M U L T I 8   T E R M I N A L   M A N U A L

Fourth printing, july 1979

- MULTI8 Terminal Manual -

TABLE OF CONTENTS

This manual is intended to assist the terminal users of the MULTI8 realtime/timesharing system. It describes the features of MULTI8 V7A. Specific information on the system's internal operation can be found in the MULTI8 System Manual. Copies of these manuals may be ordered from:

Westvries Computer Consulting B.V.
Rijksstraatweg 19, 1969 LB Heemskerk
The Netherlands

1. Introduction.

This manual is intended for those who will use a terminal connected to a MULTI8 system. MULTI8 is a powerful realtime/timesharing system for the family of PDP8° computers. Multiple users can have access to their own "virtual" PDP8 and run the standard OS/8° operating system, including a full complement of utility programs, editors, assemblers, compilers and a large library of user programs. Supported peripherals include disks (RF08, DF32, RK8E, RL01, System Industries 3040), DECtape (TC08 and TD8E°°), lineprinters, papertape reader/punch, cardreader, magtape, floppy disks, plotter, EAE and a variety of terminals. This manual is not an introduction to OS/8, but rather discusses the differences between stand-alone OS/8 and the MULTI8 background. You should be familiar with the OS/8 system.

Getting On-Line (chapter 2) will help you the get started on a MULTI8 terminal. In chapter 3 we explain that nothing has to be explained about sharing peripherals; it's all automatic. Chapter 4 of this manual describes the way in which terminal input/output is affected by MULTI8. Chapter 5 describes the commands that enable the user to operate his virtual machine from the terminal. Chapter 6 details differences in the operation of standard OS/8 programs. For the assembler programmer chapter 7 gives details of the virtual machine's I/O instructions. In the appendix (chapter 8) you will find a list with all the IO instructions and how the are handled by MULTI8. The reader may use the OS/8 Handbook for reference.

2. Getting On-Line.

When MULTI8 V7A is started, all terminals will display:

```
        PASSWORD ? _
```

You must enter the correct password to gain access to the system. MULTI8 is distributed with the password 'PLEASE', but the password may have been changed by your system manager. If you enter the correct password (followed by Carriage Return), the system displays a welcoming message that identifies your terminal number. It then displays a message from your system manager and starts OS/8.

Under MULTI8 several new CCL commands are available, and one CCL command is modified. The new commands are implemented by the program XCL.SV, (for eXtended Command Language) which can be seen as an extension to CCL.SV. It implements the following commands:

BYE	Starts the MULTI8 logout procedure.
TALK n xxxxxx	Sends the message xxxxxx to terminal n. If n=0, the

------

° PDP8 and OS/8 are trademarks of Digital Equipment Corporation.
°° The use of TD8E DECtape restricts the realtime capabilities of the system.

	message is broadcasted to all terminals on the system.
OPEN DSKn:	Opens the device DSKn: which gives read-only access to the disk-area of terminal n.
OPEN DSKn:/W	Same, but allows you to write as well.

N O T E

Do not write onto another user's disk area when he has an open output file to SYS:. You would thereby delete his tentative file in his directory.

CLOSE DSKn:	Make DSKn: inaccessible
CLOSE ALL	Make all DSKn:'s inaccessible, except DSK0:.
CLOSE	Same as CLOSE ALL

The CCL command DATE has been modified to give the date and the time. The day-of-the-week is no longer displayed, eg.:

```
        .DATE

        17-FEB-79 10:45
```

As compared with a regular OS/8 user, a MULTI8 user has a slightly more complex (and powerful) 'environment'. Each terminal has its own private disk area, that is used as SYS: and DSK:. But each terminal also has (read-) access to DSK0:, a disk area that holds the MULTI8 system files plus any files that are useful for all terminals, eg. all CUSPS (PIP, DIRECT, etc.) and any installation specific utility or application programs. You can run these programs by typing .RUN DSK0:PIP. This, of course, is not very convenient, and more important, you can't CHAIN to programs that are not on SYS:. Therefore MULTI8 includes a utility program SHSAVE (SHort SAVEfiles), that creates small files on your SYS:. Such a file gets the name of a program that is on DSK0:, and it realy seems that that program is on your SYS:. You can 'R' it, chain to it, etc. Yet it takes only 2 blocks of disk space. Here follows an example of the use of SHSAVE.

```
.R PROG
PROG.SV NOT FOUND
.RUN DSK0:SHSAVE
*PROG$
.R PROG
XXXXXXXXXXXXXX (PROG RUNS)

.DIR PROG.SV

20-FEB-79

PROG   .SV   2 13-MAR-77

   1 FILES IN   2 BLOCKS - 2431 FREE BLOCKS

.
```

