FIELD SERVICE SOFTWARE ASSISTANCE BROCHURE

O S - 8

Prepared by:

Rick Moore

INTRODUCTION

This is a reference document for Field Service use in diagnosing system problems. The information presented in this chapter is extracted from various OS-8 reference manuals and literature which are listed in the appendix.

It is not intended to be a course on OS-8 or a substitute for standard DEC software manuals. However, the topics presented here are some of the more commonly used features and some of the commands that should help you to better serve your customer through an enhanced knowledge of his OS-8 operating system.

It is strongly recommended that you be extremely careful using your customer's software as a little knowledge is a dangerous thing. If you do work on a customer's files, be sure to use copies of his system only. Tell him what you feel the problem is and let him take any corrective action.

OS-8 consists of an executive and library of system programs which reside on a mass storage device called a "system device". This could be a disk, dectape or floppy unit. The executive supervises the overall program processing and consists of the following four major components:

- Keyboard monitor (KMON)
- Command Decoder (CD)
- Device Handlers
- User Service Routine (USR)

The Keyboard Monitor provides communications between the user and the OS-8 Executive. The root segment of the monitor resides perminently in memory and occupies 256 words. Commands from the console terminal are interperted and whenever needed, the Command

Decoder, USR, KMON, and ODT are brought into memory in a series of overlays while the contents of the memory used is swapped out to the system scratch area. In this manner it is possible to run a baseline system in only 8K words of memory.

The commands that are executed directly from the KMON are:

ASsign - assigns a new, user defined device name to a perminant device.

DEASsign - restores perminant device names

GET - loads core image files into memory from a device

START - starts execution of a program already in memory

RUn - loads core image (.SV) files into memory from a device and starts execution

R - loads core image (.SV) files into memory from the SYStem device and starts execution

SAve - creates core image programs on a device

ODt - invokes the Octal Debugging Technique

DAte - sets / prints the system date

The Command Decoder is called when a system program needs additional information from the operator concerning files to be acted upon or devices to be used. You are in the Command Decoder whenever the prompt character "*" is issued.

Understanding the Command Decoder line format is probably the most important step to successful usage of the OS-8 Operating System. It is as follows:

*OUTDEV:OUTFIL.EX<INDEV:INFILE.EX,INFILE.EX/SW1/SW2=N

... to translate ...

OUTDEV: - the output device (SYS:,RKB0:,DTA0:,LPT:)
OUTFIL.EX - the name of the output file and it's extension

INDEV: - the input device
INFILE.EX - the name of the input file and the extension

SW? - an optional switch ... differint for each program
=N - numerical value (ie: in direct =N# of columns)

There may be 0-3 input files, 0-9 output files, 0-3 option switches, depending on the requirements of the individual program. In several programs, it is not necessary to specify every detail ...

certain programs have default values for devices, switches, etc. The only way to be sure of this is to know the program you are working with.

Again, the value of learning the command decoder format cannot be overstressed ... the same basic format is used thoughout DEC software. (ie: RT-11, RSTS, etc.)

Device Handlers are subroutines which handle data transfer to and from peripheral devices. Device handlers for the particular hardware system you are using are activated by the program "BUILD". These subroutines are then available to all OS-8 programs. It is through the use of device handlers that a sort of "device independence" is acheived.

The USR is a collection of subroutines that perform operations of opening and closing files, loading device handlers, program chaining, and calling the CD. The USR provides these functions not only for the system itself, but for any programs running under OS-8.

INITIALIZING OS-8

Mount the OS-8 System on the appropriate hardware and execute the bootstrap. This may be as simple as pressing a "BOOT" button or require a toggle-in program from the operator's console. Various bootstraps are listed in the appendix.

After the bootstrap is executed, the monitor will print a dot "." on the console. At this point, KMON is active and the system is ready to accept one of the monitor level commands listed on page 1.

