NAKED MINI4° SYMPLE

REFERENCE MANUAL

71000-90

JULY 1980



ComputerAutomation®

This product has been developed by Computer Automation in Europe

and all support enquiries should be referred to:

NAKED MINI® Division

HERTFORD HOUSE
DENHAM WAY, MAPLE CROSS
RICKMANSWORTH, HERTFORDSHIRE WD3 2XD

Telephone Rickmansworth 71211 Telex 922654

Engineering Reference Number
90-71000-00D1

REVISION HISTORY

Revision	Issue Date	Comments
A0 to B1	-	No History
B2	Nov 1979	First Production Release
C1	Jan 1980	NM4/95 instruction mnemonics added; SCOUT SAL mnemonic added; Mini-Flexible Disk driver added.
D1	July 1980	N and P commands added.

PREFACE

This Reference Manual describes the capabilities and use of SYMPLE, a symbolically-oriented program entry and debugging system which, apart from its symbolic features, resembles the NM4 Debugging Monitor DEBUG4 in many respects.

Its major advantage over DEBUG4 is the ability to inspect and list memory locations in instruction mnemonic formats and to enter modifications or new code in the same symbolic manner.

Despite the many similarities with DEBUG4, the SYMPLE Reference Manual is written as a complete stand-alone document.

CONTENTS

SECTION 1 INTRODUCTION 1-1

SECTION 2 HARDWARE REQUIREMENTS 2-1

SECTION 3 SOFTWARE INTERFACE 3-1

3.1 SUBROUTINE RORL 3-1

3.2 USER TRAP INSTRUCTION 3-3

3.3 MEMORY LOCATION ZERO 3-3

3.4 GENERAL REGISTERS G0 TO G9 AND GA TO GZ . . . 3-4

3.5 EXTERNAL NAMES 3-4

3.6 PATCH AREA 3-4

SECTION 4 OPERATOR COMMUNICATION 4-1

4.1 GENERAL 4-1

4.2 INITIAL ENTRY TO SYMPLE 4-2

4.3 PROGRAM CONTROLLED RETURN TO SYMPLE 4-3
4.3.1 General 4-3
4.3.2 True Breakpoint (B) 4-3
4.3.3 Trapped Interrupt (I) 4-3
4.3.4 Bus Monitor Interrupt (M) 4-4
4.3.5 Hardware Trap (T) 4-4
4.3.6 Dummy Breakpoint (D) 4-4

4.4 OPERATOR CONTROLLED REENTRY TO SYMPLE 4-5

4.5 ENTRY OR REENTRY TO USER PROGRAM 4-6

4.6 OUTPUT OF VALUES AND ADDRESSES 4-8

4.7 SYMBOLIC OUTPUT OF INSTRUCTIONS 4-9

4.8 INPUT OF COMMANDS - GENERAL 4-10

4.9 INPUT OF VALUES AND ADDRESSES 4-11

4.10 INPUT OF ADDRESS RANGES 4-14

4.11 SYMBOLIC INPUT OF INSTRUCTIONS 4-15

4.12 STOPPING OUTPUT OPERATIONS 4-18

SECTION 5 SYMPLE COMMANDS 5-1

APPENDICES

APPENDIX A COMMAND SUMMARY A-1

APPENDIX B LINKING, LOADING AND CALLING SYMPLE B-1

B.1 LINKING SYMPLE B-1
B.1.1 Files Supplied to Users B-1
B.1.2 Linking SYMPLE Only B-1
B.1.3 Linking SYMPLE to User Program B-1

B.2 LOADING SYMPLE B-2

B.3 CALLING SYMPLE B-2

APPENDIX C SUMMARY OF RECOGNISED INSTRUCTION MNEMONICS . . . C-1

SECTION 1

INTRODUCTION

The SYMPLE system provides for the on-line checkout and testing of user software on the Computer Automation Naked Mini 4 series of computers. An important feature of SYMPLE is the ability to inspect and modify memory words using symbolic instructions which are very similar in form to those of the MACRO assembler. SYMPLE can take advantage of the Bus Monitor if it is installed, but will operate usefully without this feature. SYMPLE may be link loaded together with the program being debugged, or it may be loaded into a different area of memory. Control of the machine passes back and forth between SYMPLE and the user program as determined by commands from the operator. Whenever SYMPLE has control the operator may issue commands to inspect, list, modify, search or execute any location in memory. Commands are available also to dump, load and verify programs on paper tape, floppy disk, mini-flexible disk or magnetic tape.

SYMPLE saves all of the user program's hardware registers each time it regains control, and restores them when it is directed to return control to the user program. The operator can use SYMPLE to examine and modify the stored register contents, hence it is very rarely necessary to have access to the computer's control panel once SYMPLE is available in memory.

When SYMPLE is loaded independently, usually it should be run before entering the program to be debugged. Entry at its first word (which has the externally accessible names SYMPLE, DEBUG and DEBUG4) stores the present value of the user's status register (S) and sets up memory locations :0, :86, :8A, :94, :96, :9A, :9E, :A2, :A3 and :AA for use by SYMPLE and to provide a link which may be used subsequently by the user program or by the operator to return control to SYMPLE. SYMPLE is now in control of the machine, and the user program can be entered by means of a J, B or X command.

When SYMPLE has been link loaded as part of the user program, that program should make an early call to the SYMPLE routine called RORL. The effect of calling the subroutine RORL is similar to entry at the first word of SYMPLE, except that the return link is stored in the user's P-register location in SYMPLE so that use of command RUN, J, B or X can return control to the next instruction in the user program. Also, RORL optionally outputs a program title and revision number on the operator terminal. The title is supplied by the user program, but the revision number is held within SYMPLE and is updated each time a dump is made.

If the user program is entered (or reentered) by any of the commands RUN, J or B, SYMPLE retains control of the monitoring and reporting

of hardware trap conditions (except for power fail and restart). Also it arranges for console interrupt to reenter SYMPLE, and it sets up the vector for the UTRAP instruction and memory location 0 so that any use of UTRAP or JST *0 or JMP *0 in the user program will cause a breakpoint return to SYMPLE, which will output the contents of all the user program registers, including the P-register showing the address at which the trap occurred. Hence it is suggested that where a program would halt on an error, it is advisable to code a UTRAP instruction instead of a HLT, and to code HLT in memory location :A2 and JMP $-1 in location :A3 (in the user trap vector). This means that when the program is run under control of SYMPLE any such error is immediately reported on the operator terminal, and when run without SYMPLE locations :A0 and :A1 will provide useful information about the error. If the user program requires to take its own action on any of the hardware traps or console interrupt, it should set up the vector locations concerned after return from its call to RORL.

If SYMPLE is to relinquish control of the computer to the user program or operating system, command X should be used instead of RUN, J or B. This command restores all the low memory locations that were saved. However any which were subsequently modified deliberately by the user program or by the operator are restored to their most recently set values.

If the hardware Bus Monitor is fitted, it can supply the addresses of the most recent 80 memory accesses (together with indication of read, write, DMA, etc). SYMPLE includes a facility to list the contents of this buffer, which can give very useful information after a program failure or trap. In addition, SYMPLE can be used to instruct the Bus Monitor to interrupt the processor if certain selected memory accesses occur, thus for example it is possible to trap immediately an unexpected access is made to some part of memory.

SECTION 2

HARDWARE REQUIREMENTS

SYMPLE will operate on any Naked Mini 4 series computer.

SYMPLE occupies just over 4K words of core or RAM memory. It may be loaded anywhere in memory, but if the binary dump and load facilities are used, the last word of SYMPLE must be at least 512 words (:200) below the last word of core or RAM memory. This requirement is reduced to 256 words (:100) if only paper tape dumping and loading are required.

An Operator Terminal must be available, driven through a standard serial pico-processor. Ideally, the terminal should be a CRT with at least 20 lines and 80 characters per line. A smaller CRT or a TTY may be used, but with a consequent reduction in the effectiveness of some facilities. There are no facilities to use the paper tape reader or punch on a TTY. The standard device address for the terminal is :F8. The device address can be modified by a register setting on initial entry to SYMPLE, or it can be altered by operator command to transfer control to a different terminal.

A Line Printer (driven through a standard line printer pico-processor) or a Printer driven through a standard serial pico-processor can be used to log all output and input at the operator terminal, and to list large areas of memory independent of the terminal. There are commands to select which printer, if any, is to be used. SYMPLE does not use both printers simultaneously.

The standard device address for a line printer is :FE. If there is a printer it should have, preferably, at least 120 columns per line and be able to print at least 60 lines per page. The default assumption is that the printer requires separate carriage-return and line-feed characters, and that it accepts and acts upon form-feed characters. There are operator commands to vary the device address and to indicate that line-feed and/or form-feed characters are not to be used. If form-feeds are not used SYMPLE treats the line printer as unpaged, and takes no notice of end of page conditions. If the printer has less than 60 lines per page it is advisable to instruct SYMPLE to treat it as unpaged. If the printer has less than 120 columns SYMPLE can be instructed to print less items per line than usual. In commands, the line printer is sometimes referred to as P1 (indicating Printer 1) or L (for Line printer).

There is no standard device address for a serial printer. If a serial printer is in use its device address must be specified by an operator command or by a register setting on initial entry to SYMPLE. A serial printer should have at least 80 columns per line. It must accept both carriage-return and line-feed characters, and must be capable of operating without form-feed characters. In commands, the serial printer is sometimes referred to as P2 (indicating Printer 2) or S (for Serial printer).

Paper tape dumps may be made through a High Speed Punch driven by a standard punch pico-processor with device address :FC. There is no facility to vary this address.

Paper tape dumps may be read/verified through a High Speed Reader driven by a standard reader pico-processor with device address :FA. There is no facility to vary this address.

Floppy Disk unit 0 may be used to dump, load or verify programs. The controller must have the standard device address (:88).

Magnetic Tape drive 0 may be used to dump, load or verify programs. It must be on a PERTEC compatible formatter using a standard magnetic tape pico-processor with device address :F2 and interrupt vector address :C8. Note that the only time that SYMPLE permits interrupts while it is running is during magnetic tape operations.

Mini-Flexible Disk unit 0 may be used to dump, load or verify programs. The disk system must have the standard device address :FC.

If a Bus Monitor is present it must have the standard device address (:B0) and vector address (:96). All delay counter switches should be off. If there is no Bus Monitor, there must be no other device installed with these addresses.

SECTION 3

SOFTWARE INTERFACE

3.1 SUBROUTINE RORL

The subroutine RORL (which stands for RUN OR LOAD) is included in SYMPLE and the user program obeys a JST RORL to make an initial entry to SYMPLE. RORL must be be declared in an EXTR statement in the user program, and SYMPLE must be link loaded with the user program. When SYMPLE is entered in this way it saves the user's S-register, and the return link from RORL as the user's P-register. Other registers are not saved at this time.