Note that the short file gets some atributes from the real file that resides on DSK0:, its creation date and job status word. In fact it contains a small program that reads the real program from DSK0: and starts it. SHSAVE.SV has a number of options:

/A	All, combine option B, F, P, S, X AND 4.
/B	Short-save the BASIC system.
/F	Short-save the Fortran II system.
/L	List the files while they are processed.
/N	No message for files that are not found on DSK0:
/P	Short-save the PAL8 system.
/Q	Ask the user to confirm each file.
/S	Short-save the Standard programs, eg. PIP, DIRECT, etc.
/X	Short-save a set of installation-dependent programs.
/4	Short-save the Fortran IV system.

Programs that are overlayed or otherwise access their own .SV files can not be processed by SHSAVE, eg. BATCH, MACREL, LINK and the BASIC overlay files.

N O T E 1

FOTP (ie. the COPY command) gives a misleading error message when you try to write on a device that is read-only. FOTP calls out NO ROOM, SKIPPING-XXXXXX. This will occur if you try to write on eg. DSK3: after the command .OPEN DSK3:.

N O T E 2

If, before installing MULTI8, you used to have SYS: and DSK: assigned to different devices, you may have created programs that voilate the rule that never more than one output file can be open on a single directory. Under MULTI8 each terminal has SYS=DSK. The results will be most peculiar, for instance your disk may seem to 'shrink' each time you use that particular program. It is a permanent restriction of OS/8 that no warning can be given in this situation.

3. Sharing peripherals.

Sharing peripherals (lineprinter, DECtape, floppy disk, etc) between users is completely automatic. You never need to 'assign' or 'allocate' devices. The first user that grabs a device is the first to use it; if another user tries to use the same device, his program is delayed until the device can be used again. That may be just after the first user finishes his current operation (eg. for DECtape, floppy- and other disks) or after he ends his file (eg. lineprinter, cardreader). If you sent a file to the lineprinter and it turns out that another user is already using the printer, you can interrupt your program with CONTROL/B. This makes your terminal free again, so that you can perform a different task first. If you do not interrupt your program, it will proceed as soon as the printer is available.

A warning is in place regarding multi-unit devices like DECtape and floppy disk. MULTI8 gives no protectection agains simultanious use of one tape of diskette by multiple users. If more than one program writes on a device, directory problems will occur. Generally these problems do not occur as each user will use its own media.

4. Terminal I/O.

MULTI8 tries to behave just as a bare OS/8 machine would do. However, on a few minor points it is impossible or undesirable to maintain exact correspondence.

- When you enter a RUBOUT (DELETE on some keyboards), there may be a short delay in the programs reaction due to program swapping. This may result in garbled output. Enter LINEFEED to have the line retyped (this is an OS/8 feature not supported by all programs).

- The characters CONTROL/S and CONTROL/Q stop/start terminal output. This is a convenient feature for video display terminals. CONTROL/S and CONTROL/Q are interpreted by MULTI8 and are never passed to your program. CONTROL/C, CONTROL/O and CONTROL/B perform an implicit CONTROL/Q.

N O T E

When your terminal is apparently "dead", ie. does not echo any characters, you probably struck CONTROL/S by accident. Type CONTROL/Q to reactivate the terminal.

- All keyboard input is buffered by MULTI8. You may type one or more new commands while the system is still processing a previous command ('type ahead'). At a certain point, depending on the activity on the system, you will reach the end of the input buffer. In that case all further input is rejected; input characters are not echoed; instead, the terminal bell is rung. Wait until some of your commands have been processed and continue with the first character that was rejected.

- On receipt of CONTROL/C or CONTROL/O, MULTI8 will clear both the terminal input and output buffer to insure quick response. Normally your program will be ahead of the terminal output. This is caused by the output buffering. So if you interrupt your program with CONTROL/C (or CONTROL/O) the last characters displayed do not acurately show how far your program did proceed.

- MULTI8 can give each terminal the right number of filler characters. If your terminal hesitates at the beginning of each line or if the first characters of each line are scrambled, see your system manager. He can adjust the amount of filler characters for your terminal. See also the DELAY command, chapter 5.

- If you have a video terminal and the RUBOUT sequence of OS/8 (backspace, space, backspace) does not function correctly, it could be the case that your terminal uses a non-standard code for backspace. Your system manager can arrange that MULTI8 translates backspace to the proper code for your terminal. See also the LEFT command, chapter

5.

- If you have difficulties with entering ESCAPE or ALTMODE, check with your system manager. MULTI8 can be instructed to recognize any code as ESCAPE. See also the ESCAPE command, chapter 5.