CCL

The Concise Command Language ia a means of providing an easy to understand "English-like" command interface between the operator and the monitor. It translates the commands, chains to the appropriate system programs, passes arguments and filespecs, and starts execution. Any CCL command can be simulated by typing the necessary monitor and CD responses manually. For example:

.DIR	is equivalent to	.R DIRECT *TTY:<SYS:/E=2
.COPY RXA1:<RKA0:	is equivalent to	.R FOTP *RXA1:*.*<RKA0:*.*

The use of CCL is optional in OS-8; manditory in OS-78. This is because you cannot normally access monitor level commands in OS-78. There is a way around this; by typing ".SET SYS OS8" it is possible to turn on this attribute.

GETTING DIRECTORIES

.DIR	- types directory of system device on the console
.DIR-L	- prints directory of system device on line printer
.DIR RKB0:-L	- prints directory of device RKB0 on line printer

SET

The program "SET.SV" is called by the ".SET" CCL command and can be used to modify some of the attributes of the various device handlers. Some useful ones are:

.SET SYS OS8	- allows monitor level commands in OS-78.
.SET SYS OS78	- normal OS-78 mode; CCL commands only
.SET SYS INIT	- causes the command file "INIT.CM" to be executed upon boot-up.
.SET SYS INIT [cmd]	- causes the command [cmd] to be executed upon boot-up.
.SET SYS NO INIT	- negates the above
.SET LPT LA78	- allows PDP-8A (M8316) parrallel port as controller for LA-180 (device code=0570)
.SET LPT LA8A	- restores LPT to normal 0660 device code
.SET TTY SCOPE	- causes a rubout to actually erase the character from the screen.
.SET TTY NO SCOPE	- causes the rubout to echo the character/slash combination for hardcopy terminals.
.SET TTY PAUSE	- causes the terminal to pause every "height" lines for ease of reading.
.SET TTY HEIGHT 24	- tells the handler the screen has 24 lines
.SET TTY WIDTH 80	- allows 80 character width to VT52 screen

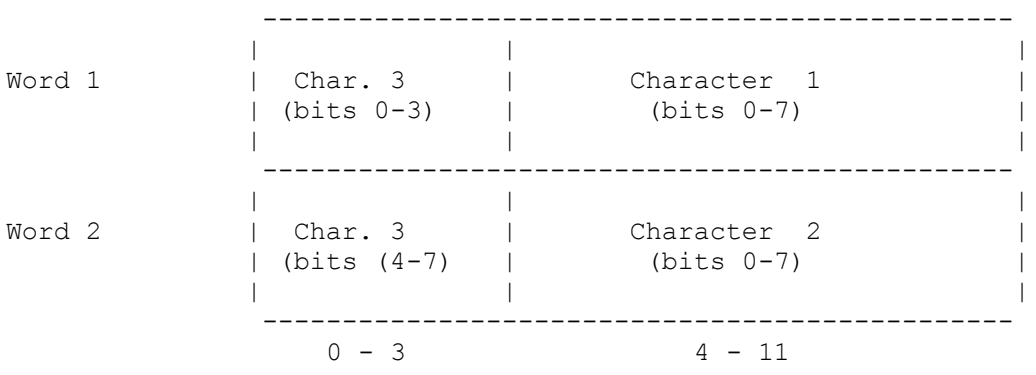
*** WARNING ***

The command ".SET SYS NO INIT" must be given before attempting to use the "BUILD" program, since execution of the "INIT.CM" file or any CCL command will cause the previous contents of memory and the system scratch area to be altered. Also, since the process of executing this file may entail several overlays, the usefullness of it on any media except disks or diskettes is questionable.

OS-8 FILES

There are three standard types of OS-8 file formats used by OS-8 and associated system programs; ASCII, Binary and Core Image (.SV).

ASCII and Binary files are packed three characters to every two 12 bit words as follows:



In Binary files, the binary data must be preceded by one or more frames of leader/trailer code (ASCII 200 code). The first character must be either 100-177 octal (origin setting for absolute binary files), 240-257 octal (a COMMON declaration frame for relocatable binary files), or 300 octal which is an origin setting. The end of binary data is one or more frames of leader trailer code. ASCII and Binary files are terminated by a CTRL/Z code (ACSII 232).