The entry to RORL requires two parameters to be supplied in the A and Y registers. If the A-register is non-zero it specifies the device address of the operator terminal and , optionally, the device address of a serial printer. The right hand byte contains the device address of the operator terminal, and must be given if a non-standard address (i.e. not :F8) is used, and/or if a serial printer address is specified. The left hand byte, if non-zero, contains the device address of a serial printer (i.e. a printer driven by a serial pico-processor). If an address is given here, SYMPLE stores it as the device address for printer 2. Also it selects printer 2 for use, enters the mode in which all terminal input/output is logged on the printer, and sets the default line width to 80 characters. All these settings can subsequently be changed by operator commands. If the left hand byte of the A-register is zero, no change is made to any of these settings; and if the whole of the A-register is zero the terminal device address is left unaltered.

If the Y-register is non-zero it contains the address of the first word of a program title message stored two ASCII characters per word and terminated by a zero word. The message is output to the operator terminal, followed by the revision number for the program. The revision number is initially set to zero, and it is incremented each time a dump is made. Thus it can be used to determine which version of a program is in use, if it has been patched and dumped since the original issue. If the Y-register is zero, no title or revision number is output.

The normal operator prompt is now output. This is a greater than sign (>). The operator is free to use any of the facilities of SYMPLE, but normally he would use RUN (or optionally carriage-return only) if the program is to be run, or LOAD if a different program is to be loaded.

Before returning to the user program, SYMPLE sets up the following memory words:-

:0 so that any use of JST *0 or JMP *0 will cause a breakpoint entry to SYMPLE.

:86 so that any Unimplemented Instruction trap will cause a breakpoint entry to SYMPLE.

:8A so that any Uninstalled Memory trap will cause a breakpoint entry to SYMPLE.

:94 so that operation of the Console Interrupt switch will cause a breakpoint entry to SYMPLE, provided the user program has enabled the console interrupt and general interrupts.

:96 so that any interrupt from the Bus Monitor will cause a breakpoint entry to SYMPLE.

:9A so that any Character/Numeric Exception trap will cause a breakpoint entry to SYMPLE.

:9E so that any Stack Overflow trap will cause a breakpoint entry to SYMPLE.

:A2/:A3 so that any UTRAP instruction will cause a breakpoint entry to SYMPLE.

:AA so that any Arithmetic Error trap will cause a breakpoint entry to SYMPLE.

If the user program includes its own routines to deal with any of the above conditions, it can overwrite the locations concerned after return from RORL. However in general it is wise to leave them as set by SYMPLE as this will provide useful information in the event of any error. As advised earlier, the user program may contain UTRAP instructions instead of error halts, since they will cause SYMPLE to output the trap address and register contents if such an instruction is obeyed.

SYMPLE does not automatically overwrite the power fail and power up restart locations. A useful simple power down/power up sequence is to place a HLT instruction in location :8E and a UTRAP in location :80. This will halt the processor on power fail, and reenter SYMPLE on power up.

3.2 USER TRAP INSTRUCTION

As already indicated, the User Trap instruction (UTRAP) is normally reserved for use as a breakpoint entry to SYMPLE. When the user program is entered from SYMPLE by means of a J or B command, the vector for the user trap instruction is set so that any execution of the instruction causes an immediate return to SYMPLE. The user may code UTRAP instructions into the program, or plant them there with the I command in SYMPLE.

When the user program is entered by command J or B from SYMPLE, there is an option to set up breakpoints in any memory locations. SYMPLE plants a UTRAP A,0 instruction in each such breakpoint and preserves the original contents of the location. On return to SYMPLE for any reason, all such breakpoints which still contain the UTRAP instruction are restored to their original contents. Any breakpoints whose contents have been altered by the user program itself are not restored but retain their new contents.

3.3 MEMORY LOCATION ZERO

Whenever the user program is entered by command J or B from SYMPLE, location zero is set to contain a pointer to a subroutine which simulates the action of a UTRAP instruction. Thus if the program obeys JST *0 the effect will be virtually identical to that given by a UTRAP instruction. This facility is provided for two purposes:-

(i) it allows the continued use of programs which previously used JST *0 to force entry to DEBUG4,
(ii) it provides a method of operator controlled reentry to SYMPLE which is described in more detail later.

3.4 GENERAL REGISTERS G0 TO G9 AND GA TO GZ

SYMPLE contains 36 workspace locations which are referred to as general registers. They are called G0 to G9 and GA to GZ. They may be used in various ways, but their main purpose is to provide base addresses for use when examining and modifying relocatable program modules. Thus for example, GQ might be set to hold the address of relative word 0 of a program module, and relative word :123 of that module could then be addressed as 123GQ (or GQ+123).

The G registers can be set up by command when SYMPLE is running. However when SYMPLE is link loaded together with the user program, the G registers may be preset to point to particular places in the program by use of NAM declarations. For example to preset GQ to point to the first word of a module, that module should include NAM G:Q at its head, and G:Q EQU $ at the point whose address is to be stored in GQ. Note the use of the colon between the letters. This is compulsory as within SYMPLE each G location is initialised with a sequence like:-

```
     SEXT G:Q
     WORD G:Q
```

Any G register not initialised within a module of the user program is set by the OS4 linker to contain :0000, and its name will be listed as MISSING if the linker is directed to list unresolved secondary externals.

3.5 EXTERNAL NAMES

SYMPLE can be force loaded from a library containing its object by calling for any of the following primary externals which are declared in SYMPLE:-

```
     SYMPLE
     DEBUG
     DEBUG4
     RORL
```

3.6 PATCH AREA

SYMPLE contains a patching area of 64 words (hex 40 words). This may be used by the operator for storing program patches, etc. Its words may be accessed in SYMPLE commands by use of the letter P together with an offset. The letter P addresses the first word of the area. Offsets to P must not be greater than 3F.

SECTION 4

OPERATOR COMMUNICATION

4.1 GENERAL

When entered at its origin address, SYMPLE outputs its name and version number, followed by a revision number. The version number is coded into SYMPLE and only changes when a new version of SYMPLE is issued. The revision number is updated each time a dump is made, so it relates more to the total memory load of programs than to SYMPLE alone.

If SYMPLE is entered by JST RORL, a user supplied program title is output instead of SYMPLE's name and version number. The revision number is still output, and subsequent action is the same in both cases.

SYMPLE generates a prompt to the operator, consisting of a bell character followed by a greater than sign (>) in the second position of a new line. The first position of the line is left blank, but in two cases (described later) it is subsequently overwritten. On completion or rejection of each command this prompt is repeated.

A command line consists of a unique command identifier of one or more letters, followed by parameters as required by the command. The first parameter follows the command immediately with no separator between them. If there are more parameters than one, each is separated from the previous one by a comma. The command line is terminated by a carriage-return character. Normally, a space character is not allowed within a command line, except when it is used as an ASCII text character or between the op-code and first operand of a symbolic instruction. There are exceptions to these general rules and these are explained in the descriptions of the commands concerned. In particular some commands send output to the operator terminal without the input line being completed, and in some cases a carriage-return or space by itself is allowed instead of a command.

SYMPLE rejects any incorrect command or unacceptable parameter by outputting a question mark (?) in the first position of the current line (i.e. to the left of the original > character). It then outputs a new prompt on the next line. The operator can deliberately cancel a line by typing an unacceptable character such as back-arrow, backspace or underline. A command is rejected as soon as SYMPLE determines that the input is invalid. In some cases this happens before a carriage-return has been input, and in other cases only after carriage-return has been input. Thus in all cases the operator should check whether a command has been rejected by output of a question mark before continuing with the next command. In some cases

a command which alters operating parameters (such as device addresses or items per line) may have been partially executed prior to rejection.

If any command alters memory contents, SYMPLE outputs an asterisk (*) in the first position of the current line to mark the change, before giving a prompt on the next line. This results in a record of command lines which modified the user program. If the modified program is dumped for future use without being reasembled, the SYMPLE listing is the only documentation of the new revision of the program. It is therefore highly desirable that a hard copy of the SYMPLE output should be made, using the facility in SYMPLE to log the operator terminal onto a printer.

In general if a SYMPLE command alters memory contents, it checks that the correct data is written to memory. If the data is not written correctly, an error warning is given in the form of a percent sign (%) in the second column of a new line. For multiple word operations the percent sign is followed by the address of the offending word.

Also, the percent sign warning is given when certain other errors are encountered during command execution, such as an attempt to access non-existent memory. Finally, it is used to indicate that output was abandoned at the request of the operator during a multi-line output command.

4.2 INITIAL ENTRY TO SYMPLE

The two most common methods of initial entry to SYMPLE have already been described. They are by subroutine call JST RORL (see subsection 3.1), and by direct entry to the first word (see subsection 4.1). Normally the latter method would be used to enter from an operating system such as OS4. An alternative entry may be made directly to the second word of SYMPLE if a non-standard device address is to be used for the operator terminal, or if a serial printer is to be brought into use at once. This entry requires a parameter in the Q-register. Its contents and effect are exactly as described for the A-register on entry to subroutine RORL (see subsection 3.1). Apart from selecting devices, entry at the second word has the same effect as entry at the first word.

4.3 PROGRAM CONTROLLED REENTRY TO SYMPLE

4.3.1 General

Apart from a new call of RORL, the program under test will reenter SYMPLE whenever it obeys a UTRAP or JST *0 instruction. Usually, this will occur at a breakpoint planted in the program by a B or J command in SYMPLE, or as the result of an interrupt from the Bus Monitor under the control of the M command in SYMPLE. Alternatively, the trap could be deliberately coded into the program or stored in memory, or it could occur via one of the hardware error traps or console interrupt.

On a trap entry to SYMPLE, all the user's registers are stored and displayed on the operator terminal. The form of data output at the terminal depends upon the cause of the break entry, as described below.

4.3.2 True Breakpoint (B)

If the trap occurs at a breakpoint which was planted by a B or J command, it is called a true breakpoint, and the output takes the form:-

```
B P=pppp A=aaaa Q=qqqq X=xxxx Y=yyyy
  k=kkkk L=llll S=ssss D=dddd C=cccc
```

Each register is shown as a four digit hexadecimal number, but, additionally, the P register value may be given relative to a G-register (see subsection 4.6).
D denotes the console data register, and C denotes the console sense switch for which only the least significant bit is valid.

4.3.3 Trapped Interrupt (I)

If the trap is as the result of an instruction obeyed in response to an interrupt, the letter B is omitted, and the register display is preceded by a line of the form:-

```
I iiii
```

where iiii is the address of the interrupt vector. The stored and displayed P-register value is that of the instruction which was about to be obeyed when the interrupt occurred. Note however that if the interrupt instruction is not itself a UTRAP but it traps for some other reason (such as memory exception) the vector address given may be incorrect. If it appears to be in another trap vector area, that

trap vector should be examined to find the actual vector address.