5. Console operations

Because the timesharing user has no acces to the computers console switches, there must be some alternative method to control the operation of the user's virtual machine. This mechanism is activated by typing CONTROL/B. The system will respond with ^B,CR,LF,B>. Now you may give one of the following commands (keywords may be abreviated to a single character):

AC 1234	Set AC to 1234 (octal).
BOOT	Bootstrap the OS/8 system on the virtual machine.
CONTINUE	Continue exection of your program.
DELAY 3,215	Insert 3 fillers after each CR (=215).
ESCAPE 376	Convert code 376 (ALTmode) to escape (233).
FIELD 12	Set instruction field to 1 and datafield to 2.
HOOK 3	Disconnect from current bg and hook the terminal to background 3. Output from your current background will still be send to your terminal.
KILL	Restart OS/8 at 07600.
LEFT 225	On output, convert backspace (210) to 225. (Some terminals use a non-standard code for backspace). For hardcopy terminals set LEFT 334 to convert backspace to backslash.
PC 200	Set the program counter to 200.
RESTORE	Rebuild the OS/8 system on the user's disk.
SWITCH 10	Set the virtual switch register to 0010. This is the value obtained when an OSR or LAS instruction is executed.
WHERE	Print the current status of the background program (see below).

When your program executes an illegal instruction (eg. HLT), the system produces a status display identical to that produced by the WHERE command:

```
PC=01230 AC=10000 DF=0 MQ=1300 GT=0 TRAPPED 7402 (HALT)
```

PC=instruction field (first digit) and program counter, AC=link (first digit) and accumulator, DF=data field, MQ=multiplier quotient register, GT=greater-than-flag (only if the machine has EAE). 'TRAPPED' is the last instruction trapped by the memory management unit.

Next the system enters CONTROL/B mode and any of the above commands can be issued.

Examples:

```
^B
B>KILL
.
```

```
^B
B>WHERE
PC=01210 AC=00000 DF=0 MQ=0000 TRAPPED 6031
B>CONTINUE

.ODT

200/ XXXX 7402
200G

PC=00201 AC=00000 DF=0 MQ=0000 TRAPPED 7402 (HALT)
B>KILL
.
```

6. Differences between MULTI8 and OS/8

The MULTI8 virtual PDP8 is not entirely compatible with the real machine. This results in a few patches to system programs. Normally these will have been installed by your system manager. Most of the patches are installed by running the batch PATCH.BI. This file will run the FUTIL program and apply a set of patches to various system programs. In a few instances, however, this procedure can not be followed, eg. for changes in the Fortran libraries.

Fortran II

The Fortran II I/O package (UTILTY.RL) has a silly method to wait for the keyboard flag. After the KSF it jumps back to a routine that looks for a CONTROL/C in the keyboard buffer (although they know that there is no charachter !). This causes any Fortran II job that waits for terminal input to be continually active. The GENIO routine was modified to read KSF;JMP .-1

A number of devices are 'kicked' when a Fortran program starts. In this way the papertape reader, puncher and lineprinter are activated. This causes every Fortran program to claim all these devices, even when it does not use them! This is cured by inserting a SM8 (Skip-on-MULTI8=6254) in the code and jump over these instructions when running under the timesharing system. An addapted version of UTILTY.RL (and a complete LIB8.RL) are distributed with the MULTI8 system.

The file PATCH.BI contains a patch to LOADER.SV. Note that this patch is mandatory and should also be applied to any existing save images of Fortran II programs. The patch changes a CDF CIF into CDF. Without this change, programs wil fail in an unpredictable and irreproducible way.

BATCH

On systems configured with 12K or larger backgrounds the BATCH program can be used from any terminal. MULTI8 will treat a batch as a single program and does not release devices between jobsteps. An optional patch to BATCH.SV (which is in PATCH.BI) changes BATCH so that the batch log is by default sent to the terminal. The new option /L should be given to get the log on the lineprinter.