A Core Image file consists of a header followed by the actual core image. The header is called the Core Control Block (CCB). It consists of first 128 words of the first block (256 words) of the file. The CCB is a table of information that contains the length of the file, the program's starting address, and the Job Status Word (JSW). The CCB for a program at the time it is loaded into core is always saved in words 200-377 (octal) of block 37 on the system's scratch area. It is placed there by the GET or RUN operations or by the ABSLDR program. This information is then used when performing a SAVE without arguments.

FILE NAMES and EXTENSIONS

Files are referenced symbolically by a name of up to six characters followed, optionally, by a period and a two character extension. The extension to a file name is generally used to type the files according to their formats or defaults to particular system programs.

In most cases, the user will want to conform to the standard file name extensions established for OS-8. If the extension is not specified for a files, some system programs will append an assumed extension by default. This default extension can be overidden by stating the extension in the CD line explicedly.

The most common OS-8 standard default extensions are:

SV	- core image programs	
BN	- binary programs	
PA	- PAL8 assembly language sources	
BA	- BASIC source files	
BI	- BATCH control files	
SY	- system heads	
DG	- diagnostic program	**
BX	- DECX8 exercisor modules	**
X8	- customised DECX8 system exercisor	**

** refers to psuedo standards imposed by the New England District PDP-8 Diagnostic System.

FILE MANIPULATIONS

Files can be transfered between OS-8 devices by using either "FOTP.SV" (File Oriented Transfer Program) or "PIP.SV" (Peripheral Interchange Program) or the CCL commands that call them. These commands and their uses are:

COPy	- copies files from one device to another
DELete	- deletes files from a device
LIst	- lists an ASCII files on the LPT
RENname	- changes the name of a file in the directory
SQuish	- eliminates all empty and deleted files on a device
TYpe	- types an ASCII file on the console terminal
ZERO	- zeroes the directory of a device, deleting all existing files on that device

For example, to copy file FOO.PA from RXA1: to RKB0:

```
.COPY RKB0:<RXA1:FOO.PA
```

(note the similiarity between this and the format of the Command Decoder Line on page 2.)

To copy an entire device using PIP and FOTP type:

```
.R PIP                          first transfer the system heads
*RKA1:<RKA0:/Y/Z
*RKB1:<RKB0:/Y/Z

.R FOTP                         now all the files
*RKA1:<RKA0:*.*                 using wildcard transfers
*RKB1:<RKB0:*.*
```

For floppies, it is easier since the program "RXCOPY.SV" or the CCL command ".DUPLicate" will copy the entire device.

SYSTEM HEADS

The "/Y" option of PIP.SV can be used for system head manipulation. A system head is 50 blocks long and is located at the beginning of the system device. It contains the system bootstrap, the monitor, and all its overlays. System heads are sometimes stored as files on a device for ease of use with the extension ".SY".

To move a system head:

```
.R PIP
*RKA1:<RKA0:/Y
```

if "RKA1" is a virgin device, a skeleton directory must be created, so type:

```
*RKA1:<RKA0:/Y/Z
```

USING WILDCARDS

Certain commands allow wildcards in the file name specifications. These commands are COPY, DELETE, DIRECT, LIST, RENAME, and TYPE.

Wildcards allow a filename or extension to be totally replaced with an asterisk (*) or partially repalced with a question mark (?). Wildcards are particularly useful when doing multiple file transfers. This is illustrated in the following examples:

TEST1.*	- all files with the name TEST1 and any extension
*.BN	- all files with a BN extension and any name
.	- all files with any name and any extension
TEST2.B?	- all files with the name TEST2 and any extension starting with a B
TES??.PA	- all files with a PA extension and any name from 3 to 5 characters beginning with TES

the asterisk amd question mark can be used together

???.*	- all files with any extension and with file names of three characters or less

BUILD

The program "BUILD.SV" is the system generation program for OS-8 which allows the user to:

1. Maintain and update device handlers in an existing OS-8 system.

2. Add device handlers to a new or existing system.

3. Change the system characteristics to reflect hardware modifications.

Device handlers are supplied with the OS-8 system and are in the format of Binary (.BN) files.