4.3.4 Bus Monitor Interrupt (M)

If the trap occurred as the result of an interrupt from the Bus Monitor, the normal breakpoint output is given, but it starts with the letter M instead of B. Also the register output is followed by a list of the most recent sixteen memory accesses recorded by the Bus Monitor. The list is in four columns of four entries each. An entry contains the memory address accessed, in G-relative and absolute forms, followed by one or more letters showing the status of the memory access. The letters used and their meanings are:-

R - read access to memory
W - write access to memory
D - DMA access to memory (R or W will be present also)
I - interrupt request present on the bus (R or W and possibly D will be present also).

It is possible that less than 16 memory accesses will be listed, as only those since the last return to the user program by a B, J or X command are given.

4.3.5 Hardware Trap (T)

If the entry to SYMPLE is not through an interrupt, but it is by a UTRAP instruction obeyed at any address between :80 and :AB, it is probably the result of a UTRAP instruction being obeyed in some other trap vector. This will always happen on a hardware trap if its vector is still as set within SYMPLE. In this case output is similar to that for a trapped interrupt (see topic 4.3.3) but the first line of the output does not start with I. Instead it starts with T and contains the address of the first word of the apparent trap vector involved, and the contents of the first and second words of that vector, e.g:-

```
T 0084 1234 7600
```

would indicate trap vector :84 (unimplemented instruction). The trap was the result of attempting to obey the instruction :7600 at memory address :1233. (Note that P was incremented before the trap occurred, so the first word of the trap vector holds :1234).

4.3.6 Dummy Breakpoint (D)

If the trap is not an interrupt or within the trap vector area, and it is not at a location where a breakpoint was planted by a B or J command, it is called a dummy breakpoint. The output is similar to

that for a hardware trap except that the first line has the form:-

```
D 1B03
```

The D indicates a dummy breakpoint, and the hex number is the actual UTRAP instruction obeyed in the user program. Hence the program can use the variable parts of the UTRAP instruction to show different stopping reasons, and the dummy breakpoint output can be used to identify the particular error stop in the program. If the dummy break was by JST *0, the value shown after D is 0000.

4.4 OPERATOR CONTROLLED REENTRY TO SYMPLE

If the program under test never reaches a UTRAP or JST *0 instruction, it will not return control to SYMPLE. The following three methods are available to the operator to force reentry to SYMPLE, but only the last can be used if SYMPLE was previously exited by an X command:-

(i) Press the Console Interrupt button. This will cause a type I trap to SYMPLE provided the program has general interrupts and console interrupt enabled. The interrupt address will be shown as 0094.

(ii) If there is a normal or remote programmer's console available, the following sequence will return control to SYMPLE in a useful way, provided that location 0 has not been altered by the program:-
a) Halt the processor.
b) Display the P-register and preserve its value (possibly in the console data register)
c) Display the contents of memory word 0, and copy to P-register.
d) Restart the processor.
This will give a dummy breakpoint entry to SYMPLE with all registers except for P correctly saved and displayed. The RP command can be used to set the stored P-register to the value saved in step (b).

(iii) As a last resort, the operator can set the P-register to the origin address of SYMPLE. In this case only the S-register is saved correctly, so it is advisable to note the contents of all the registers first and then to restore them using R commands. The origin address can be found from a load map if the program under test was link loaded together with SYMPLE. Also, when SYMPLE is running, its low and high addresses can be found by means of command Y.

4.5 ENTRY OR REENTRY TO USER PROGRAM

Control can be passed to the user program by any of the commands RUN, J, B or X. The registers are loaded with the values held for them within SYMPLE. They may, of course, have been modified by commands while SYMPLE was running. Optionally, the J and X commands may specify a new P-value, but the RUN and B commands may not do so. The main features of the different commands are as follows:-

RUN (or J with no parameters) loads all the registers with their saved values, clears all breakpoints set in a previous B or J command, and leaves the hardware traps, UTRAP, etc, set to return to SYMPLE on any error. RUN also enables the Bus Monitor to collect data and to break into the program in accordance with the settings of M0, M1, M2 and M3.

J with a single parameter is exactly like RUN, except that the parameter gives a new P-register value.

J with more than one parameter uses the first parameter to give a new P-register value, and the remaining parameters as up to 4 addresses for normal breakpoints to be planted. These replace all breakpoints specified in any previous B or J command. Otherwise action is similar to RUN.

B with no breakpoint parameters is similar to RUN, except that it retains any breakpoints specified in the previous B or J command. It is only acceptable if at least two breakpoints exist.

B with one or more breakpoint parameters is similar to RUN also, except that it uses the newly specified breakpoints instead of any previously existing ones.

X with or without a parameter is exactly like J being used in the same way, except that all the hardware trap vectors, UTRAP vector, etc, are restored to their saved values, and the Bus Monitor is not allowed to break into the program (although it is allowed to collect data). This is the normal means of returning control to an operating system such as OS4 which might overwrite SYMPLE. Also, it is used if SYMPLE is required just to make some initial patches to a program or if the user program requires to make private use of the trap vectors, etc.

When breakpoints are being set or retained on return to the user program, UTRAP instructions are planted in the breakpoint locations (after saving the current contents). However if there is to be a breakpoint at the word pointed to by the user's P-register, that word

is not overwritten by a UTRAP, thereby allowing the instruction to be obeyed correctly. It does mean, however, that there must be at least one other breakpoint if the program is to return control to SYMPLE automatically; a common technique is to plant breakpoints at two instructions which are known to be obeyed consecutively.

If the RUN, J, B or X command is given without P being altered from what was saved on last entry to SYMPLE, the exact effect depends upon the method by which SYMPLE was entered, as follows:-

(i) If the entry was at a true breakpoint (type B), the program continues, obeying first of all the instruction located at the point where the break occurred.

(ii) If the entry was by a trapped interrupt (type I), or a Bus Monitor interrupt (type M), the program continues from the instruction which it was about to obey when the interrupt occurred.

(iii) If the entry was through a hardware trap (type T), or a dummy breakpoint (type D), the program will immediately obey the same instruction again. Thus normally it will break again immediately with the same output at the terminal. This prevents the operator from allowing the program to continue past an intentional error stop. If command X is used following a type T or D entry to SYMPLE, all the trap locations will have been restored to the values set within the user program, and in that case the program will enter whatever code it contains for dealing with the particular trap condition.

(iv) If SYMPLE was entered at either of its first two words, no legitimate return address is known, so P is set to point to a UTRAP instruction within SYMPLE. Thus normally a type D break will occur at once, but if command X is used the user program's UTRAP routine will be entered.

(v) If SYMPLE was entered by JST RORL the stored P-value is the return address to the next instruction in the user program. Thus the user program will continue normally.

NOTE There is a special case in connection with returning from a call of RORL. If the operator inputs just carriage-return in reply to the first prompt, this is treated the same as RUN, and causes immediate return to the user program. Once any other SYMPLE command has been accepted this facility is disabled, and the normal RUN or J command must be used subsequently.

4.6 OUTPUT OF VALUES AND ADDRESSES

All values output by SYMPLE are in hexadecimal. Usually a complete word is given as four hexadecimal digits, including leading zeros where necessary. However when the value is an operand in a symbolic instruction leading zeros are omitted, except that the value A is given as 0A in order to distinguish it from register A.

When a value is likely to be an address, SYMPLE examines the 36 G-registers described in subsection 3.4. It selects the G-register whose contents are nearest to but not greater than the desired value. Then, if the G-register does not contain zero and the value in question is not more than hex FFF greater than the G-register contents, the value is output in the form

abcGg

(where abc is the hexadecimal difference between the actual value and the G-register contents, and g is a single digit or letter showing which G-register is involved). When the value is an operand in a symbolic instruction this G-relative form is given by itself; but when given as the current address or as a full 16-bit value it is followed by an = sign and the full hexadecimal value. When no valid G-relative interpretation exists for a value, it is output in hexadecimal only.
Note that when G-register values are being compared, they are treated as unsigned quantities in the decimal range 0 to 65535. Addresses are, of course, in the same range.

4.7 SYMBOLIC OUTPUT OF INSTRUCTIONS

When instructions are output in symbolic form by SYMPLE, they look very similar to true MACRO assembler instructions except that all numbers are in hexadecimal but without the leading colon that is required in true assembler programs. The op-code is output first, separated by a space from the operands. Operands are separated from each other by commas. The following additional general remarks apply to symbolic output of instructions:-

(i) If a word does not contain a valid instruction according to the decode table in the version of SYMPLE being used, the symbolic output is given as ????.

(ii) If an apparently valid instruction has an invalid combination of operand bits, question marks can appear in the operand fields. An example would be a left shift instruction which included the "arithmetic" marker.

(iii) For double indexed memory reference instructions, the indexing is shown as (XY) instead of (X,Y).
Similarly, for those 4/95 instructions which are stack-relative memory reference,the index combinations may appear as (KX), (KY), (KXY) or just (K).

(iv) Whenever possible memory address operands are given in the G-relative form described in subsection 4.6.

(v) As already indicated, hexadecimal numbers omit the leading colon.

(vi) Negative hexadecimal immediate quantities omit both the = sign and the leading colon. Thus, for example, the assembler instruction

```
COPY =-:F,A
```

would be shown by SYMPLE as

```
COPY -F,A
```

(vii) When byte mode addressing has been selected by the Y option of some commands, all those instructions which are affected by byte mode have a B added to their op-codes (e.g. COPYB, ADDB), but their memory address operands are shown as the address of the WORD accessed by the instruction. If the instruction accesses the second (i.e. right hand) byte of the word, the character > is given immediately after the word address. This departure from normal byte addressing has been adopted because during debugging one is usually more interested in the word address than the byte address.

4.8 INPUT OF COMMANDS - GENERAL

Each SYMPLE command is input in reply to the > prompt. A command consists of one or more letters defining the command, together with one or more parameters (or operands) which are interpreted according to the particular command given. Sometimes some additional letters are given between the command proper and the first parameter, controlling options such as memory addressing modes. Normally the first parameter follows immediately after the command and option letters. If there are two or more parameters they are separated from each other by commas. The final parameter is terminated by carriage-return. If there are no parameters the carriage-return is given immediately after the command and option letters.

In some situations input of a single space or carriage-return is acceptable instead of a full command.

Some commands do not require a closing carriage-return, and they act as soon as they have recognised sufficient input to specify what is required of them.

Many commands allow a default option on some of their parameters. This means that if no characters are given for the parameter except for its closing comma or carriage-return, a default value is taken. The default value depends upon the command and parameter concerned, but it is almost invariably taken as the value previously specified for that operand. For example an I command given with no address will continue at the address reached when the previous I command terminated. Where several parameters in the same command are defaulted it is not necessary to terminate the line with a series of commas - the carriage-return can be given after the last non-defaulted parameter.

4.9 INPUT OF VALUES AND ADDRESSES

Many of the items input to SYMPLE in commands are values or addresses. These items are input in a standard form whether they are parameters in commands or operands in symbolic instructions (which may themselves be parameters in commands). Each value input consists of an expression with one or more "terms".

