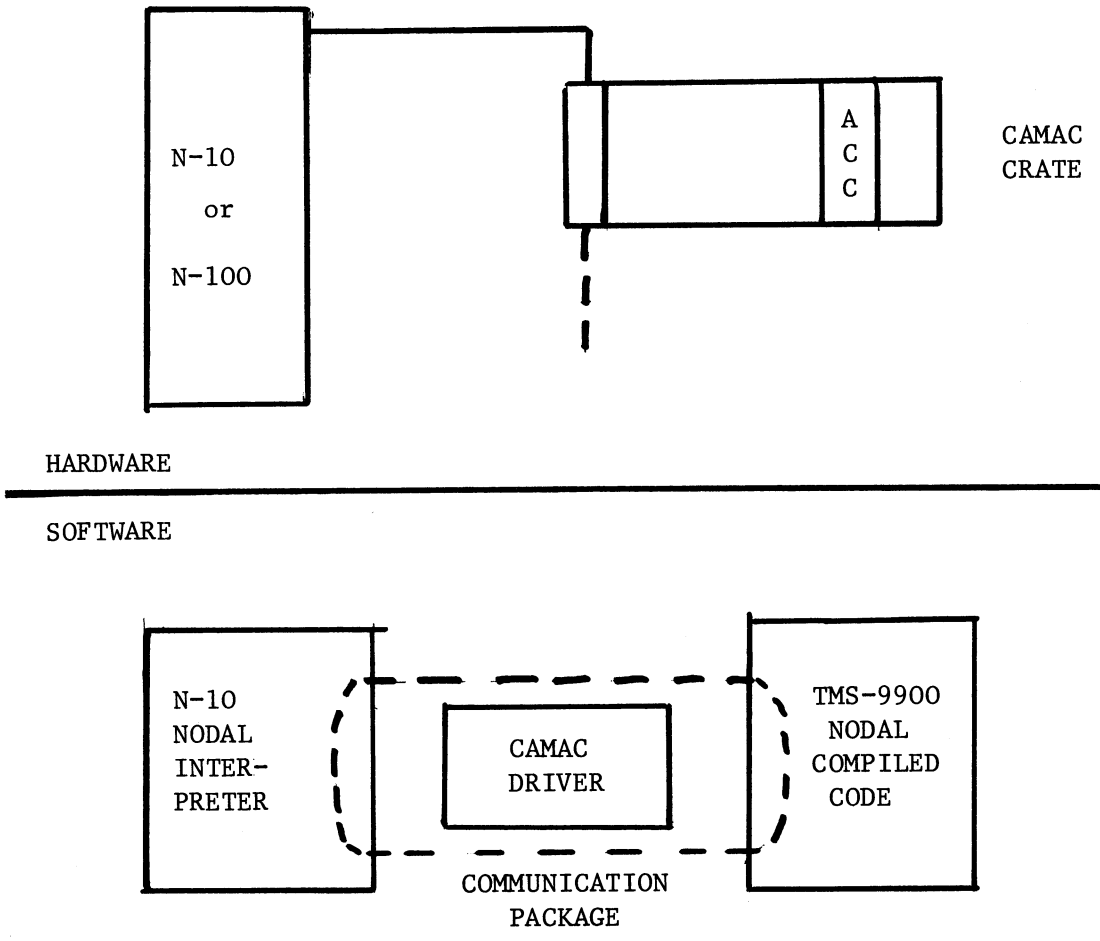

Figure 1

This arrangement can be considered for parallel as well as for serial CAMAC crates.

As well as the NORD host the ACC is capable of accessing all CAMAC modules housed in the same crate.

This paper describes how the ACC is used to unload the NORD CPU by allowing the ACC to control the equipment in an autonomous way. The NORD computer communicates with the ACC as if the logic situated in the ACC were resident in NORD. The software layout is also shown schematically in fig. 1. A Nodal interpeter[3] resident in the NORD allows the calling of user dependent functions to control the equipment. This so-called DATA-MODULE[4,5] will then

perform the specified action on the equipment and its associated tables. The ACC makes it possible to place the code for this function in the TMS-9900 microprocessor and execute it there. While this function is executed the NORD CPU is free for other purposes. The NODAL programs do not see any difference between a call to a DATA-MODULE either resident in the NORD computer or in the ACC.

The DATA-MODULE code is written in NODAL language and compiled on the NORD machine. Communications between the NORD and ACC passes through a standard communication package which uses the CAMAC driver.

