RSX-11M VERSION 4.0
FW DRIVER AND UTILITY
REFERENCE MANUAL

Scientific Micro Systems, Inc.

RSX-11M VERSION 4.0
FW DRIVER AND UTILITY
REFERENCE MANUAL

RSX-11M VERSION 4.0

FW DRIVER AND UTILITY

REFERENCE MANUAL

DOCUMENT NO.: 3000669
REVISION: D
DATE: MARCH 15, 1983

CONTENTS

I. INTRODUCTION.....I - 4
A. FW: DRIVER DESIGN AND IMPLEMENTATION.....I - 5
B. 22-BIT ADDRESSING.....I - 5
C. FW: LIMITATIONS.....I - 5
D. CARTRIDGE MAGNETIC TAPE (MS:).....I - 5

II. INSTALLATION OF THE FW: DRIVER AND UTILITY.....II - 6
A. DISTRIBUTION DISKETTE CONTENTS.....II - 7
B. SUPPLIER AND USER CONSIDERATIONS.....II - 8
C. DRIVER CONDITIONALS.....II - 8
D. INSTALLATION OF FW: DRIVER.....II - 9
E. SYSGEN CONSIDERATIONS.....II - 12

III. COMPLETE RSX11M DISTRIBUTION - FLOPPY DISK.....III - 14
A. DISKETTE DISTRIBUTION.....III - 14
B. INSTALLATION OF RSX11M DISTRIBUTION.....III - 15
C. SYSTEM GENERATION.....III - 18
D. SYSTEM DOES NOT BOOT.....III - 20

IV. COMPLETE RSX11M DISTRIBUTION - CARTRIDGE TAPE.....IV - 21

V. FW DISK UTILITY (FWU).....IV - 22
A. FWU COMMAND ENTRY.....IV - 22
B. FWU FUNCTION SUMMARY.....IV - 23
C. FWU FLOPPY DISK FUNCTIONS.....IV - 25
D. FWU WINCHESTER DISK FUNCTIONS.....IV - 28
E. FWU GENERAL PURPOSE FUNCTIONS.....IV - 31
F. FWU UTILITY EXAMPLES.....IV - 35
G. FWU MESSAGES.....IV - 37

V. FW: DRIVER FUNCTIONS.....V - 42
A. THE GLUN$ (GET LUN INFORMATION) MACRO.....V - 42
B. STANDARD QIO FUNCTIONS.....V - 43
C. DEVICE-SPECIFIC QIO FUNCTIONS.....V - 45
D. QIO FUNCTION MODIFIERS.....V - 48
E. I/O AND RELATED ERROR CONDITION PROCESSING.....V - 49
F. DRIVER ENTRY POINTS.....V - 52
H. PROGRAMMING CONSIDERATIONS.....V - 53

I. INTRODUCTION

The SMS FWD controller can be used with two floppy disk drives in COMPATIBLE mode using the RX02 driver (DYDRV) supplied by DIGITAL with the RSX-11M Operating System. In this mode, the device will perform exactly as an RX02. However, in order to take advantage of the significant functional and performance advantages of the FW series of controllers when operated in EXTENDED mode, it is necessary to use the SMS-supplied FW driver (FWDRV). With the inclusion of FW:, the RSX-11M user has the following advanced capabilities:

* Ability to read and write RX01, RX02, and RX03 diskette formats on a per unit basis in the same manner as the RX02 driver.

* Ability to read and write IBM single sided and double sided diskettes on a per unit basis for all single density and double density formats.

* Ability to read and write one or two Winchester fixed disks using the same driver as the floppy disk units.

* Ability to DMA transfer up to 32K words of data in a single I/O operation without incurring any Executive overhead, in any of the above floppy and Winchester formats.

* Ability to use IBM formatted diskettes as FILES-11 volumes, allowing 1216 blocks of storage per diskette with single sided drives, or 2432 blocks of storage with double sided drives (compared with 988 blocks allowed using the RX02 format).

* Ability to format diskettes in any of the IBM or DEC data formats described above.