Terms are separated by plus or minus signs. Optionally, the first term of the expression may be preceded by a plus or minus sign. Terms and expressions are evaluated strictly from left to right. At the end of each term its value is added to or subtracted from the value of the expression accumulated so far, according to whether it was preceded by a plus sign or a minus sign. If the first term in an expression is preceded by a sign, the term is added to or subtracted from the default value for the expression. If the first term is not preceded by a sign, the term is stored as the current value of the expression and the default value is ignored. If the first character of the expression is not a plus or minus sign or the start of a valid term, a null expression has been input so the default value is retained and the first character is the terminating character. Otherwise the terminating character is the first character, other than a plus or minus sign, which follows a term but cannot form part of that term. The input is rejected if a plus or minus sign is not immediately followed by a valid term.

Each term consists of two parts. Either part may be absent but they cannot both be absent.

The first part of a term consists of a series of valid hexadecimal digits (0 to 9, A to F) and/or ASCII text characters. Each ASCII text character is input as two keystrokes, first a single quote character (') and then the actual character required. As a term is accumulated, only the least significant 16 bits are retained, so errors can be corrected by following them by a series of zeros and then the required digits.

The first part of a term is accumulated by the following steps:-

(i) Clear the term.
(ii) Get next character.
(iii) If the character is a valid hexadecimal digit, shift the interim result left by 4 bits, add in the new digit, and return to step (ii); otherwise continue to step (iv).
(iv) If the character is not a valid hexadecimal digit, test if it is a quote ('). If it is a quote, get the next character as a standard ASCII character (usually with a zero bit 7), shift the interim value of the term left by 8 bits and add in the new ASCII character value, then return to step (ii).

(v) If the character is neither a hex digit nor a quote the first part of the term is complete. The character is either the start of the second part of the term or not part of the term.

The second part of the term, if present, specifies a modifying value to be added to the first part. The modifier can take one of four forms:-

(i) The contents of a G-register, indicated by the letter G followed by a single digit or letter selecting the G-register.
(ii) The saved contents of one of the hardware registers, indicated by the letter R followed by one of the letters P, A, Q, X, Y, K, L, S, D or C to select the register.
(iii) The current address indicated by a dollar sign ($).
When used within a single-word symbolic instruction, $ is equivalent to the address of the instruction.
When used within a two-word symbolic instruction, $ is equivalent to the address of the second word of the instruction.
When used in a non-symbolic value to be stored or tested for in memory, $ is equivalent to the address of the memory location.
When used in defining a memory address $ is usually taken as equivalent to the current default value for that address.
(iv) The base address of the 64-word patch area in SYMPLE. This is indicated by the letter P. (Note that this is different from the user's P-register value which is denoted by RP).

The value of the term is formed by adding the values of its two parts (taking an absent part to have the value zero). Thus for example if GQ contains the value :1234 the following are three valid terms:-

```
33GQ    has the value :1267
33      has the value :0033
GQ      has the value :1234
```

Note that since a leading minus sign implies that the first term is subtracted from the default value, it is advisable to use a construction such as 0-3 to be sure of forming a true negative value, unless it is known that the current default value is zero.
Note also that when a term introduced by a minus sign contains a G, R, $ or P part, that part is effectively negated, since the value is added to the first part of the term and then the whole term is negated.

Some examples of valid expressions are given below, together with their equivalent values. The examples assume G1=2, G2=4, G3=6, G4=8, G5=A, G6=C, RA=E, RQ=10, $=1000, P=1234 and current default=2000. (All the values just given are hexadecimal):-

```
7FFF            = :7FFF
E08             = :0E08
+50             = :2000+:50 = :2050
123$            = :123+:1000 = :1123
3P+G1           = :3+:1234+:2 = :1239
5555+4          = :5555+:4 = :5559
G4              = :0008
600G2           = :600+:4 = :0604
-4G1            = :2000-(:4+:2) = :1FFA
227G6+G4+G3     = :227+:C+:8+:6 = :0241
RA+6RQ          = :E+:6+:10 = :0024
12'3            = :1200+:33 = :1233
'4'5            = :3435
'6G5            = :36+:A = :0040
```

4.10 INPUT OF ADDRESS RANGES

Several commands require a pair of operands to specify the range of addresses over which they are to act. In all such commands the value taken for the low end of the range (whether it is input or defaulted) becomes the default value for the high end of the range. This can be useful when the number of locations to be covered is known, but the high address is not known. For example to list the word located hex 436 after the contents of GH and the next hex 65 words, the command would be given as

```
L436GH,+65
```

This is exactly equivalent to

```
L436GH,49BGH
```

The commands which use address ranges (F, C, K, L and S) all use a common internal address counter. At the end of each such command this address counter contains the address of the word after the last word accessed by the command. This value is taken as the default for the low address on the next use of any of these commands. Thus consecutive areas may by accessed in a sequence of commands using the default option for the first address in a range. For example the command sequence

```
L1,+FF
L,+1FF
L,+F
```

is equivalent to the sequence

```
L1,100
L101,300
L301,310
```

In the case of the L command there is a further default option. If the first address is followed by carriage-return instead of a comma, the second address is taken as :FFFF instead of being taken equal to the first address. This allows listing to continue to the end of the store or until stopped by the operator.
A further extension of the default facility means that an L command given with no parameters and no commas, causes listing to continue from the present address (i.e. default for first parameter) to the end of memory or until stopped by the operator (i.e. special default for second parameter).

4.11 SYMBOLIC INPUT OF INSTRUCTIONS

Instructions input to SYMPLE in symbolic form look very similar to true MACRO assembler instructions. The most obvious difference is that all numbers are in hexadecimal but without a leading colon. There are other differences as will be seen in some of the operand descriptions given below. In general the form of input is identical with the form of output described in subsection 4.7.

The op-code is input first. If there are any operands there should be a space between the op-code and the first operand, and a comma separating each operand from the next. In fact SYMPLE allows commas or spaces to be used in either case, but the format given here makes the input layout the same as the output layout produced by SYMPLE.

When an operand is described as being an address or a hex value, input can be in any of the forms described in subsection 4.9, provided that the resulting value is possible as an operand for the function in question. Care should be taken when inputting hex values which could be confused with op-codes or register names (e.g. A, ADD, ADDB, ADDE). It is suggested that such values should always be given a leading zero in order to avoid all chance of confusion. This is particularly important when giving a memory reference instruction with hex address A, since to SYMPLE the instruction will almost certainly look like a reference to register A unless the address is given as 0A.

In analysing symbolic instructions, SYMPLE allows for 24 different operand constructions, as detailed below. Note that to the user some of these types appear to be identical, but within SYMPLE each type causes some unique action. The types are:-

1. Normal single word memory reference. Basically this consists of a memory address. If the instruction is indirect the address expression is preceded by an asterisk (*).
   If the instruction is indexed the address expression is followed by one of the following sequences depending on the type of indexing :
   (X)
   (Y)
   (XY)
   Note that the construction (X,Y) is not allowed.

2. Double word memory reference. Allowable input is exactly as for type 1.

3. Any one of the 8 register identities A, Q, X, Y, P, S, L or K.

4. Identical to type 3.

5. Any one of the 4 register identities A, X, Q or Y.

6. Either of the register identities A or Q.

7. Register and type specification for a single register SHIFT or ROTATE instruction. This covers two operand fields, the first specifies a single register (A, X, Q or Y), and the second specifies the shift type and direction (L, LO, R, RO or RA).

8. A condition tested by a conditional jump instruction. One of the letter pairs SS, OV or CY.

9. An 8-bit hex value, such as the device address and function code for an input/output instruction. Any expression which results in a value between :00 and :FF is acceptable.

10. A bit number. Any hexadecimal expression in the range :0 to :F

11. Same as type 9.

12. A register pair used in a double register instruction. One of the letter pairs AX, AQ or AY.

13. An 8-bit immediate value. This must be given as an = sign followed by an expression which results in a value between :00 and :FF.

14. A negative immediate value. This must be given as a minus sign followed by an expression which results in a value between :1 and :100 inclusive. Note that the leading minus sign is not part of the expression in this case; the expression is evaluated as though the sign were not present, and the sign is then used to indicate that a negative byte immediate instruction is to be formed.

15. A positive immediate value which is used negated within the instruction. This must be given as an = sign followed by an expression which results in a value between :1 and :100. (for example SUB =3,A).

16. S-register reference. Simply the letter S.

17. A count value for a SHIFT, ROTATE or SIN instruction. Any expression which results in a value between :1 and :10 (i.e. decimal 1 to 16).

18. The destination for a relative jump. Any address value which is within reach of the relative jump instruction.

19. A hexadecimal value for the second word of instructions such as PUSH. Any expression will do.

20. Memory reference for a byte mode instruction. Equivalent to type 1, but used when the byte version of a memory reference instruction is given (e.g. COPYB instead of COPY). The address expression given is treated as a word address, and the result is adjusted by SYMPLE so that it is stored as the byte address of the left hand byte of the word indicated by the address expression. If the right hand byte of the word is to be accessed the word address expression should be terminated by the character >. Note that if the instruction is indirect, the address does not have to be adjusted by SYMPLE, as the address part of an indirect instruction is always a word address.

21. A register and type specification for a double register SHIFT or ROTATE instruction. Identical with type 7, except that the first field specifies one of the register pairs AX, AQ or AY.

22. Stack relative version of type 2, but the address is always treated as an absolute offset value and the address expression is always followed by one of the following 4 sequences:
    (K)
    (KX)
    (KY)
    (KXY)

23. The same as type 22 except that the reference to the K register may be omitted.

24. Single word stack-relative memory reference that always has (K) only following the address expression. The instruction may be indirect, in which case an asterisk (*) preceeds the address expression.
    The address expression must evaluate to an absolute value in the range :0 to :3F.

4.12 STOPPING OUTPUT OPERATIONS

If a command is producing a large amount of output on the terminal or printer, the operator can halt the output at any time by pressing any key on the terminal's keyboard. The current line of output will be completed. Then a % error indication will be output and a prompt will be given for a new command.

If a memory listing command (L or LS) is interrupted in this way, input of a single space character in reply to the prompt will allow the listing to continue.

SECTION 5

SYMPLE COMMANDS

This section describes all the commands available in SYMPLE, in alphabetical order of their identifying letter or letter sequence. The information appears also in a condensed form in Appendix A.

Earlier sections describe the general method of giving commands and the construction of addresses and values.

Some of the items listed as separate commands are really variants on a single command. (For example LS is a special case of L).

AD : ASSIGN DEVICE FOR DUMP/LOAD/VERIFY

Command AD is used to select the medium for subsequent use in commands to make binary dumps of memory or to load or verify previously made dumps. The following options are available:-

ADF - Assign floppy disk unit 0 for dump/load/verify.
ADM - Assign magnetic tape unit 0 for dump/load/verify.
ADP - Assign paper tape (high speed reader/punch) for dump/load/verify. This is the default assignment if no AD command is used.
ADZ - Assign mini-flexible disk unit 0 for dump/load/verify

See commands D, LOAD and V for further information.