1. The Auxiliary Crate Controller (ACC)

The ACC-SPS 2420 is a one width CAMAC module containing a TMS-9900 microprocessor produced by Texas Instruments. The ACC conforms to ESONE 6500 specifications for the communication with standard SCC-L2 and A2 CAMAC controllers. It is moreover compatible with CERN's standard NORD dedicated CAMAC crate interface. A teletype port, a programmable real time clock and a minimal LAM handing facility are available.

The TMS-9900 has a 64 k byte direct memory addressing scheme. These 32 kwords are divided into two distinct 16 k blocks. The lower block is implemented in an on-board memory while the higher 16 kword block represents the NAF field of the crate where the ACC resides. Consequently the CPU can execute all of its instructions without distinction between memory and CAMAC equipment addresses. The 16 k Ram can be partially replaced by EPROM (by internal bridges, up to 4 k in steps of 1 k).

Any memory location can be written or read from the NORD host computer while the microprocessor executes instructions.

For reading a memory location an internal register is set to the appropriate address value by one CAMAC command. Another CAMAC command enables the reading of one word from the memory modules, and automatically increments the memory pointer. A similar sequence applies for a write into the memory.

Four interrupts can be generated from four connections on the front panel. A programmable interrupt is raised with one CAMAC command while the ACC can also generate a LAM.

2. The NODAL interpreter and compiler

NODAL is a high level programming language, based on FOCAL and SNOBOL, designed for interactive use by non-programmer specialists for the daily control of the SPS accelerator. The most important aspect of NODAL in this context is that accelerator control programs can be written, debugged and modified interactively by accelerator physicists, engineers and technicians in a much faster and easier way than with many other computer based languages.

The NODAL interpreter executes several types of commands. In the table below the list of commands for creating or modifying (actually the same type of operation) the value of a variable is given :

ASK	sets variable to value typed on terminal
SET	sets variable to value of an expression
DIM	creates a floating point array
DIM-INT	creates an integer array.

NODAL works exclusively on floating numbers and when integer values are needed, the floating point number is converted to an integer.

GOTO	go to line
DO	execute line or group as subroutine
RETURN	exit from DO call
END	end program.

Commands for decision and loops are:

IF	conditional execution or branch
FOR	loop execution
WHILE	conditional execution-

Each program line is identified by a unique line number in the range of 1.01 to 99.99. Program lines which have the same number to the left of the point constitute a group. In the following example, contrived to show facilities rather than to perform a specific task, the program is made up of two groups: one and three.

```
1.10        ASK 'START VALUE' XS; IF XS<0; SET Y = 1.10 : GOTO 1.9
1.15        ASK 'END VALUE' XE; IF XE<XS; SET Y = 1.15; GOTO 1.9
1.20        FOR X = XS, XE; DO 3
1.25        GOTO 1.10
1.90        TYPE 'WRONG LIMIT'; GOTO Y
1.99        END

3.10        TYPE "X, EXP(X)=" X EXP(X) !
3.15        RETURN
```

The group structure allows the execution of sub-sets of program as a subroutines by means of the DO command. The subroutine calls can be made recursive and nested to any depth. A useful capability of the DO command is that it is possible to specify exception handling. For example, the statement DO 2!3 executes group 2, and when an error has been detected in the execution of group two, group three is executed.

The NODAL cross compiler[6] compiles code for a virtual machine. The associated virtual machine instructions are then translated into code for the target machine. For each target machine one code generator is needed. All arithmetic instructions operate on a three-word floating point number. Consequently the results obtained from the NODAL compiled code in the target machine are identical to the results obtained under the NODAL interpreter in the NORD machine.

The compiler runs under the SINTRAN III operating system on a NORD computer. Code generators are available for the TMS-9900[7], the M68000[8] and the NORD[9] itself. The compilation of NODAL code imposes restrictions on the language, for example, the command

GOTO expression

should be GOTO N

where N is a line number.

Several commands are available for providing flexibility in the generation of the NODAL code wanted in the ACC.

- Command COMPILE will create a target machine object file from a NODAL source program.

- Command SYTBL processes a SYstem TaBLe file, which contains information about the functions available in the NODAL programs.

- Command SYSVR creates a list of SYStem VaRiables which are loaded into the ACC memory together with the compiled programs. These variables have exactly the same structure as the variables used by the NODAL interpreter. This permits the loading of the variables into the storage area allocated to the NODAL interpreter in the NORD and allows them to be inspected there. Thus the execution of the NODAL programs in the ACC can easily be reconstructed in the host NORD machine.