* Ability to use the Winchester or IBM format floppy units as the system device (that is, the FWD controller can boot RSX-11M from power-up without another FILES-11 device present on the user's system).

* Ability to dynamically change the diskette format either by an application program or by using a supplied utility program (MCR function).

* Support for 22-bit addressing with the appropriate hardware configuration.

A. FW: DRIVER DESIGN AND IMPLEMENTATION

The FW: driver was designed to provide for all of the RSX-11M sysgen options supported by DEC-written mass storage device drivers by means of standard conditional assembly parameters. Applicable options are 11/70 Unibus mapping, LSI-11 processor, error logging, memory management, patient power fail recovery, and multiple controllers.

In addition, the inherent capabilites of the FWD controller in extended mode were exploited to the fullest, resulting in an assembly size shorter than would otherwise be possible. Specifically, for device-independent I/O functins (IO.RLB and IO.WLB), the controller hardware performs the logical block to track/sector conversion, the interleave mapping, the error retry policy handling, end-of-volume detection, illegal block number detection, and track re-seek handling.

The coding standard followed was that of DEC's RSX-11M development group (a modified version of that promulgated in Appendix E of the MACRO-11 Reference Manual). If a detailed understanding of the functional aspects of this driver (or any others on the user's system) is required, the RSX-11M Guide to Writing an I/O Driver can be used as a reference. This manual contains an appendix defining the symbolic terms used in the driver source code.

B. 22-BIT ADDRESSING

RSX-11M V4.0 supports 22-bit extended memory. It is possible to configure a system with 1/2 megabyte or more of memory. Contact SMS for further details. The complete RSX-11M V4.0 distribution from SMS contains a bootable floppy that supports 22-bit addressing.

C. FW: LIMITATIONS

The FW driver supports all standard DEC utilities with the following exceptions:

> Bootable disk save and compress (DSCS8.SYS) is not supported. Disk backup is performed by the disk save and compress task (DSC.TSK), BRU.TSK, or FWU.TSK.
>
> Bootable PRESRV.SYS or FMT.TSK are not supported for the FW driver. Use FWU.TSK or the SMS installation and test floppy to format disks.

D. CARTRIDGE MAGNETIC TAPE (MS:)

RSX-11M V4.0 supports the SMS cartridge tape. The tape unit is TS-11/TSV05 compatible and uses the MS device handler.

II. INSTALLATION OF THE FW: DRIVER AND UTILITY

If the complete RSX11M floppy distribution was purchased from Scientific Micro Systems, proceed to section III.

If the complete RSX-11M tape distribution was purchased from Scientific Micro Systems, proceed to Section IV.

This section describes the installation of the FW: driver and FW utility.

The distribution kit supplied is for a host system operating RSX-11M from a DEC-supported system device. The medium is one RX02-compatible diskette containing all requisite sources. This diskette is readable using the RX02 driver DYDRV and the SMS FWD0100 hardware controller, or with the DYDRV and the RX211 hardware controller.

With the host system, an RSX-11M user may add the FW driver as an auxilliary FILES-11 device to his present system or build a target system with the FW driver as the system device.

A utility program (task) is also supplied, in source form, with the distribution kit. This task (FWU) may be installed on the host or target system as an MCR function, or else invoked with the RUN MCR function. FWU provides a convenient means for dynamically changing diskette formats in order to accommodate the selective reading and writing of various IBM or DEC diskette formats, plus provides additional device control capabilities. (See Section IV.)