AL : ASSIGN DEVICE FOR LISTING OUTPUT

Command AL is used to select the listing output device. The options are:-

ALB - Assign listing to both terminal and printer. In this mode all output from SYMPLE is sent to both the terminal and the printer, and all input at the terminal is logged on the printer. Hence a hard copy is kept of all SYMPLE operations.

ALP - Assign listing to printer. In this mode output from commands L (including its variants), S, K and Z is sent only to the printer. All other output is sent to both the terminal and the printer, and all input at the terminal is logged on the printer. This is similar to the ALB mode except that commands which are likely to produce large volumes of output send their output only to the printer.

ALT - Assign listing to terminal. In this mode all output is sent only to the terminal. This is the default mode if no AL command is used (except as described in subsection 3.1).

The AL command accepts any combination of the following parameters, in addition to the letter B, P or T:-

,Fn - Feed control word for line printer (NOT serial printer).
n has the following meanings:-
0 = Printer does not require separate line-feeds, and does not react to form-feeds.
1 = Printer does require separate line-feeds, but does not react to form-feeds.
2 = Printer does not require separate line-feeds, but does react to form-feeds.
3 = Printer does require separate line-feeds, and reacts to form-feeds.

The initial default value is 3. Note that form-feeds should only be selected if the line printer can print at least 60 lines per page. For smaller page size this parameter should be set to 0 or 1, and the printer will then ignore page boundaries.

,Lda - Set device address of line printer (not serial printer) to the hex value of the parameter "da". The default value is :FE.

,P1 - Select the line printer (printer 1) for use in ALB and ALP modes.

,P2 - Select the serial printer (printer 2) for use in ALB and ALP

modes. Printer 1 (line printer) is the normal default setting, but see subsection 3.1 for the effect of entry by JST RORL.

,Sda - Set the device address of the serial printer to the hex value "da". There is no default value. If a serial printer is used, its device address must be set up by this command or as described in subsection 3.1.

,Tda - Set device address for the operator terminal to hex value "da". (Also see subsection 3.1). The default address is :F8. This option can be used to transfer operator control of SYMPLE to a different terminal when there is more than one connected to the computer.

It is not advisable to use any of the options Lda, P1 or P2 while a printer is actually in use. It is safer to select terminal only by giving command ALT with no parameters, then to use ALT again with the required parameters, and finally to reselect the printer with command ALB or ALP.

If SYMPLE attempts to output to a device which is not available, the program is held up until the output can be completed. Faulty device address selection in an AL command can cause this to happen. It is not easy to recover from faulty printer device address selection, but since there is usually only one printer on a system the device address should not often need to be changed. Faulty terminal device address selection can be cleared by entering SYMPLE again at its second word with the correct device address in the Q-register (see subsection 4.2).

B : CONTINUE USER PROGRAM WITH BREAKPOINTS

The action of traps and breakpoints has been described in sections 3 and 4. The B command is used to allow the user program to run with one or more operator selected breakpoints planted in it. Also, the command enables the Bus Monitor in accordance with registers M0, M1, M2 and M3 (see command M later in this section). Normally, the user program, after being entered by command B, will return control to SYMPLE subsequently by a type B or type M break, as described in topics 4.3.2 and 4.3.4.

A special facility in the B command can be used to enable the program to pass through a selected number of type B breakpoints before actually returning control to SYMPLE. This facility can only be used if at least two breakpoints are specified, and it will only work if the program while running does not go through one of its breakpoints repeatedly without going through any of the others. This is because it is implemented by actually breaking in the normal way; checking if the count has expired; and if not, returning to the user program with the breakpoint location restored to its correct contents but with traps in all the other breakpoints.

Command B can be given in any of the following forms:-

B	Continue to next breakpoint, using previously set breakpoints. (Rejected if there are not at least two breakpoints still set from a previous B or J command).
B*n	Continue through the next n breaks encountered, and break again at the (n+1)th. Uses previously set breakpoints with restrictions as above.
Bba,bb,bc,bd	Remove previous breakpoints. Set new breakpoints at ba, bb, bc and bd. Continue to the next breakpoint. Can specify one, two, three or four breakpoints.
Bba,bb,bc,bd,*n	Remove previous breakpoints. Set new breakpoints at ba, bb, bc and bd. Continue through the next n breaks encountered, and break again at the (n+1)th. Can specify two, three or four breakpoints, but not just one.

C : COPY MEMORY CONTENTS FROM ONE AREA TO ANOTHER

Command C copies the contents of a specified area of memory to another area of the same length. Command C is usually given in one of the following 2 formats:-

Cb,e,n copy contents of locations b to e inclusive to a new area of memory starting at location n.

Cb,+f,n copy contents of location b and the next f locations to a new area of memory starting at location n.

However, the b parameter may be omitted, or it may be specified as an offset to the current default value established by the last F, C, K, L or S command. See subsection 4.10 for complete details on address range specification.

In the final phase of the Copy operation, the destination contents are checked against the original values and any mismatch is indicated by a % , as noted in subsection 4.1.
It is recommended that the origin and destination areas do not overlap as mismatches may occur.

D : DUMP MEMORY CONTENTS

Command D dumps the selected area of memory to the device specified by the last AD command, or to the High Speed Paper Tape Punch if no AD command has been given. The dumped code can be loaded later either through the LOAD command (of SYMPLE) or via the hardware AUTOLOAD feature.
The format of the Dump command is:-

Db,e[,x]	Dump contents of memory locations b to e inclusive with an execution address of x. If x is omitted, the value hex 80 is assumed.

The dumped code should be verified by using the V command - see description later on in this section.
If the device (to which the dump is directed) has a problem of any sort (e.g. Write Protect is set), SYMPLE aborts the dump and outputs % on a new line before re-issuing its command prompt.

F : FILL MEMORY LOCATIONS WITH A DEFINED VALUE

Command F stores a 16-bit value into each location of a selected area of memory. The value is determined by the third parameter given in the command, the first two specifying the memory area. The command is usually given in one of the following formats:-

Fb,e,val	Fill locations b to e inclusive with values defined by val - see below
Fb,+f,val	Fill location b and the next f locations with values defined by val - see below.

However, the b parameter may be omitted, or it may be specified as an offset to the current default value established by the last F, C, K, L or S command. See subsection 4.10 for complete details on adddress range specification.

"val" may be expressed as a hex constant, in which case that value will be stored in each location selected.
"val" may be expressed as a hex expression which involves $ (current address), in which case the value stored in each selected location will be determined by the address of that location.
"val" may expressed as a symbolic instruction.

Examples:-

F2000,+F,3000	Fills locations 2000 to 200F inclusive with the value hex 3000.
F2000,+F,$	Fills each location from 2000 to 200F inclusive with the address of itself.
F2000,+F,$+$	Fills each location from 2000 to 200F inclusive with 4000,4002,4004...401E.
F2000,+F,JMP $	Fills locations 2000 to 200F inclusive with hex 9E7F (JMP $).

G : GENERAL REGISTER DISPLAY AND MODIFICATION

Command G enables the values of any of the 36 general registers to be displayed, and modified if necessary. In addition, the value displayed may be re-displayed as double or half its value. There are two versions of the G command:-

G — G followed immediately by carriage-return will produce a display of all 36 General Registers.
Note that any register which has not yet had a value assigned to it, always has the value 0000.

Gg — General Register g is displayed as soon as g is identified.
At this point, a choice of 5 actions is available:-

1. An immediate carriage-return terminates the command leaving the selected register value unchanged.
2. A new value, terminated by carriage-return, may be entered.
3. A # causes the value to be re-displayed in G-relative form.
4. A / causes half the value to be displayed, but no change is made to the actual register value.
5. A % causes twice the value to be displayed, but no register value change occurs.

After entering #, / or %, the choice of 5 actions is still available.

Examples:-

```
GH 0000 200
```

The example above displays General Register H (currently unassigned) and sets it to the value 0200.

```
G1 1234 /      091A %      2468 D283
```

The example above displays the value of General Register 1, displays half that value, displays twice that value and then sets a new value of D283.

I : INSPECT MEMORY NON-SYMBOLICALLY AND CHANGE CONTENTS

Command I allows the contents of any memory location to be displayed in hexadecimal and ASCII, and for the contents to be replaced with a new value, which may be expressed symbolically or as a hex expression, etc.

Two forms of this I command exist:-

Io Inspect the location which was inspected last by the previous I or IS command, or location 0 if this is the first I-type command.
Option o is any combination of the letters W, X and Y, or none of them, and their meanings are:-
W use non-extended addressing
X use extended addressing
Y use byte addressing where relevant
Since the displayed contents can be used to specify which is the next location to be inspected, option o is provided to control the way the contents are interpreted. See below for complete details.

Ioa Inspect location a if a is an absolute hex expression.
If a is an expression with a plus (+) or minus (-) prefix, then it is used as an offset either to the last address inspected by the previous I or IS command, or to zero.

When the location's contents have been displayed, the following options are available:-

display contents again, but in hexadecimal, and in G-relative form if appropriate.
Any of the options is still available.

/ display half the value in hex, and in G-relative form if possible.
Any of the options is still available.

% as for / but display twice the value.

. display the contents of the previous location on the next line.

blank (space) display the contents of the next location on the next line.

carriage-return terminates the command. However, following the subsequent > prompt, entering a blank (space) causes the I command to be re-invoked for the

same location just listed.

* save the address of the current location on a stack (in SYMPLE) which is 8-words deep. Then inspect the location addressed by the displayed value which SYMPLE treats as an address pointer according to the mode option of the I (or IS) command or the saved S-reg. The pointer is treated as a Byte address only if the Y option were selected.

@ if the current location contains a Memory Reference or Jump-type instruction, inspect the location addressed by this instruction whose own address is stacked as for the * option.
Extended addressing by the instruction is controlled by option o of the I (or IS) command or by the saved S-reg. Byte-mode addressing is used if the Y option is selected. The saved X and Y registers are used if the instruction specifies indexing.
If the instruction uses indirection, all intermediate addresses are listed.

! as for @ except any index- or indirect-address mode bits in the instruction are ignored.

; pop the last-stored address from the stack and continue inspection at that address.

I or IS (with usual parameters/options) continues inspection as specified by this new command.

'a new value' for the current location. This may be expressed as a value to be added/subtracted from the current value, or as a completely new value.
The value is expressible in any combination of hex., ASCII, G-reg, R-reg and $ forms, or as a symbolic instruction.
After entering the 'value', a carriage-return will cause inspection to be continued from the next location, whereas a .(period) will cause the newly-updated current location to be displayed on the next line.

When any error is noted by SYMPLE in the entered options, a ? is output at the beginning of the current line, and the current location's contents are re-displayed on the next line without the preceding address.