- Command DATAMOD compiles a NODAL program and packs it in memory so that it can be accessed as a DATA-MODULE.

3. The DATA-MODULE concept

Contrary to the traditional approach to equipment control, where a central data base is kept up to date by the control computer on a regular refresh cycle, the SPS equipment control uses another mechanism. The data belonging to a specific kind of equipment are kept local to the computer to which the equipment is connected.

This solution implies the splitting up of the equipment in groups of the same type and same access requirements. They are grouped in software routines, called DATA-MODULES, which are accessed through a rigidly-defined calling sequence. This device-oriented method directs all device actions through the same software module.

A DATA-MODULE has two parameters. The equipment number and the property. The equipment number refers to a specific element of the equipment group. The numbering of the equipment in one group is sequential and local to the computer to which the equipment is connected.

The property parameter, with a three-character word preceded by the ≠ sign, indicates which function of the equipment is wanted.

The example of a power supply equipment will be described in more detail. Independent of the use to which the power supply will be put, commands like ON, OFF, status reading, interlock settings etc.. will be needed. Also within the DATA-MODULE verifications are required to ensure that currents or voltages stay within maximum and minimum values, and comparisons of acquired values versus reference values are possible. The power supply should also have certain software properties such as the conversion factors for the setting and measuring of apparatus. To illustrate the above-mentioned concepts an example for a call to a DATA-MODULE with specific properties is given below.

```
SET A =  MAGNET(6, ≠CUR)
```

will put the measured value of the current that flows in magnet 6 into the variable A.

The statement

```
SET MAGNET(7, ≠MAX) = 10.16
```

will set the maximum current allowed in magnet 7 to 10.16.

The DATA-MODULES can actually be used as simple variables and their calls can be put in arithmetic expressions. The statement

```
TYPE MAGNET(6, ≠CUR)/MAGNET(6, ≠MAX)
```

will print out the relative value of the current flowing in magnet 6 as compared to its maximum current.

Apart from being very efficient for equipment access, the DATA-MODULE is also very practical from the implementation side since all information concerning one type of equipment is packed in a single module.

The DATA-MODULE is an obvious candidate for export to a micro-computer. It has a clean interface with the rest of the software system with a minimal data set to be shared and a maximum autonomy.

4. NORD and ACC software layout

The ACC contains NODAL compiled code to control the equipment associated with it. Six priority levels are available:

- front panel interrupt 1
- front panel interrupt 2
- front panel interrupt 3
- front panel interrupt 4
- DATA-MODULE request
- background program.

The code for all six levels is written in NODAL. The interrupt and background coding are declared by the commands WAIT-I and BACKG, which have been added. In the example below possible coding for five levels is shown.

```
1.1          WAIT-I 1 ;          SET INT = 1
1.2          WAIT-I 2 ;          SET INT = 2
1.3          WAIT-I 3 ;          SET INT = 3
1.4          WAIT-I 4 ;          SET INT = 4
1.5          BACKG ;             SET BC = BC + 1
```

Supposing that the front panel interrupt two is activated, the code immediately following WAIT-I 2 command will be executed. In this case the contents of the variable INT are set to two, and the interrupt level is left when the next WAIT-I command is encountered. Control then returns to the next highest level, which in general will be the background routine, here defined at line 1.5. In the example the background routine continually updates the counter BC.

The variables BC and INT should be situated in the system variable area if they are supposed to be known to the DATA-MODULE or the NORD computer. In that case they are loaded into memory with a preset value together with the NODAL compiled code.

The DATA-MODULE code, which is also written in NODAL, provides the link between NORD computer and ACC. In the example given below, the DATA-MODULE DMS will be able to return the value of INT or BC to a NODAL program running in the NORD. A NODAL program can also store a new value into BC.

```
DATA-MODULE DMS

10.10        % Reset Background counter
10.10        GET BC; END

20.01        % Return Background counter
20.10        PUT BC; END

30.01        % Return Interrupt counter
30.10        PUT INT; END
```

A table, which is not shown here, provides the automatic switching to line 10.10, 20.10 or 30.10 depending on the property of the DATA-MODULE call.

For example, the command in the NORD computer

```
TYPE  DMS (1, ≠ BC)
```

will send the request to execute the DATA-MODULE coding for property ≠BC. The equipment number does not apply in the example.