A. DISTRIBUTION DISKETTE CONTENTS

The distribution diskette contains the following files.

```
Directory DY0:[1,20]
SAVBLD.BLD;1                ;SAV BUILD
BOOBLD.BLD;1                ;BOO BUILD

Directory DY0:[1,24]
SAVBLD.CMD;1                ;SAV BUILD
BOOBLD.CMD;1                ;BOO BUILD
SAVBLD.ODL;1                ;SAV ODL
BOOBLD.ODL;1                ;BOO ODL

Directory DY0:[1,54]
FLX.TSK;1                   ;FLX FOR FW
CFL.TSK;1

Directory DY0:[200,200]
FWGEN.CMD;1                 ;USED TO INSTALL FW
FWINS.CMD;1                 ;BUILD FW AFTER SYSGEN

Directory DY0:[1,24]
PREFW.MAC;1                 ;FW CONDITIONAL ASSEMBLY PREFIX FILE
FWDRV.MAC;1                 ;FW DRIVER SOURCE
FWTAB.MAC;1                 ;FW TABLE SOURCE
FWU.MAC;1                   ;FW UTILITY SOURCE
FW.CMD;1                    ;BUILD FW DRIVER
FWU.CMD;1                   ;BUILD FW UTILITY
FWUBLD.CMD;1                ;FWU TKB
FWDRVBLD.CMD;1              ;FW DRIVER TKB
SAVFW.MAC;1                 ;SPECIAL DRIVER FOR SAV AND BOO
SAVBOOBLD.CMD;1             ;SAVFW TKB
SAVFWASM.CMD;1              ;SAVFW ASSEM.
CREATEWIN.CMD;1             ;SAMPLE COMMAND TO CREATE BOOTABLE WINC
CREATEFLP.CMD;1             ;SAMPLE COMMAND TO CREATE BOOTABLE FLOPPY
CREATEDIS.CMD;1             ;SAMPLE COMMAND TO CREATE DISTRIBUTION
CREATERXO.CMD;1             ;SAMPLE COMMAND TO CREATE DRIVER
RSX.CMD;1                   ;VMR COMMAND FILE
EFW11.CNF;1                 ;ERRLOG
DEVSM1.CNF;1                ;ERRLOG
EFW11ASM.CMD;1              ;ERRLOG
DEVSM1ASM.CMD;1             ;ERRLOG
FWERRINS.CMD;1              ;ERRLOG
GETDEV.OBJ;1                ;FILEX
FIXFLX.CMD;1                ;FILEX
MSDRVASM.CMD;1              ;BUILD MS DRIVER
MSTABASM.CMD;1              ;BUILD MS DRIVER TABLE
MSDRIVER.CMD;1              ;BUILD MS DRIVER COMMAND FILE
MSDRVBLD.CMD;1              ;BUILD MS DRIVER TKB

Directory DY0:[11,10]
MSDRV.MAC;1                 ;MS DRIVER SOURCE
MSTAB.MAC;1                 ;MS TABLE SOURCE
```

B. SUPPLIER AND USER CONSIDERATIONS

It is assumed that the RSX11M system supports loadable drivers. Executive space and runtime overhead required to support loadable drivers is almost zero. Disk space required for the actual support tasks LOA and UNL is minimal. For the purposes of future updates, not only of FWDRV but also of DEC-supplied drivers, the complexity of installing or changing a driver is increased at least an order of magnitude if Loadable Driver support is not included.

C. DRIVER CONDITIONALS

RSX-11M drivers contain many conditional assembly parameters which are associated with Sysgen options selected by the user or distributed by DIGITAL. These conditionals correspond to questions asked by Sysgen Phase One and should be known to the user. Care should be taken to insure that the conditional assembly prefix file used when assembling the driver agrees with that used when the Executive of the target system was assembled. If it does not, one of the following will happen:

1) The task builder (TKB) will issue error messages indicating that some global is undefined.

2) The load task (LOA or VMR) will indicate that some driver feature is inconsistent with the Executive symbol table file.

3) The system will crash when the driver is loaded or when a disk access is attempted.