IS : INSPECT MEMORY SYMBOLICALLY AND CHANGE CONTENTS

The IS command is identical to the I command in all respects except that the selected location (by specification or according to the last I or IS command) is displayed in hexadecimal, ASCII and symbolic instruction forms.
The two forms of the IS instruction are:-

ISo Inspect Symbolically either location 0 if this is the first use of an I-type command , or the last location inspected by the previous I or IS command.

ISoa Inspect Symbolically. If a is an absolute expression, inspect the location whose address is the value derived from a.
If a has a plus (+) or minus (-) prefix, inspect the location whose address is evaluated by adding or subtracting the value of a from the address last inspected by an I or IS command. If this is the first use of IS, the value zero is used as the base address.

If the o option is specified and includes Y, any location displayed by the IS command which has a Byte form, (e.g. COPYB) will be listed as such, and the address accessed by the instruction will be displayed as the word address. If the instruction accesses the least significant byte, this word address will have a > suffix, provided the instruction is not indirect.

Examples of IS commands:-

```
IS202                                              - inspect loc. 202
     0202 0040   @ COPY Q,A ..........              - enter blank to inspect 203
     0203 BE8A >   JST 20E ........... @ - view loc. 20E
     020E 0E0D     HLT ............... ; - return to 203
     0203 BE8A >   JST 20E ........... ISX+1 - inspect 204; set option
                                                     X for extended addressing
     0204 1B00     UTRAP A,0 ......... * - inspect 1B00
     1B00 87B1   1 COPY A,*1B32 ...... @ - inspect loc. addressed
                                                     by contents of 1B32
     1B32 1AD7
     1AD7 4D50 MP CSK Q,=50 ......... ; - return to 204
     0204 1B00     UTRAP A,0 .........              - carriage-return terminates
                                                     command.
```

J : JUMP TO USER PROGRAM

The complete discussion regarding the use and effect of the J command is presented in subsection 4.5.
The J command enters the user program at the address specified by the saved P-register or as specified in the command, and clears all old breakpoints. New breakpoints may be set by the command. The command also enables the Bus Monitor according to the current settings of the M-register set and therefore, the user program having been entered by the J command, will normally return control to SYMPLE subsequently by a B- or M-type break.
The same facility exists as with the B command of allowing the user program to return only when a specified number of B-type breakpoints have been passed through.

Command J can be given in any one of the following forms:-

J	Return to user program according to the saved P register, restoring all the user's register values, and clear all breakpoints.
Jt	Return to user program at location t, restoring all other register values, and clear all breakpoints. If the user's XA bit in the S-reg. is zero, the J command will be rejected if t is >7FFF.
Jt,ba,bb,bc,bd	Return to user program at location t, restoring all other register values, clear all breakpoints and set new ones at ba, bb, bc and bd. One, two, three or four breakpoints may be specified.
Jt,ba,bb,bc,bd,*n	Return to user program at location t, restoring all other register values, clear all breakpoints and set new ones at ba, bb, bc and bd. The next n breaks encountered will be passed through, but control will be returned to SYMPLE on the (n+1)th break. Two, three or four breakpoints may be specified but not just one.

See also the X and RUN commands later in this section.

K : COMPARE TWO AREAS OF MEMORY

Command K compares the contents of one area of memory with those of another area, and lists the addresses of those words in the first specified area which do not match.

The command is usually given in one of the following formats:-

Kb,e,n — compare contents of locations b to e inclusive with the contents of another area of memory starting at location n.

Kb,+f,n — compare the contents of location b and the next f locations with the contents of another area of memory starting at location n.

However, the b parameter may be omitted, or it may be specified as an offset to the current default value established by the last F, C, K, L or S command. See subsection 4.10 for complete details on address range specification.

L : LIST MEMORY CONTENTS NON-SYMBOLICALLY

Command L lists the contents of consecutive memory locations in hexadecimal on the CRT and/or Printer as specified by the last AL command.
Listing continues until the specified limit is reached or when the user enters any character from the CRT keyboard.

The initial default listing mode is to print the address of the first location to be listed (including G-relative form if possible), followed by 8 consecutive contents across the line, and then a new line in the same format.
The number of values per line may be varied by a parameter in the L command. This number then becomes the new default.

The L command is usually given in one of the following formats, although there are variants, as explained below:-

Lb,e,n if all parameters are given, list the contents of locations b to e inclusive, with n location's contents per line.
If n is omitted (and the preceding ,) use the last value of n or the initial default of 8.
If e is omitted, listing continues until the end of physical memory is reached or when the operator stops it.
Note that if e is omitted, so also must the n parameter be omitted, and both commas.
Note also that the b parameter may be omitted, or it may be specified as an offset to the current default value established by the last F, C, K, L or S command. See subsection 4.10 for complete details on address range specification.

Lb,+f,n if all parameters are present, list the contents of location b and the next f locations, with n location's contents per line.
If any parameter is omitted, the effect is the same as described for the previous format and the rule about omitting the commas prevails.

When listing ends for whatever reason, it may be continued from the last address listed by entering a single Blank (Space) character. The effect is as if the user has entered an L command with no parameters.

See the LS command for examples of the Listing commands.

LM : LIST BUS MONITORED MEMORY REFERENCES

Command LM enables a list of the most recently recorded memory references to be output to the current listing device(s).
The number of references is limited to 80 by the Bus Monitor, and the command can limit this number even further if necessary.

The monitoring of memory references is initiated by SYMPLE when an X, B or J command is used to enter a user program, and monitoring is turned off when SYMPLE is re-entered. A subsequent LM command can access only those references that have been recorded since the last X, B or J command.

The list output is formatted into 4 vertical columns. Each column entry comprises the memory address in hexadecimal and G-relative forms and one or more of the following flags:-

R	Read Memory access
W	Write Memory access
I	Interrupt Request pending
D	DMA Memory access

The LM command format is:-

LMn List the last n memory references recorded by Bus Monitor.
n is a hex number in the range 1 to 50.
If n is omitted, the default value :10 is used.

If the Bus Monitor is not installed, LM produces no output.

See the M command later in this section for further details on controlling the Bus Monitor.

Example:-

Suppose the following code is executed while the Bus Monitor is enabled:

```
:1FBA.....COPY =2,A
:1FBB.....ADD  20,A
:1FBC.....COPY A,21
:1FBD.....<SYMPLE breakpoint>
```

if the user subsequently enters the command LM with no parameters, the listing produced will include the following recorded memory accesses, the last being the UTRAP sequence of the breakpoint code:-

1FBA R	0021 W	00A2 R	0211 R
1FBB R	1FBD R	00A3 R	
0020 R	00A0 W	020F W	
1FBC R	00A1 W	0210 R	

LOAD : LOAD NEW PROGRAM

The LOAD command accesses the device specified by the last AD command, loads the new binary program into memory and executes it according to the stored execution address.

If the load operation fails for any reason, e.g. the load may have over-written SYMPLE, the processor is halted.

The new program loaded from diskette or magnetic tape will have been dumped there by an earlier D command. Loading from paper tape, however, can be achieved using any Autoloadable paper tape. The V command can be used to verify that the binary program is loadable.

Command LOAD has no parameters.

LS : LIST MEMORY CONTENTS SYMBOLICALLY

Command LS lists the contents of consecutive memory locations in hexadecimal, ASCII and symbolic instruction formats. The output is formatted into columns which read vertically.

The formats of the LS command are similar to the L command:-

LSob,e,q — if all parameters are given, list the contents of locations b to e inclusive, with q columns across the 'page' .
q may be 1 or 2 for CRT output, or 1, 2 or 3 for printer-only output.
If q is omitted, output 2 columns as default, or as specified by the last occurrence of q.
If e is omitted, so also must q, and the upper address range FFFF is used.
Note that the b parameter operates as specified for the L command.

LSob,+f,q — if all parameters are given, list the contents of location b and the next f locations, with output formatted in q columns.
The omission of any parameter incurs the same effects as for the other form of the LS command, and the rule concerning the presence of the commas is as specified for the L command.

Option o has the same effect as in the IS command; that is, if Y is specified, the listing produced by the LS command will incorporate the byte versions of those instructions for whom it is applicable.

If the CRT receives the listing, output stops after 16 lines have been output. The next 'page' may be produced by entering a single Blank (Space) character, provided that the listing range has not been reached. If it has, a parameter-less LS command will suffice.

Example:-

The listing example on the next page illustrates the use of the LS command with the Y option included: note the ">" in some of the operands signifying the lower byte is being addressed; note also that the General Register G1 is set to the address :2000.

(LS example continued...)

```
LSY2000,+F

000G1=2000 36C2 6B JF OV,3G1
001G1=2001 0901    COPY =1,A
002G1=2002 FFA1  ! JSK *24G1
003G1=2003 0900    COPY =0,A
004G1=2004 8454  T COPYB A,0A(Y)
005G1=2005 F853 xS CSKB Q,9>(Y)
006G1=2006 0000    NOP
007G1=2007 FF9D    JSK *25G1
008G1=2008 C453 DS COPYB Q,9>(Y)
009G1=2009 2070  p COPY K,X
00AG1=200A 8082    COPYB 1(X),A
00BG1=200B A084    COPYB 2(X),X
00CG1=200C 4E79 Ny SHIFT Q,RO,8
00DG1=200D 580F X  AND =F,Q
00EG1=200E 430A C  XNX Q
00FG1=200F FF8A    JSK *1AG1

010G1=2010 8052  R COPYB 9(Y),A
011G1=2011 0AE8  h SUB =18,A
012G1=2012 13C1  A JLT A,14G1
013G1=2013 FF91    JSK *25G1
014G1=2014 8054  T COPYB 0A(Y),A
015G1=2015 1142  B JEQ A,18G1
016G1=2016 FEDE ~^ JSK 75G1
017G1=2017 FEF8  x JSK 90G1
018G1=2018 6309 c  POP
019G1=2019 2309 #  RSK
01AG1=201A 2027  ' OR X,X
01BG1=201B 202A  * CSN X,X
01CG1=201C 202F  / TBIT 2,X
01DG1=201D 2034  4 AND S,X
01EG1=201E 203B  ; CSK X,S
01FG1=201F 203D  = RBIT 3,X
```

M : BUS MONITOR REGISTER DISPLAY AND MODIFICATION

There are 4 special M registers associated with the use and control of the Bus Monitor, namely M0, M1, M2 and M3, and they are displayed and set in the same way as the G and R registers.

M0 holds the Bus Monitor control mode value - see below.
M1, M2 and M3 hold memory addresses which are to be stored in the Bus Monitor's comparator registers A, B and C respectively.

