SOME SUGGESTIONS FOR AA APPLICATIONS PROGRAMS

D. Blechschmidt, S. van der Meer

1. Introduction

In order to obtain a minimum uniformity of AA NODAL applications programs, it is recommended to follow a few simple rules as outlined below. At the same time, some useful general purpose routines (FREE, EQU and LISTER) are described.

2. Touch-button tree

The machine will mainly be controlled through the touch-button screens. A nodal program may create touch-buttons, using the function LEGEND and read what button was pushed by BUTTON or BUTS. This is described in detail in Shering's note "The Touch Terminals for AA", CERN/PS/AA/Mem. 80-10.

If many buttons are written, it is recommended to follow the convention of doing this by the command

```
FOR I = 1, 17; $IF LEGS(I)<>; $SET LEGEND(I-1) = LEGS(I)
```

LEGS is a string array (saved together with the program; see section 4) whose first element contains the heading for the touch screen. For the rest, LEGS(I) contains the text for button I-1. The check with the $IF command is not strictly needed; however, with the present slow transmission to the touch terminal, it saves much time if many of the buttons are empty.

The action of a program will depend upon the button that is pushed. Often, the action will be something like switching on a piece of equipment or reading a status and it is performed directly by the same program that reads the buttons. Usually, after such an action the display is updated and the buttons are read again for the next action. It is recommended to program this by the sequence

```
SET Z = BUTTON;  DO Z + 10
```

where the group Z + 10 will execute the relevant action. Group 27 will then be executed if the back button (No. 17) was pushed; it should

contain

```
27.1 RUN BACK
```

In future, this will hopefully be replaced by the function BACK.

Often, if a button is pushed, the program corresponding to the next tree level (i.e. further from the trunk) should be run. In future, this will be done by calling the function

```
NEXT ("filename")
```

However, this function does not exist yet and at present one should use

```
$SET FI = "filename";  DO 95
```

where group 95 contains the following:

```
95.1  $SET TR = "TREE"%2 ODEV + 9
95.2  LOAD $TR; SE N.FY = N.FY+1;  $SE FYLE(N.FY) = FI
95.3  SAVE $TR N.FY FYLE; RUN $FI
```

(In future, if NEXT is used, group 95 may be suppressed).
Line 95.1 defines the file where the program stack is kept, which must be different for each touch terminal, to avoid interference. This file contains the variables N.FY and FYLE.

Note that this sequence (or, in the future, NEXT) will start the next program at its first line, whereas BACK will start the preceding tree program at the lowest line of group 2. Thus, group 1 may be used for actions that are not needed if a program is entered while going back towards the trunk. Such actions could be the writing of fixed components of a display, or the loading of defined functions.

If many different programs may be called, depending on the button that was pushed, the following convention is recommended:

```
2.7  SET Z = BUTTON;  IF Z = 17;  RUN BACK
2.8  $SET FI = PROGS(Z+1); DO 95
```

(or, NEXT(PROGS(Z+1)), when this function will be available). The string array PROGS must again be saved with the program. PROGS(Z+1) contains the filename corresponding to button number Z.

All functions concerning the touch buttons and the main display screen are contained in the file (RT)CONSOLE that is loaded into the defined function area by the trunk program. If any other defined functions are needed by a branch program, they must be loaded together with (RT)CONSOLE. For instance,

```
LDEF CONSOLE FILE1 FILE2
```

will load FILE1 and FILE2 in addition.

Note that the prefix (RT) before filenames BACK, TREE and CONSOLE may be left out in programs that are executed under RT-NODAL. This saves 50 ms per filing operation.

3. Line 1.01

It is recommended to follow the convention of starting each program with a comment line that gives a very short description of a program together with the corresponding filename between square brackets and the abbreviated author name between round brackets. For instance,

```
1.01 %STACK TAIL BETATRON COOLING - [(VDM)CCT3] - (VDM)
```

4. Group 99

It is also recommended to end each program with group 99 as follows:

```
99.98  $SE DT = DATE
99.99  SAVE filename ALLP DT
```

Whenever the program is modified, it may be saved by typing DO 99. This avoids errors in the filename and includes the date of modification in the file. The latter will be listed with the file if the listing program LISTER (see section 5) is used.

Any further fixed data may of course be saved at the same time with the program, e.g.

```
99.99  SA(VDM)COOLING DT LEGS PROGS
```

and the advantage of using group 99 is that this is now automatic and no mistakes are possible.

Note that the program, if executed normally, should never enter group 99. This will usually be impossible, since the higher-numbered groups are normally used as DO subroutines and the main program sequence should be terminated by END or by RUN BACK. However, if one wants to be quite sure that group 99 cannot normally be executed, it is possible to insert

```
98.1  END
```

5. Program LISTER

This general purpose program will list all contents of a NODAL file. It is called (at present from RT-NODAL only) by

```
RUN LISTER
```

and will respond by

```
( ) FILENAME:
```

The requested filename (with its user name prefix) should then be typed. This file will be listed both on the terminal screen and on the attached printer. The printer must of course be switched on, but need not be "on line". The listing will consist of

a) present date, time and filename
b) NODAL text if any (LIST output)
c) LISV output (including the contents of simple numerical variables)
d) a list of simple string variables saved in the file, with their contents (if any)
e) for each array saved in the file, a table giving the contents of each element
f) a listing of each defined function saved in the file.

After the listing is finished, the program asks for the next filename. If no more files should be listed, the RETURN key should be pressed.

6. Function EQU

This defined function is included in the file (RT)CONSOLE. It may be used to call equipment modules. Its use will remove some routine operations from the calling program, which then becomes more easily readable.

The call

```
SET A = EQU("EM", N, "PR")
```

will read the equipment module with name EM, using equipment number N and property PR. The result will be in A.

If the equipment module returns a completion code that is not zero, function EQU will write the relevant error message (in inverted mode) on the bottom line (No. 24) of the main display, followed by the

relevant EM name, equipment number and property*). The program then stops with the error message HARDWARE ERROR, even if the error belonged to the "non-fatal" class.

This program stop may be prevented by placing the EQU call inside a DO group and using the DO ..!.. facility. In this way, recovery action may often be combined for many different equipment module calls.

Alternatively, the function EQU may be called with a negative equipment number N. This will have exactly the same effect (the absolute value of N being used); in case of an error the relevant message will be printed on the bottom line of the display, EQU will return the value - 1 and the program will not stop. It may check on the occurrence of an error by looking at C.CODE. This is a variable in the defined function area (also loaded with (RT)CONSOLE) that will contain the completion code returned by the equipment module. This way of using EQU sometimes makes the calling programme more readable, because the call need not be placed in a DO group.

EQU may also be used in the setting mode, e.g.

```
SET EQU("EM", N, "PR") = A
```

The setting will be preceded by reservation and followed by release. The error handling is exactly the same as for reading.

Using EQU will make an equipment module call slower by about 50 ms. This is of no importance if only a few calls at a time are made. If, however, many calls are made in sequence, it may be preferable not to pass through EQU.

7. Function FREE

This defined function is also included in (RT)CONSOLE. The command

```
FREE("EM", N1, N2)
```

will cancel any existing reservation (in particular from other terminals), and suppress any simulation status for equipment numbers N1 up to and

---

*) This line should not be used for anything else, except other error messages.

including N2 of the equipment module with name EM.

It is recommended to use this facility with caution, especially during the next few months, to avoid interference with equipment tests.