D. INSTALLATION OF FW: DRIVER

The following procedure should be used to generate and install supplied software. Insert the RX02 diskette into SMS supplied floppy unit. Execute the following commands. Your responses are underlined:

```
 SET /UIC=[200,200]
>MOU DY0:/OVR
>PIP SY0:/NV=DY0:FWINS.CMD
>@FWINS
>;THIS COMMAND PERFORMS THE FOLLOWING FUNCTION:
>;        1-CREATE THE NECESSARY UFD'S
>;        2-TRANSFER ALL FILES FROM THE RX02 FLOPPY DISTRIBUTION TO
            YOUR SYSTEM DISK.
>;        3-ALLOW YOU TO EDIT THE FW CONDITIONAL FILE "PREFW.MAC"
>;        4-ASSEMBLE AND TASK BUILD THE "FW" DRIVER
>;        5-ASSEMBLE AND TASK BUILD THE FW UTILILTY "FWU.TSK"
>;        6-RUN VMR TO INSTALL NEW "FW" DRIVER AND FWU UTILITY
>;        7-TASK BUILD A NEW VERSION OF "SAV.TSK"
>;        8-TASK BUILD A NEW VERSION OF "BOO.TSK"
>* DO YOU WANT TO CONTINUE? [Y/N]:Y
>;
>;
>INS $MAC
>INS $TKB
>INS $VMR
>SET /UIC=[1,54]
>;THE FOLLOWING FILES MUST BE PRESENT ON YOUR SYSTEM DISK BEFORE
  WE CAN PROCEED.
>;
>;        1-[11,10]RSXMC.MAC    (YOUR SYSTEM CONDITIONAL ASSEMBLE FILE)
>;        2-[1,1]EXEMC.MLB      (EXEC MACRO LIBRARY)
>;        3-[1,1]RSXMAC.SML     (SYSTEM MACRO LIBRARY)
>;        4-[1,1]EXELIB.OLB     (EXEC LIBRARY)
>;        5-[1,1]SYSLIB.OLB     (SYSTEM LIBRARY)
>;        6-[1,54]RSX11M.STB    (SYMBOL TABLE)
>;        7-[1,24]SAV.OLB,BOO.OLB,MCR.OLB
>;
>;ESPECIALLY MAKE SURE THAT THE PROPER RSXMC.MAC  IS IN [11,10]
>;
>*ARE ALL THE ABOVE FILES PRESENT IN THE PROPER UIC? [Y/N]: Y
>;
>;WE ARE ASSUMING THAT THE PROPER INPUT DEVICE DRIVER IS LOADED!!!!
>* WHAT IS THE INPUT FLOPPY DEVICE NAME (I.E. DY0:,FW0:,ETC) [S]: DY0:
>DMO DY0:/DEV
>ALL DY0:

>MOU DY0:/OVR
>;
>PIP SY0:[*,*]/UF/FO/NM/NV=DY0:[*,*]*.*/FO/NM
>;
>DMO DY0:/DEV
>SET /UIC=[1,54]
```

```
>;
>;YOU MUST EDIT THE FW DRIVER CONDITIONAL ASSEMBLY FILE TO SPECIFY THE
>;TYPE OF WINCHESTER DISK THAT YOU HAVE ON YOUR SYSTEM.
>;      (THE DEFAULT WINCHESTER TYPE IS A QUANTUM 2020 WITH MULTIPLE
>;       LOGICAL VOLUMES ENABLED FOR 4 DEVICES.  IF YOUR PHYSICAL
>;       WINCHESTER IS A QUANTUM 2020,2040, OR 2080 IT IS RECOMMENDED
>;       THAT YOU NOT EDIT THIS FILE OR CHANGE ITS DEFAULTS.  A QUANTUM
>;       2040 BY DEFAULT WILL BE TWO LOGICAL 20MB DEVICES OF EQUAL
>;       SIZE.  A QUANTUM 2080 BY DEFAULT WILL BE FOUR LOGICAL 20MB
>;       DEVICES OF EQUAL SIZE).
>;
>* DO YOU WANT TO EDIT THE FW CONDITIONAL ASSEMBLY FILE? [Y/N]: N