Floating Point Package (EAE, 23 bit)

The keyboard input routine of this package doesn't work under MULTI8. Patch it:

```
.GET SYS PROG              /PROGRAM WITH EAE PACKAGE
.ODT

6350/ 6032 1376
6351/ 1376 6034
6352/ 6034 3053
6353/ 3053 6032
^C

.SAVE SYS PROG
```

FOCAL

As Focal uses the interrupt mechanism, it will not run unmodified. Because the patches are tedious to install with ODT or FUTIL, a PAL8 sourcefile is supplied which contains all patches (FOCIOF.PA). This relates to the FOCAL '69 version. Proceed as follows:

```
.PAL FOCIOF.PA
ERRORS DETECTED: 0
LINKS GENERATED: 0

.R ABSLDR
*FOCAL69.BN/S              (FOCAL69 + INIT)
*FOCIOF.BN
*8KOLAY.BN$
.SA SYS FOCAL
```

Fortran IV

It is NOT possible to use the FPP12 or the FPP8A floating point processor in the background of MULTI8. A patch has been made to the Fortran IV runtime system (FRTS) so that it can run in the MULTI8 background. This version executes with or without EAE. The patch (FRTSXX.PA) applies to FRTS V4. The patched FRTS can still run stand-alone (even with FPP). To install this patch proceed as follows:

```
.PAL FRTSXX
ERRORS DETECTED: 0
LINKS GENERATED: 0

.R ABSLDR
*FRTS.SV/I
*FRTSXX$

.SA SYS FRTS
```

Futher the library routine CHARS was coded so bad as to voilate the rules for the virtual memory system. The MULTI8 distribution kit contains a replacement file CHARS.RL that should be included in FORLIB.RL, eg.:

```
.R LIBRA
*CHARS/R
```

Also, an addapted version of the FORTRAN IV plotter routine (XYPLOT.RL) is supplied.

7. PAL8 programming for MULTI8.

If you who want to write assembler programs for the MULTI8 background you should observe the following points.

- Keep in mind that IOT instructions have to be emulated and therefore take much longer than on a real machine. By keeping these instructions out of tight loops, you may assure that your program is not slowed down noticably. This does not apply to CDF, CIF, RDF and RIF instructions which are handled by the hardware of the memory management unit. Many programs use a sequence with KSF/KRS to test for CONTROL/C or other special characters in the input buffer. This is perfectly legal, but remember that the overhead is larger when running in the background of MULTI8 than on a bare machine. So don't test too often.

- Note that all instructions after a CIF but before the first JMP or JMS are executed with the interrupt system shut off. Consequently no IOTs can be emulated in this position. Preferably don't put anything in between:

```
        Bad:                            Good:

        CDF CIF 10                      CDF 10
        TAD I X                         TAD I X
        DCA Y                           DCA Y
        .....                           .....
        CIF 20                          CIF 20
        JMP I A                         JMP I A
```

Depending on the actual memory allocation during execution of these instructions, the CIF 20 may be trapped (if te user's virtual field 2 is not loaded) or not. In the bad example, if the CIF 20 is trapped, the trap interrupt is not honoured until after the JMP I A. Thus the program jumps into field 1 (the pending instruction field still in effect) and then an error message is displayed. (MULTI8 will probably detect the error because of the difference between the contents of the trap-register in the memory management unit and the memory location the user's program counter is pointing to).

- Keyboard IOT'S are emulated in the following way:

```
6030 KCF  Advance pointer in input buffer.
6031 KSF  Skip if one or more characters are available  in  the  input
          buffer.   If   the   inputbuffer   is   empty   AND   the  next
          instruction is JMP .-1,  then  the  background  is  declared
          inactive until new input has arived.
6032 KCC  Clear AC; advance pointer in input buffer.
6034 KRS  Ors the current input character in AC 04-11.
6035 KIE  No-op.
6036 KRB  Loads the current input character in the users AC.  Then the
          pointer in the input buffer is advanced.
```

- A good way to read keyboard characters is:

```
        KSF
         JMP .-1            /SHOW YOU ARE WAITING FOR IT...
        KRB                 /THERE IS YOUR CHARACTER !
```

- To test for CONTROL/C use:

```
        CLA                 /AC ZERO OR 200 (FOR FORCED PARITY)
        KRS                 /READ CHAR, BUT LEAVE IT IN THE BUFFER
        TAD (-203
        SNA CLA             /CONTROL/C ?
         KSF                /FLAG UP TOO ?
        JMP NOBREAK         /NOT CONTROL/C
        JMP I (7600         /YES, HE WANTS TO STOP IT
```

- Teleprinter IOT'S are emulated as follows:

```
6040 TFL  No-op.
6041 TSF  Is changed in SKP.  Do not use it as a constant!
6042 TCF  No-op.
6044 TPC  Prints a character from AC 04-11.
6045 TSK  No-op.
6046 TLS  Prints a character from AC 04-11.
```

- To print a character one of the following will do:

```
        TAD CHAR                         TAD CHAR
        TSF                              TLS
         JMP .-1                         TSF
        TLS                               JMP .-1
        CLA                              CLA
```

N O T E

The TSF;JMP .-1 sequence is superfluous in MULTI8; However, it insures that your program will work outside the timesharing system also.