To run the Build program, type:

```
.RUN SYS BUILD
```

NOTE:

It is important that the user specify the "RUN" command, rather than the "R" command when loading BUILD into core. This will allow the CCB to be stored on the system scratch area for use with the SAVE command!

Also, since the system scratch area is used at times when the "INIT.CM" file is executed, it is imperative that the command ".SET SYS NO INIT" be given prior to beginning the BUILD process. After the program is SAVEd, ".SET SYS INIT" can be reinstituted.

This matter cannot be overemphisized ... SAving "BUILD" without these precautions will appear to work and the system will boot without failure, however, the next time BUILD is run, it will not work! This is because the proper CCB parameters will have been lost.

RUNNING BUILD

Changing a system's parameters can be a complicated procedure with too many variables to be explained in this document. For complete instructions on BUILD, see the appropriated OS-8 Manual.

- DEVICES

Some of the devices that are available in the OS-8 system are:

SYS:	-	the system device
DSK:	-	the default device (usually SYS:)
RKA0:	-	"A" side of RK05 drive #0
RKB0:	-	"B" side of RK05 drive #0
TTY:	-	console teletype
LPT:	-	line printer
LQP:	-	letter quality printer (Diablo)
PTR:	-	paper tape reader
PTP:	-	paper tape punch
DTA0:	-	dectape drive #0
DTA1:	-	dectape drive #1
RXA0:	-	RX01 floppy drive #0
RXA1:	-	RX01 floppy drive #1
RL0A:	-	RL01 area "A" drive 0
RL0B:	-	RL01 area "B" drive 0
RL0C:	-	RL01 area "C" drive 0

FUTIL

The OS-8 File UTILity program was orginally developed by Jim Crapuchettes of Menlo Computer Associate,Inc., Menlo Park, CA. It is now included within the OS-8 Extension Kit after release V3D, and is also contained on the New England District PDP-8 Diagnostic System after release V8.04.

FUTIL enables a user to examine and modify the contents of mass storage devices, including the SYStem device, and is therefore as dangerous as it is useful! It can be used to recover damaged files, repair destroyed directories, patch programs on the mass storage device, and check for bad blocks on the device.

It is this last feature that is particularly useful to us in Field Service as we often have questions concerning the validity of a customer's disk or diskette. The following procedure can verify this:

```
.R FUTIL
```

SET DEVICE RXA0	floppy drive 0 is the target device
SCAN 0-755	scans the 756 (octal) blocks on the device
EXIT	returns to OS-8

If any bad blocks were encountered, the program would report

BAD BLOCK nnn	where "nnn" was faulty

The cause of this could be due to either a bad CRC character due to blown data (can be recopied) or blown format (device must be reformatted). In the case of floppies, the later condition is fatal as they cannot be reformatted.

FUTIL can be used to scan devices regardless of their program contents, for example, OS-8, COS 300, or any customer written file structure can be scanned, as long as standard sector format is used.

to SCAN an RK05 Disk:

```
.R FUTIL

SET DEVICE RKA0                 or RKA1, etc.

SCAN 0-6257                     there are 6260 blocks

SET DEVICE RKB0                 now the other side

SCAN 0-6257

EXIT

.
```

NOTE:

At this time, a note on the "sides" of a disk is in order. The RK05 is broken up into two (2) logical devices called "RKA0" and "RKB0" as the RL01 is broken into three (3) devices called "RL0A", RL0B", and "RL0C" by OS-8. This is because of the system's inability to address the entire disk and the limited number of entries the directories can hold.

These "sides" are actually not physical sides (who ever heard of a three sided RL01), but refer (on the RK05) to the first 6260 sectors as "RKA0" and the remaining 6260 sectors as "RKB0". A look at the hardware shows that after the sector counter overflows (at 16 sectors) the heads switch to the other surface. After the sector counter overflows again, the heads switch back and then the cylinder address increments. Thus RKA0 is on the outer tracks of the disk and RKB0 is on the inner tracks.