>;IT MAY BE NECESSARY TO EDIT THE FW DRIVER TASK BUILD COMMAND FILE
>;TO CHANGE THE NAME OF THE PARTITION.
>;
>* DO YOU WANT TO EDIT THE TASK BUILD COMMAND FILE FOR THE FW
  DRIVER?  [Y/N]: N
>MAC [1,124]FWDRV=[11,10]RSXMC/PA:1,[1,1]EXEMC/ML,[1,124]PREFW,FWDRV
>MAC [1,124]FWTAB=[11,10]RSXMC/PA:1,[1,1]EXEMC/ML,[1,124]PREFW,FWTAB
>TKB @[1,124]FWDRVBLD.CMD
>;
>;IT MAY BE NECESSARY TO EDIT THE FW UTILITY TASK BUILD COMMAND FILE.
>;TO CHANGE THE NAME OF THE PARTITION.
>;
>* DO YOU WANT TO EDIT THE BUILD FILE FOR THE FW UTILITY? [Y/N]: N
>MAC [1,124]FWU,FWU/-SP=[1,1]EXEMC/ML,[11,10]RSXMC/PA:1,[1,124]PREFW,FWU
>TKB @[1,124]FWUBLD
>TIM
>;
>;YOU WILL NOW HAVE A CHANCE TO RUN VMR TO LOA THE NEW DRIVER
>;AND INSTALL THE NEW COPY OF THE UTILITY PROGRAM.  EXECUTE THE
>;FOLLOWING COMMANDS WHEN YOU ARE PROMPTED BY VMR:
>;
>;      ENTER FILENAME:RSX11M
>;                     ------
>;      VMR>UNL DY:                    (IF DY: IS LOADED)
>;          ------
>;      VMR>LOA FW:/PAR=GEN            (OR THE APPROPRIATE PARTITION)
>;          ---------------
>;      VMR>INS FWU
>;          -------------------
>;      VMR>(CONTROL Z)
>;          -       -
>VMR
Enter filename:  RSX11M
VMR>UNL DY:
VMR>LOA FW:/PAR=GEN
VMR>INS FWU
VMR>Z
>;
>;
>;IF YOU DESIRE TO CREATE A BOOTABLE SYSTEM DEVICE ON THE SMS SUPPLIED
>;DISK, YOU MUST REBUILD "SAV.TSK" AND "BOO.TSK".  IF YOU WANT
>;TO USE THE SMS SUPPLIED DISK ONLY AS A PERIPHERAL DEVICE THEN
```

```
>;ANSWER NO TO THE FOLLOWING QUESTION.  ANSWER YES TO THE FOLLOWING
>;QUESTION ONLY IF YOU ARE SURE OF WHAT YOU ARE DOING.  IT IS
>;RECOMMENDED THAT YOU BACK UP YOUR SYSTEM.
>;BEFORE RE-BUILDING SAV AND BOO.
>* DO YOU WANT TO REBUILD "SAV.TSK" AND "BOO.TSK"? [Y/N]: Y
>ASN SY0:=TK:
>ASN NL:=MP:
>SET /UIC=[1,24]
>;
>; Indirect commandfile to reassemble and rebuild SAVE and BOOT
>; - use this file when you change from/to multiple logical volume
>; support.
>* Do you want to continue? [Y/N]: Y
>ASN SY:=TK:
>ASN NL:=MP:
>MAC @[1,124]SAVFWASM
>SET /UIC=[1,24]
>TKB @SAVBLD
>TKB @BOOBLD
>SET /UIC=[1.24]
>;
>; DONE
>;
>;
>;A NEW COPY OF THE BOO AND SAV TASK ARE NOW ON YOUR SYSTEM DEVICE.
>;YOU MAY EXECUTE THE FOLLOWING COMMAND:
>;      >REM BOO
>;      >REM SAV
>;      >INS LB0:$SAV
>;      >INS LB0:$BOO
>;      >ACS SY0:/BLKS=0
>;      >SAV
>;
>;USE @[1,124]CREATEWIN.CMD OR @[1.124]CREATEFLP.CMD TO CREATE SYSTEMS
>;ON THE WINCHESTER OR FLOPPY DEVICES.
>;
>SET /UIC=[200,200]
>ASN =
>@ <EOF>
>
```