When a User Program is entered via a J, B or RUN command, the Bus Monitor is enabled according to the settings of the M registers. Initially, i.e. upon first entry to SYMPLE, all 4 M registers are set to zero.
The registers are set by issuing the following command:-

Mm register m is displayed as soon as m is identified.
At this point a choice of 5 actions is available:-

1. An immediate carriage-return terminates the command leaving the current value unchanged.
2. A new value may be entered followed by a carriage-return.
3. A # re-displays the value in G-relative form.
4. A / displays half the value, but the original value remains unchanged.
5. A % displays twice the value, but no change is made to the original.

M0, the Bus Monitor control mode register, controls the conditions under which the Bus Monitor interrupts or resets the processor.
The initial default value of zero inhibits the Bus Monitor from invoking a reset or an interrupt, but the function of recording memory references continues while the User Program is running following a J, B or X command.

The value to be stored in M0 is computed by adding bit-values as shown in the table on the next page.

BIT	VALUE	FUNCTION
0	:001	monitor all non-DMA WRITE accesses
1	:002	monitor all non-DMA READ accesses
2	:004	NOT TO BE USED - should be zero
3	:008	monitor all DMA accesses
4	:010	check for address reference = M1
5	:020	check for address reference = M2
6	:040	check for address reference = M3
7	:080	check for address reference > or = M1 and < or = M2
8	:100	check for address reference < M1 or > M2
9	:200	Reset CPU when any comparator condition is met
10	:400	Interrupt when any comparator condition is met

Any bit in the bit-group 8,7,6,5,4 may be on, thereby producing a combination effect. Similarly, any bit in the bit-group 3,1,0 may be on.
If a memory reference address satisfies any of the comparator conditions, and it is one of the types selected by the 3,1,0 bit-group, then the action taken by the Bus Monitor is as specified by bits 10 and 9.

The following command sequence sets the Bus Monitor registers so that SYMPLE will regain control if an address in the inclusive range 2000 to 204E is generated by the User Program. This address may be of an operand or an instruction or an address pointer used in indirection:-

```
M0 0000 400+80+2+1
M1 0000 2000
M2 0000 204E
```

N : CONTINUE USER PROGRAM TO NEXT LOCATION

Command N is a shorthand form of the command

BRP+1

That is, it causes any previous breakpoints to be removed and a new one to be stored at the location following the current location (whose address is the saved P-register value).

Command N requires no carriage return to terminate it.

Command N is useful when single-step execution of a user program is required.

N.B. The instruction at the current location must be a single-word instruction.

P : CONTINUE USER PROGRAM TO PREVIOUS LOCATION

Command P is a shorthand form of the command

```
BRP-1
```

That is, it causes any previous breakpoints to be removed and a new one to be stored at the location preceding the current location (whose address is the saved P-register value).

Command P requires no carriage return to terminate it.

Command P is useful for checking program loop sequences when it is required to continue executing the loop code until the previous location is reached.

R : USER PROGRAM REGISTER DISPLAY AND MODIFICATION

Command R enables the values of the saved User Program register context to be displayed, and modified if necessary. In addition, the displayed value may be re-displayed as double or half its original value, or in G-relative form. The two formats of the R command are:-

R — R followed immediately by a carriage-return will produce a display of all the saved User Program registers in the same format as given when a breakpoint occurs, except that the B is dropped from the display.

Rr — User Program register r is displayed as soon as r is identified. At this point, a choice of 5 actions is available:-

1. An immediate carriage-return terminates the command leaving the selected register value unchanged.
2. A new value, terminated by carrriage-return, may be entered.
3. A # produces a re-display of the value in G-relative form.
4. A / causes half the value to be displayed, but the register value is unchanged.
5. A % causes twice the value to be displayed, but no change occurs to the register.

After entering #, / or %, the choice of 5 actions is still available.

Example:-

```
RA 0020 2A20
```

Register A of the User Program contains :0020 and the user has changed it to :2A20. This new value will be stored in the A register when the User Program is re-entered.

RUN : RUN USER PROGRAM

Command RUN enables the User Program to be (re-)entered according to the saved/current User register context. Any breakpoint still outstanding will be cleared, but the trap locations will be left to return to SYMPLE in the event of an error. The Bus Monitor, if installed, will be enabled in accordance with the latest M commands.

A RUN command is the same as a J command with no parameters.

S : SEARCH MEMORY

Command S provides the user with means of searching memory for a particular value, or for like values, or for references to a specified address.
It is also possible to search an area of memory for no matches with a specified value.

Each of the 4 different versions of command S has 2 commonly used formats:-

Sb,e,val Sb,+f,val	Search memory from locations b to e inclusive, or location b and the next f words, for a match with val. All matches are listed.
Sb,e,val,m Sb,+f,val,m	Search memory from locations b to e inclusive, or location b and the next f words, for value val under mask m, where zero bits in m are "don't care" bits. All locations matching the search conditions are listed.
Sb,e,val,- Sb,+f,val,-	Search memory from locations b to e inclusive, or location b and the next f words, for no matches with val. All non-matches are listed.
Sb,e,val,R Sb,+f,val,R	Search memory from locations b to e inclusive, or location b and the next f words, for references to address val. This form of search will find any instruction whose effective address is equal to "val". In calculating the effective address, the saved X and/or Y registers will be used if the instruction uses indexing. See below for additional reasons for determining a match in this Search command.

In the above command formats, b may be omitted, or it may be specified as an offset to the current default value established by the last F, C, K, L or S command. See subsection 4.10 for complete details on address range specification.

All listings produced by the above forms of the S command give the addresses where the 'match' occurred together with the contents in hexadecimal and ASCII. The symbolic instruction represented by the contents can be listed also by including the option S after the S command letter.
This is most useful in the last form of the Search command where one is looking for any instruction which addresses a particular location.

The optional letters W, X and Y may be included, in any combination, following the Command letter S, and have the same effect as when used with the I and Is commands. That is, they affect the addressing mode for both the search value and the output.
In the reference-search command, if word mode is indicated (by default), any word containing "val" or "val" plus :8000 will be listed. If byte mode is indicated (by Y), any word containing 2x"val" or 2x"val" plus 1 will be listed.

"val" may be expressed as a simple value (e.g. 55) or a symbolic instruction or a complex expression involving ASCII, registers and $. If $ is included in the search "val", it is interpreted afresh for each location that is tested.
If "val" is a memory reference instruction in symbolic form, it may not be valid at all points in the search area, in which case a % will be output at the end of the search operation.

Examples:-

1. To search for any SIN instruction in locations :2000 to :2FFF inclusive, the following command and reply sequence might occur:-

```
SS2000,2FFF,SIN 10,FF0F
    261B 0E2F / SIN 3
    2F5F 0EBF ? SIN C
    2F8B 0E4F 0 SIN 5
```

   The mask FF0F is necessary because the variable part of a SIN is the bit-field represented by the 0.

2. To find all references to location :13FD in the area :1000 to :1FFF inclusive, the following command and reply sequence might occur:-

```
SS1000,1FFF,13FD,R
    138E DEEE ^n IMS 13FD
    13BA C6C2 FB COPY Q,13FD
    140C DE70 ^p IMS 13FD
    141E A65E &^ COPY X,13FD
    1B3A 13FD
```

V : VERIFY DUMP IS LOADABLE

Command V verifies that the medium selected by the last AD command contains a binary program that is capable of being loaded by the LOAD command of SYMPLE.

It does NOT verify the binary program against memory contents.

It can be used to check any Autoloadable Paper Tape or any dump on Floppy Disk, Mini-Flexible Disk or Magnetic Tape produced by SYMPLE's D command.

Command V has no parameters.

If SYMPLE finds that the current load/dump/verify device does not contain a loadable program, it will output % and prompt for the next command.

X : EXIT FROM SYMPLE

Command X is similar to a J command, but it is used in the case where the user has finished debugging and wishes to return to the user program or, more usually, to the operating system, like OS4.

The command differs, therefore, from the J command in the following respects:-

(i) X removes all breakpoints and does not allow new ones to be specified in the command.
(ii) X clears register M0 so that the Bus Monitor cannot interrupt; however, it will continue to record memory references.
(iii) All trap locations (except Power-fail) and location 0 are restored to the values they had when SYMPLE was last entered, or re-entered (by a J or B breakpoint), or as specified in an I or IS command, whichever is the most recent event. Note that any attempt to change these location contents by Inspect commands results in the saved values changing, not the locations themselves.

The format of the X command is:-

Xp Exit from SYMPLE to address p. If p is omitted, the saved P-register value is used.

Thus the command

```
X101
```

returns control to OS4, causes EXEC to be loaded and a JOB function to be invoked.

Y : LIST SYMPLE'S HIGH/LOW MEMORY ADDRESSES

Command Y lists the first and last addresses of SYMPLE.

Thus, if SYMPLE is executed from OS4, the command and output values will be similar to:-

```
Y 0200 130F
```

Z : PRINT CHAIN

Command Z prints the contents of chains (sometimes known as linked lists or threaded lists). These data structures are used throughout RTX4/IOS4 and are created by means of the Assembler Directive CHAN.

Each entry in a chain is called a node and a word in each node is reserved to hold the address of the next node in the chain. This link-word can be in any position in the node but the default position assumed by SYMPLE is word 0.
The end of a chain is indicated by the link-word containing zero instead of an address.

The format of the Z command is:-

Zs,w,d,n Print chain, where
- s = address of word 0 of the first node
- w = number of words to list per node
- d = relative displacement of link-word in node
- n = number of words to print on each line

If w is less than n, one line will be printed for each node. Otherwise, the number of lines necessary to print each node are printed with the second and subsequent lines indented to the right.
The first line of each node begins with the address of that node.

Any or all of the parameters w, d and n may be omitted (with a corresponding dropping of trailing commas). If a parameter is omitted, SYMPLE uses the last value specified, or the initial default for that parameter. The initial default values are:-

w=1 d=0 n=8

Thus the command

```
ZFFE3,4
```

produces a printout similar to

```
FFE3  FFAF 0000 0000 FFC7
FFAF  EFC5 0000 0000 FF93
EFC5  EFAC 0000 FFE3 EA72
.
.
.
```

FFE3 is the address of the first node and the link-word, FFAF of the

second and so on.
The contents of the first 4 locations for each node are printed alongside each start address.
If the address of a node can be expressed in G-relative form, that value will precede the absolute address in the normal way.
If more than one line per node is printed, the indentation start at the absolute address position.

[: COMMENT LINE

A comment to assist in the later analysis of one's debugging efforts can be printed on the TTY/Line Printer by means of the [character; thus

[I think that's the correct mod now

will appear on the log if an ALP or ALB command had been given previously.

APPENDIX A

COMMAND SUMMARY

ADF .....Assign Floppy Disk Unit 0 for D/V/LOAD commands
ADM .....Assign Magnetic Tape Unit 0 for D/V/LOAD commands
ADP .....Assign Paper Tape (High Speed Reader/Punch) for D/V/Load commands.
ADZ .....Assign Mini-Flexible Disk Unit 0 for D/V/Load commands.

