SPS/ACC/Note/85-3

february 1985

# NODAL IN MODULA-2:
# major enhancements and modifications.

H.P. Christiansen, A. McKeeman, K. Lundberg, P.D.V. van der Stok

## 1 Introduction

For some time an effort has been going on to write the NODAL interpreter in the MODULA-2 language. The development has been done for several target machines at the same time to assure the portability of the final product. The main targets are IBM/PC, MC68000, VAX/VMS and LILITH (Diser) machines. The size of the interpreter does not allow to run it easily on machines with 16 bit addressing space. So for the moment no version is foreseen for TMS9900 or NORD-100.

By the beginning of 1985 the first installations on the MC68000 should be possible. Although the NODAL language itself has not been modified, some changes have been made to the facilities around NODAL. The main differences concern :

- Editor
- LIST, LOAD and SAVE facilities
- Structure of the interpreter
- Debugging enhancements

The object of this note is to describe the differences and to discuss possible future enhancements. This note does not constitute a complete description but an introduction to the new interpreter. More detailed information especially about the Editor will be the subject of a set of different notes.

## 2 NODAL Editor.

The most important aspect for former users of NODAL is the modification of the control characters,for the line editor. The present set of control characters have been copied from those used by PED on the NORD computers. Their functionality is now closer to the functionality one meets in other editors. ( ned,vi, EDIT, WYLBUR). The total line is edited, which means that the whole line is displayed while characters are added or deleted from the line. The arrows ( ←,→) are used to place the cursor inside the line. A forward delete and a backward delete control character is used for the deletion of characters inside the line. The control B character still displays the last line terminated with an error. The control P,CR sequence retypes the last edited line.

With the EDIT command one edits the MAIN program

```
>EDIT
```

or a defined function

```
>EDIT FUNC
```

Consequently it is possible to edit a defined function without opening it,formerly the OPEN command destroyed the main program and the function header. It is now perfectly feasible to first define all wanted functions with a DEF-S, DEF-C or DEF-F command, and to write them consecutively by invoking the editor.

To obtain a good feeling of the editor, one is strongly encouraged to

try PED on a NORD computer.

Enhancements made to the editor for NODAL with respect to PED are:
- automatic line numbering
- display of groups
- display of function header

A possible future enhancement is an automatic syntax check for all typed NODAL lines.

3 LIST, SAVE and LOAD facilities.

The facilities around the above commands are modified to take into account the evolution of the defined function inside the interpreter. The defined functions now form an integral part of MAIN NODAL program. This is possible as the defined functions are mostly used local to a program and are no longer general facilities as intended in the distant past. Physically they are no longer stored in a seperate area but they mix freely with the MAIN NODAL program.

The commands ZDEF, LDEF and SDEF have been suppressed. Instead of

| | | |
|---|---|---|
| >LDEF file1 file2 | one uses now: | >LOAD file1 file2 |

Instead of

| | | |
|---|---|---|
| >ZDEF | one should use: | >ERASE ALLD |

and instead of

| | | |
|---|---|---|
| >SDEF file | one types: | > SAVE file ALLD |

In general the following mnemonics are reserved words.

```
ALLV    : All Variables
ALLP    : All main Program
ALLR    : All Resident functions
ALLD    : All Defined functions
ALL     : ALLP + ALLV
```

These mnemonics can also be used in other commands. Consequently

```
LI ALLV   ::=  LISV
LI ALLR   ::=  LISR
LI ALLD   ::=  LISD
```

The above mentioned mnemonics are also recognised by the ERASE command with one exception :

```
> ER ALLR
```

will not erase all Resident functions.

Additional ease of handling is introduced as shown by the following examples :

```
>SAVE file ALLP ALLD A
```

will save all defined functions, the whole main program and the

variable A on file. The command

```
> RU file
```

will load all stored items and start the execution of the MAIN program. The command

```
>EXEC (COMP) 1 FUNC A

1.1 FUNC(A)
1.2 REMIT A
```

will send group 1 ,the defined function FUNC and the variable A to COMP for remote execution. The computer COMP should contain the new interpreter to load the function and execute group 1 correctly.

## 4 Free functions

The free functions type has been suppressed. This has some consequences for the parameters tolerated for functions which are of this type. Two major differences concern :

BIT : only single elements are tolerated as parameter either array elements both REAL as INTEGER, or simple variables.
SORT : The call is no longer SORT(arr,+) or SORT(arr,-) but SORT(arr,ASCEN) or SORT(arr,DESCEN).

A more complete list will follow later.

## 5 Internal Interpreter structure

The sequence in which NODAL programs are interpreted and executed has been modified. Instead of one "interpretation" phase, there are now two phases with an optional third one.

phase 1 : syntax verification and validation of a NODAL line. the line is encoded in a set of tokens.
phase 2 : Execution of the set of tokens.
phase 3 : compilation of a set of tokens into machine instructions and their execution.

The advantages of this strategy are threefold

- An encoded line stays encoded to accelerate the speed of execution.
- The syntax verifier can be used for a syntax directed editor.
- Compilers of the NODAL language for different targets are easier to write.

In addition it is possible to decode and display the line in a preferred format ( pretty printer ). Also it will be easier to implement automatic program conversion routines in the case of major NODAL syntax modifications.

In a first version the above mentioned phase 3 will not be implemented, and the way it will be invoked is not yet determined. It is linked with another strategy decision. For the moment the ASCII NODAL line is thrown away after encoding to save storage space. For editing or SAVE purposes the ASCII line will be reconstituted from the set of tokens. The example below will illustrate the consequence of

this action. the line :

WHILE0=0

will be reconstituted as

WH 0=0

Based on experience with the first distributed version the final encode,decode and compile strategy will be decided.

6 Debugging enhancements.

Much less work has been done on this subject for the moment. The internal interpreter structure was however designed such that most information about the variables and the program flow would be at the disposal of an eventual debugging facility. The information which will be available to the debugger will include

- display of a selection of all variables of all active defined functions and the main program.
- Exact sequence of DO commands and Defined Function calls.

It will be possible to set breakpoints in the programs ( unconditional for the moment) . When a breakpoint is met the program flow can be inspected and the variables at different levels of execution can be inspected or modified. It will not be possible to erase variables or redefine arrays at any of these levels.After such an inspection execution can be resumed . An additional useful facility may be the monitoring of variable values; it is however unclear under which boundary conditions this latter facility may eventually be implemented.

7 Installation program

In the first semester of 1985 the new interpreter will be installed on the MC68000 based HAMAC CPU of SPS LEP controls, which will form the initial base for the Process Control Assemblies. this installation will be done with the aid of the VAX/UNIX x-compiler facilities available on PRIAM VAX. Another version will be installed on the IBM/PC with the aid of the LOGITECH compiler. These first two preliminary releases will allow us to receive the first comments and to find the inevitable bugs. In the second semester of 1985 a first official version will be distributed on a possibly wider range of CPU's.