E. SYSGEN CONSIDERATIONS

If the target system's RSX11M files differ from those of the host system, the complete Sysgen procedure must be followed. Since this procedure changes from release to release, an attempt here to recap the steps outlined in the Sysgen manual would be pointless. The following notes may, however, be helpful:

1) It is possible to perform a SYSGEN specifying all drivers to be loadable. In this case, one can answer affirmatively to build many of the DEC-supplied drivers. SYSGEN will create tables for each driver specified (in the SYSTB.MAC output file). This may be desirable expecially if a number of drivers are potentially needed one at a time. However, the Executive space consumed by such a list of tables becomes quite large.

One can remedy this situation by editing the SYSTB file and saving the generated tables, then unlinking them from the list to be task built with the Executive, leaving only the pseudo devices and possibly "TT;" in SYSTB.MAC. Then the saved table file may be broken down into individual table files and used as described in the Guide to Writing an I/O Driver (see FWTAB.MAC as an example of such a table file). Following this, the DRVBLD indirect command control file can include the table files in the resultant driver task build operations which take place during Sysgen Phase Two.

This procedure was used by DEC for the creation of the RSX-11M baseline distribution systems. It is likely that future releases of RSX-11M will include the table files as separate sources so that this inconvenient procedure can be avoided.

2) Pool space is allocated permanently by the MCR or VMR LOA commands for drivers which have a table segment and whose names do not already exist in the device tables. This is because the tables must always be addressable to the Executive (in the lowest physical 16K of a mapped system). If the default pool space value suggested by SYSGEN is used, the result will be an inadequate amount of pool space left after drivers are loaded. Once a system is SAVed, the partition containing SAV.TSK cannot be re-allocated, so be sure to have an adequate amount of pool space before SAVing the target system. A ballpark figure would be 1000(10) bytes plus 120(10) bytes for each terminal/task combination to be concurrently operating after INStalling all tasks desired at boot time and LOAding all drivers required at boot time plus those required concurrently without re-booting.

3) The FW driver does not require the $BLKCK subroutine or the $GTWRD or $PTWRD subroutines. The default vector address is under 400(8). Therefore, the user written driver question does not have to be answered positively. However, if some driver requiring one of these routines is desired at a later time, it cannot be LOAded unless all referenced routines are present in the Executive. The Sysgen program asks the user about the $GTWRD and $PTWRD routines, but not about the $BLKCK routine. Therefore, we suggest that the user locate this routine to be defined only if certain DEC drivers are specified in the original Sysgen Phase 1. That is, make the routine expand always if it is expected that any driver will be added to the system at some later time.

4) The UNLoad command does not remove the tables corresponding to a driver that is unloaded, even if the driver task contains internal table definitions.

Therefore, never save a system with drivers loaded when it is intended that alternative driver/table tasks (as with the baseline systems) will be used unless you have copied the system image with PIP before doing so. Once a system is saved with a driver table, the table is there to stay.

5) When a new Sysgen is performed, execute the normal Sysgen Phase I and Sysgen Phase II. Save and boot will be built during Phase II. At the end of Sysgen Phase II, but before booting your new system, you must rebuild the FW driver and the FW utility. See Section III.D.

III. COMPLETE RSX11M DISTRIBUTION - FLOPPY DISK

If your distribution consisted of one RX02 floppy and not the complete RSX11M system, then see section II.

Digital Equipment Corporation requires a software license for each computer that contains a RSX11M system. See your local SMS or DEC representive for additional information.

A. DISKETTE DISTRIBUTION

If your distribution is on cartridge tape process to section IV.

The complete RSX11M distribution from SMS consists of the following double sided IBM formatted floppy diskettes.