ALB .....Assign listing output to terminal and printer
ALP .....Assign listing output to printer only
ALT .....Assign listing output to terminal only

ALB/ALP cause all terminal I/O to be logged on printer.
All AL commands accept further parameters:-

,Fn Line printer control word where n can be:-
0 = no line-feed or form-feed to be used
1 = use line-feed, form-feed not used
2 = use form-feed, line-feed not used
3 = use both line-feed and form-feed (default)
,Lda Set Line printer device address (default - :FE)
,P1 Select Line printer for list/log
,P2 Select Serial printer for list/log
,Sda Set Serial printer device address
,Tda Set terminal device address (default - :F8)

B .....Continue to next breakpoint - at least 2 must exist.
B*n .....Continue through next n breaks and break on (n+1)th
Bba,bb,bc,bd .....Continue to one of the new breakpoints
Bba,bb,bc,bd,*n ..Continue through the next n new breakpoints and break on the (n+1)th. At least 2 breakpoints must be specified.

Cb,e,n .....Copy words b to e inclusive to n onwards
Cb,+f,n .....Copy word b and the next f words to n onwards

Db,e,x .....Dump words b to e inclusive to ADd device with execution address x
Db,e .....As before except execution address is :80

Fb,e,val .....Fill words b to e inclusive with value defined by val

Fb,+f,val .....Fill word b and the next f words with value defined by val

G .....Display all General Registers
Gg .....Display General Register g; after display, enter:-

for re-display in G-relative form, and
% for re-display as twice actual value, and
/ for re-display as half actual value, or
new value and carriage-return, or
carriage-return only to exit command.

I .....Inspect memory in non-symbolic mode (hex and ASCII) using address retained from last I or IS command.
Ia .....Inspect non-symbolically location a
IS .....Inspect memory in symbolic mode (hex,ASCII and Instruction mnemonic forms), using address from last I or IS command.
ISa .....Inspect symbolically location a

For all I-type commands, now enter:

for re-display in G-relative form
% for re-display as twice actual value
/ for re-display as half actual value
. for inspection of preceding location
blank (space) for inspection of next location
carriage-return for terminating command - a subsequent blank entered after the > prompt re-inspects same location
* for inspection of location whose address is displayed value
@ for inspection of location addressed by displayed instruction
! as @ but ignoring indirect/index address mode of instruction
; for inspection of location viewed before last *, @ or !
I or IS-type command for new inspection
new value to replace current one, expressed in hex, mnemonics, etc... followed by
carriage-return to continue inspection at next location, or
a "." (period) to re-inspect current location.

For all I or IS commands, an optional combination of W, X and Y may follow I or IS, preceding any other parameter; thus:
Y specifies Byte-mode addressing for * @ and !

parameters, and display of Byte-mode instruction variants, where applicable.
W (XA-off) and X (XA-on) are used by * @ and ! parameters to determine address of next location to be inspected.

J .....Return to user program at saved P-reg address, restore user reg. context, clear breakpoints, enable Bus Monitor and leave trap locations to return to SYMPLE.
Jt .....as J but return to address t
Jt,ba,bb,bc,bd ...as Jt and continue to new breakpoint
Jt,ba,bb,bc,bd,*n as Jt and continue through next n breaks, breaking on (n+1)th. At least 2 breakpts must be specified.

Kb,e,n .....Compare words b to e inclusive with words from n onwards
Kb,+f,n .....Compare word b and next f words with words from n onwards

L .....Continue non-symbolic listing from where last L stopped
Lb,e,n .....List words b to e inclusive, n words per line (defaults to 8 initially)
Lb,+f,n .....List word b and next f words, n words per line

When listing completes, enter blank (space) to continue listing.
Enter any character to stop listing prematurely.

LM .....List up to 16 most recent memory accesses recorded by Bus Monitor.
LMn .....List up to n (max 80) most recent memory accesses

LOAD .....Load new program from ADd device and execute it

LS .....Continue Symbolic listing from where last LS stopped
LSb,e,q .....List Symbolically words b to e inclusive in q columns
LSb,+f,q .....List Symbolically word b and next f words in q columns

Listing stops when 16 lines have appeared on CRT - enter blank (space) to continue if range not exhausted.
Enter any character to stop listing prematurely.

An optional Y may be given after LS and preceding other parameters, to list Byte-mode variants of instructions, if they exist.

Mm .....Display Bus Monitor m; after display, enter:-

for re-display in G-relative form, and
% for re-display as twice actual value, and
/ for re-display as half actual value, or
new value and carriage-return, or
carriage-return only to exit command.

M1, M2 and M3 correspond to Bus Monitor comparators A, B and C.
M0 is coded to control the Bus Monitor action as follows:-

Bit	Hex	Function
0	001	monitor Write accesses (non-DMA)
1	002	monitor Read accesses (non-DMA)
3	008	monitor DMA accesses
4	010	check address = M1
5	020	check address = M2
6	040	check address = M3
7	080	check address >= M1 and <= M2
8	100	check address < M1 or > M2
9	200	reset processor if address meets conditions
10	400	interrupt if address meets conditions

N .....Continue to breakpoint at Next location

P .....Continue to breakpoint at Previous location

R .....Display all saved User Registers
Rr .....Display User register r; after display, enter:-

for re-display in G-relative form, and
% for re-display as twice actual value, and
/ for re-display as half actual value, or
new value and carriage-return, or
carriage-return to exit command.

RUN .....Return to User Program at saved P-reg address, restore User Registers, clear breakpoints, enable Bus Monitor and leave trap locations to return to SYMPLE.

Sb,e,val .....Search words b to e inclusive for match with val
Sb,+f,val .....Search word b and next f words for match with val

Sb,e,val,m .....Search words b to e inclusive for match with val under mask m

Sb,+f,val,m .....Search word b and next f words for match with val under mask m

Sb,e,val,- .....Search words b to e inclusive for non-match with val

Sb,+f,val,- .....Search word b and next f words for non-match with val

Sb,e,val,R .....Search words b to e inclusive for any instruction whose effective address is val. Any word containing val or val+:8000 will be found also.

Sb,+f,val,R .....Search word b and next f words for any instruction whose effective address is val, or for any contents equal to val or val+:8000.

Include an extra S after Command letter S to list matches in Symbolic form in addition to normal hex/ASCII output.
The optional letters W,X,and Y after Command letter S control the search when addressing is involved. W specifies XA-off mode,X - XA-on mode and Y - Byte mode.
Y also finds twice val or twice val +1 for Reference Search.

V .....Verify dump can be LOADed.

X .....Exit from SYMPLE to saved P-reg address, restore all saved User Registers, clear all breakpoints, clear Bus Monitor control mode word M0, and restore Trap locations.

Xp .....as X but exit to location p.

Y .....Print memory limits of SYMPLE

Z,s,w,d,n .....Print chain, where:-
s = address of first node's word 0
w = number of words to list per node
d = node's link-word position
n = number of words per line

Initial defaults are w=1 d=0 n=8

[.....Comment line

APPENDIX B

LINKING, LOADING AND CALLING SYMPLE

B.1 LINKING SYMPLE

B.1.1 Files Supplied to Users

SYMPLE.SYS .....SYMPLE binary file - includes all features.

SYMPLE.OBJ .....Main SYMPLE object file.
SYMPLE:TAB.OBJ...SYMPLE's symbol table object file.
SYMPLE:MON.OBJ ..SYMPLE's optional Bus Monitor module object file.
SYMPLE:BOT.OBJ ..SYMPLE's tail-end object file.

SYMPLE:GEN1.JCL .Job file to link complete SYMPLE program.
SYMPLE:GEN2.JCL .Job file to link SYMPLE without Bus Monitor module.

B.1.2 Linking SYMPLE Only

Both Job files create a new SYMPLE.SYS file, overwriting any earlier version. The Job files assume UF contains the SYMPLE object files.

B.1.3 Linking SYMPLE to User Program

The object files of SYMPLE are identified in B.1.1. The order of linking is as presented there, and these modules would appear in the LINK (OS4 Linker) command line after the User Program module(s).

For example:-

```
LINK USERPROG+SYMPLE+SYMPLE:TAB+SYMPLE:MON+SYMPLE:BOT
```

The above command line assumes that the User Program makes an external reference to the defined entry points to SYMPLE, thus, there is no need to force-load the SYMPLE objects.

The external names are DEBUG, DEBUG4, SYMPLE and RORL, the last being the Subroutine entry point for User Programs. The normal way to reference one of the first three names is:-

```
LOAD SYMPLE
```

In the case of RORL, the user includes the statement

```
EXTR RORL
```

somewhere in his program prior to the call to it via a JST instruction. See B.3.

B.2 LOADING SYMPLE

Under OS4, SYMPLE may be called in two ways:-

SYMPLE or file;SYMPLE

In the first case, SYMPLE is loaded by itself from location :200 onwards, while, in the latter case, it is loaded after the User Program 'file' but it will be executed first.

B.3 CALLING SYMPLE

Calling SYMPLE from a User Program has been described in detail in subsection 3.1.

This Appendix summarises the call and parameters.
First of all, the two statements

```
EXTR RORL
JST RORL
```

are necessary to enter SYMPLE at the Subroutine entry point.

The possible parameters are:-

Y-reg
- =0 no program title or revision number is output.
- =a program title located at location 'a' is output, together with revision number which is zero initially, and is incremented every time a Dump is made.

A-reg
- =0 SYMPLE uses standard terminal
- =st where 's' and 't' are used to specifiy serial printer and terminal device addresses as follows:-

  's' is the upper byte and if it is non-zero, it defines the device address of a Serial printer which is assigned to log all terminal I/O as soon as SYMPLE is entered.

  't', the lower byte, specifies the initial device address of the operator terminal to be used when SYMPLE is entered.

Both device addresses may be changed subsequently by SYMPLE commands.

Example:-

```
          EXTR   RORL
          .
          .
          .
          COPY   =TITLE,Y
          COPY   =CHAN2,A
          JST    RORL          Enter SYMPLE Subroutine
          .
          .
          .
TITLE     BYTE   'REACTIVATE MKI'
          WORD   0
CHAN2     WORD   :F4           IOD CHANNEL 2
```

APPENDIX C

SUMMARY OF RECOGNISED INSTRUCTION MNEMONICS

All instructions available in the basic NM4/90 processor are recognised by SYMPLE and are output, when encountered during a list/inspect memory operation, in the format described in sub-section 4.7.
In addition to the NM4/90 instruction set, the following instructions are recognised:-

- 7 Floating Point instructions (includes FIX/FLOAT)
- SAL (Software Autoload) instruction in SCOUT (NM4/04)
- all additional instructions available in the NM4/95

Note 1: The operand to the SAL instruction is always output/input as a single hexadecimal expression,
e.g. SAL 24
where :2 is the ROM file number
and :4 is the board ID number

Note 2: The 4/95 instruction NEXTQUE is too long for SYMPLE's symbol table and therefore the mnemonic NXTQUE should be used instead.