- When using OS/8 handlers you might notice that many devices have their handlers coresident with SYS:. So you need less disk-reads and less corespace to hold the handlers.

- The papertape punch is supported for both ASCII and binary data. Any PSF (6011) instruction encountered is changed to SKP. The puncher driver will patiently wait till the puncher is switched on. If the puncher is turned off while punching, it will discard the rest of the file.

- The papertape reader emulator initially patches RSF (6011) to SKP. After a reader timeout (and 3 retries) it changes the SKP to NOP. This insures 1) full speed emulation during the reading of the tape; 2) fast completion of the background's timeout loop. Many existing programs try to timeout the reader with a loop containing a RSF-instruction. If you want to read a second tape with the same reader routine, you have to restore the RSF first. This is transparent for normal OS/8 operation, eg. reading papertape with PIP.

- The lineprinter is emulated in two ways; one way is through the OS/8 handler, which passes a full buffer to the foreground, and the other way is through direct lineprinter IOT's. The OS/8 handler is the fastest way. The lineprinter IOT's transfer only one single character per trapped instruction. Note that the LSF instruction (6661) is replaced by a SKP to speed up processing. Output via lineprinter IOT's may be finished with a CONTROL/Z (232), which outputs any characters left in the internal buffers and releases the lineprinter. When sent through the LPT: handler, CONTROL/Z merely signals end-of-buffer. This was necessary for certain user programs, eg. the MINBOL system. Lineprinter output is spooled through a diskfile (DSK0:SPOOL.LP). (Systems that are very tight on foreground memory can disable spooling at configuration time). If spooling is enabled, there should be a file SPOOL.LP on the system disk. If the spool file is not found, an emulation error (for character emulation) or handler error (for LPT: output) will result.

- The plotter (XY8E) is supported by the plotter emulator task, which emulates the normal IOT's for the unencoded plotter. All plotter directives are accumulated in buffers that are writen in a diskfile (SPOOL.PL on DSK0:). As soon as the first block is enterred in the file, a second task begins to read the file and to send the data to the plotter. In this way your program is not held up by the slow plotting device, which may still be plotting long after your program was finished. An emulation error results if the file SPOOL.PL is not found.

- Floppy disks are supported by a multi-function driver, that operates with single- and double- density, single- and double- head drives (eg. RX01, RX02 and RX04). The proper mode is automatically selected, depending on drive and formatting of the medium.

- The cardreader is supported through the fakehandler mechanism. Consequently the SET CDR: 026/029 command is inoperable.

8. IOT-list for MULTI8 background

6000	Call block driver emulator. Used by the fakehandler to pass parameters from a handler call to the foreground.

```
TAD (00DU           /D=DEVICE TYPE, U=UNIT NUMBER
6000
JMP .+4             /JUMP OVER PARAMETERS
   FUNCTION         /JUST LIKE OS/8 HANDLER CALL
   BUFFER           /
   BLOCK            /
RETURN              /AC=0 OR 4000 (=ERROR)
```

6001	ION; Invalid instruction for MULTI8
6002	IOF; A no-op for MULTI8
6003-6005	Error
6006	SGT; If EAE is present, Skip on Greater-Than flag, else no-op.
6007	CAF: clears AC and Link
601X	Reader IOT's
602X	Puncher IOT's
603X	Keyboard IOT's
604X	Teleprinter IOT's.
6050-6177	Error
6200	CDTOIF; Change datafield to the current (virtual) instruction field.
62N1	CDF N; Change data field to field N if field N is available; Otherwise no-op.
62N2	CIF N; Change instruction field to N if field N is availble; Otherwise no-op
62N3	CDF CIF N
6254	SM8; Skip-on-MULTI8
6264	Look-into-real-memory; Delivers a word from the real memory into the users AC. This IOT should be followed by a CDF to the real field that must be looked into. The address within that field is specified in the AC.
650X	Plotter IOT's.
666X	Lineprinter IOT's.

6770 Giant IOT; AC specifies function:

0	Get time of day in AC: hh.mm
1	Get terminal number in AC: 000n
2	Disable keyboard echo
3	Enable keyboard echo
4	Invoke the TAlk task.
5	Used for the OPEN/CLOSE mechanism
6	Stall the program for n seconds. Use: TAD (6 / 6770 / JMP .+2 / n
7	Reset the user's account registers
10	Read the user's account registers in AC and MQ. The result is a double precision integer that gives the approximate cpu-time used since the last reset (giant IOT 7) in units of .1 second.
11-17	Reserved for system expansion
20-7777	Extendable; No system functions assigned

6771-6777 Error