	LABEL	DESCRIPTION
1.	"BOOTABLE"	;MINIMAL RSX-11M BOOTABLE MAPPED SYSTEM
2.	"BOOTABLE CONTINUATION"	;LIBRARIES,TKB,MAC,EDI, FWDRV.MAC,ETC.
3.	"SYSGEN (1 OF 4)	;[1,20]
4.	"SYSGEN" (2 OF 4)	;[1,24],[200,200]
5.	"SYSGEN" (3 OF 4)	;[11,10]
6.	"SYSGEN" (4 OF 4)	;[11,10]
7.	"HELP"	;HELP FILE[1,2]
8.	"ERRORLOG"	;ERRLOG[1,6],[104,10]
9.	"SYSGEN PHASE III" (1 OF 2)	;[1,20]*.OLB
10.	"SYSGEN PHASE III" (2 OF 2)	;[1,20]*.BLD,[2,300],[1,51]
11.	"MCR SOURCES" (1 OF 4)	;[12,10]
12.	"MCR SOURCES" (2 OF 4)	;[12,10]
13.	"MCR SOURCES" (3 OF 4)	;[12,10],[12,24],[14,10],[11,10]
14.	"MCR SOURCES" (4 OF 4)	;[23,10],DCL,[45,10]
15.	"RMS-11" (1 OF 2)	;RMS-11 [1,1],[1,20],[1,24]
16.	"RMS-11" (2 OF 2)	;RMS-11 [1,54]