In the ACC the dispatch coding recognizes the command as an existing property. Based on the contents of the property table it will then execute the line 20.10. The PUT command returns the value of BC to the calling program and this value is consequently printed on the terminal where the command was typed. Accordingly the command:

SET DMS (1, ≠ BC) = 4

will result in the execution of line 10.10 where the GET command will store the value 4 into BC. Finally the command

TYPE DMS (1, ≠ INT)

will print 1, 2, 3 or 4 depending on the interrupt level activated last.

In the NORD computer, the resident interpreter decodes the NODAL commands which are then sent in the form of a table to the appropriate ACC. The communication package resident in the NORD will verify that no errors occurred during transmission. Errors occurring during the execution of the code in the ACC are retransmitted to the NODAL interpreter resident in the NORD in the same way as execution errors are transmitted by programs resident in the NORD computer.

5. Other facilities

The NODAL interpreter is available for the NORD as well as for the TMS-9900[10] computers. All arithmetic instructions are executed on three word floating point values. The NODAL cross compiler is available on the Nord-10 and Nord-100 machines running under the SINTRAN III run time system. Code generators are available for the N-10/N-100 and the TMS-9900 computers. The DATA-MODULES discussed in his paper can be constructed and tested interactively on the NORD as well as on the TMS-9000 machines running the NODAL interpreter. The execution of compiled programs is identical to the execution of programs running under the interpreter. In both cases checks are done on overflow, array bounds, consistent function calls, etc... .

The ACC is commercially available and forms the basis for more elaborated CAMAC computers which may contain either up to 8 k of EPROM, or access a floppy disk. For the latter a floppy disk filing scheme[11] compatible with the SINTRAN III filing system has been written. Files can be loaded and saved to/from the floppy disk when the NODAL interpreter is running; also a floppy disk bootstrap facility is available[12].

6. Conclusions

The DATA-MODULE concept together with the NODAL interpreter have enabled the exportation of control functions to a microprocessor[13]. A call from a NORD host computer to the code located in the microprocessor is exactly similar to a call to code resident in the NORD computer. The code for the microprocessor is written in the high level interpreter language NODAL and subsequently compiled into microprocessor object code. This scheme presents several advantages:

- The code can be constructed and tested interactively with the aid of the NODAL interpreter.

- The code can be easily tranported to other microprocessors by attaching other code generators to the NODAL compiler.

- The code resident in the microprocessor is easily accessed by simple NODAL commands from the host computer.

Acknowledgements

The NODAL compiler and NODAL interpreter for the TMS have been written by NITTEDATA. The interpreter is owned by NITTEDATA, the compiler is the property of CERN. The technical support of the ACC is assured by C. Guillaume. V. Frammery has written all hardware handling routines. E. d'Amico has contributed by being the first to use all the facilities for the solution of a specific instrumentation problem and by suggesting improvements. T. Stokka has contributed to the DATA-MODULE support routines and has written the code generator for the NORD. M. Krueger was involved with the floppy disk handling routines, and last but not least, J. Altaber and F. Beck have initiated and encouraged this project.

REFERENCES

1. Texas Instruments, 9900 family systems design.

2. C. Guillaume, Contrôleur auxiliaire de châssis CAMAC (ACC). SPS/ACC/CG/Note techn. 79-9 (1979).

3. M.C. Crowley-Milling, G. Shering. The NODAL system for the SPS. CERN 78-07 (1978).

4. M.C. Crowley-Milling. The data-module, the missing link in high level control languages. Presented at 3rd Int.Conf. on Trends in on-line computer systems. Sheffield, 27-29 March, 1979.

5. J. Altaber, F. Beck. The distributed data-base for the CERN SPS Control system. Presented at European Workshop on Industrial Computing Systems (PURDUE, Europe), Vienna, April 1980.

6. NITTEDATA, NODAL cross compiler. Reference manual.

7. NITTEDATA, NODAL code generator for TMS-9900.

8. NITTEDATA, NODAL code generator for M6800.

9. T. Stokka. Private communication.

10. NITTEDATA, TMS-9900 NODAL interpreter. Reference manual.

11. NITTEDATA, NODAL floppy file system. Reference manual.

12. M. Krueger. Private communication.

13. R. Bossart, A. Chapman-Hatchett, E. d'Amico, J.P. Papis, H. Rossi, V. Rossi, Beam position measurement for proton-antiproton operation in the SPS. Presented at the XI Int.Conf. on High Energy Accelerators, CERN, Geneva, 7-11 July, 1980.