B. INSTALLATION OF RSX11M DISTRIBUTION

Insert the bootable floppy into floppy unit 0 and hit the reset button. The following installation dialogue will take place: (if system does not boot, see section III.E for possible solutions). Operator responses are underlined.

```
DRV?
F0

  RSX-11M V4.0 BL32   128K   MAPPED
>RED FW:=SY:
>RED FW:=LB:
>MOU FW:FLOPPY
>@FW:[1,2]STARTUP
>* PLEASE ENTER TIME AND DATE (HR:MN DD-MMM-YY) [S]: 15-MAR-83
>TIM 15-MAR-83
>SET /UIC=[1,124]
> *DO YOU WANT TO COPY RSX11M DISTRIBUTION TO WINCHESTER DISK? [Y/N]:Y
>;YOU MUST NOW ENTER THE TYPE OF WINCHESTER DISK THAT IS ON YOUR SYSTEM.
>;THE FOLLOWING WINCHESTER DISK TYPES ARE SUPPORTED:
>;
>;      CODE    TYPE            DESCRIPTION
>;      ----    ----            -----------
>;       1      ST406           (SEAGATE 5 1/4-INCH, 5.5 MEGA BYTE)
>;       2      ST412           (SEAGATE 5 1/4-INCH, 10.6 MEGA BYTE)
>;       3      ST419           (SEAGATE 5 1/4-INCH, 15.4 MEGA BYTE)
>;       4      RO204           (RODIME 5 1/4-INCH, 21.8 MEGA BYTE)
>;       5      RO206           (RODIME 5 1/4-INCH, 32.6 MEGA BYTE)
>;       6      RO208           (RODIME 5 1/4-INCH, 43.5 MEGA BYTE)
>;      10      SA1002          (SHUGART 8-INCH, 4.5 MEGA BYTE)
>;      11      SA1004          (SHUGART 8-INCH, 8.9 MEGA BYTE)
>;      12      Q2010           (QUANTUM 8-INCH, 8.9 MEGA BYTE)
>;      13      Q2020           (QUANTUM 8-INCH, 17.8 MEGA BYTE)
>;      14      Q2030           (QUANTUM 8-INCH, 26.7 MEGA BYTE)
>;      15      Q2040           (QUANTUM 8-INCH, 35.6 MEGA BYTE)
>;      16      Q2080           (QUANTUM 8-INCH, 71.2 MEGA BYTE)
>;      20      SA4008          (SHUGART 14-INCH, 26 MEGA BYTE)
>;
>* ENTER THE CODE FOR THE WINCHESTER DISK [D R:1.-20. D:13.]: 13
>FWU FW2:/SIZE=34700/HD=4/ST=17/CY=512
>* YOU ARE ABOUT TO INITIALIZE THE WINCHESTER DISK - CONTINUE? [Y/N]:Y
>;
>SET /UIC=[1,54]
>ALL FW2:
>FWU FW2:/VE
FW2:/Verify -- continue? (Y/N): Y
CONTINUOUS VERIFY? (Y/N): N
** Begin Verification **
>BAD FW2:
BAD -- FW2: TOTAL BAD BLOCKS= 0.
>INI FW2:WINCHESTER/BAD=[AUTO]/MXF=1500/LRU=5/INF=314.
>* DID THE WINCHESTER DISK INITIALIZE CORRECTLY? [Y/N]:Y
>MOU FW2:/OVR
>UFD FW2:[1,1]
```

```
>UFD FW2:[1,2]
>UFD FW2:[1,3]
>UFD FW2:[1,4]
>SET /UIC=[200,200]
>UFD FW2:[1,6]
>SET /UIC=[1,54]
>UFD FW2:[1,7]
>UFD FW2:[1,20]
>UFD FW2:[1,24]
>UFD FW2:[1,30]
>UFD FW2:[1,34]
>UFD FW2:[1,50]
>UFD FW2:[1,51]
>UFD FW2:[1,54]
>UFD FW2:[200,200]
>UFD FW2:[2,300]
>UFD FW2:[11,10]
>UFD FW2:[11,20]
>UFD FW2:[11,24]
>UFD FW2:[11,30]
>UFD FW2:[11,34]
>UFD FW2:[12,20]
>UFD FW2:[12,24]
>UFD FW2:[1,124]
>UFD FW2:[1,120]
>PIP FW2:[*,*]/FO=SYO:[*,*]*,*/FO
>PIP FW2:[0,0]=SYO:[0,0]RSX11.SYS
>PIP FW2:[1,54]RSX11M.SYS;*/DE
>PIP FW2:[1,54]RSX11M.SYS/CO/BL:182.=FW2:[1,54]RSX11M.TSK
>ASN FW2:=SYO:
>ASN FW2:=LBO:
>INS $VMR
>VMR @[1,124]RSX
VMR -- *DIAG* -- INSTALLED TASKS MAY NO LONGER FIT IN PARTITION
SET /TOP=DRVPAR:-*
POOL=1000:4610.:04610.
>;THE WINCHESTER DISK IS NOW READY TO BOOT.
>;THIS COMMAND FILE WILL BOOT THE NEW SYSTEM. WHEN IT DOES ENTER THE
>;FOLLOWING COMMANDS: (YOUR RESPONSES ARE UNDERLINED)
>;
>;       XDT 26
>;       XDT>G
>;           -
>;       DEVICES NOT IN CONFIGURATION
>;       (ENTER A CARRIAGE RETURN)
>;       >TIM 15-MAR-83
>;       -------------
>;       >FWU FW2:/SIZE=34700/HD=8/ST=17/CY=512
>;       ------------------------------
>;       >SAV /WB
>;       --------
>;       (YOUR NEW RSX11M SYSTEM WILL BOOT AUTOMATICALLY)
>;       >ANSWER YES TO THE QUESTION "DO YOU WANT TO CONTINUE THE
>;       >INSTALLATION?"
```

```
>;
>* DO YOU WANT TO CONTINUE? [Y/N]:Y
>BOO
XDT: 32

XDT>G

RSX-11M V4.0 BL32
>TIM 15-MAR-83
>FWU FW2:/SIZE=37400/HD=4/ST=17/CY=512
>SAV /WB


 RSX-11M V4.0 BL32   128K   MAPPED
>RED FW2:=SY:
>RED FW2:=LB:
>MOU FW2:WINCHESTER
>@FW2:[1,2]STARTUP
>* PLEASE ENTER TIME AND DATE (HR:MN DD-MMM-YY) [S]: 15-MAR-83
>TIM 15-MAR-83
>SET /UIC=[1,124]
>* DO YOU WANT TO CONTINUE THE INSTALLATION OF RSX11M? [Y/N]:Y
>ALL FWO:
>INSERT FLOPPY LABELED "BOOTABLE CONTINUATION, HIT <CR>?[Y/N]:
>MOU FWO:/OVR
>PIP FW2:[*,*]/UF/FO/NM/NV=FWO;[*,*]*.*/FO/NM
>;
>;A COMPLETE RSX-11M SYSTEM NOW RESIDES ON
>;THE WINCHESTER DISK.  IF YOU DO NOT DESIRE TO PERFORM
>;A SYSGEN THEN THE INSTALLATION IS COMPLETE.
>;AT A LATER DATE YOU MAY INSTALL OTHER FLOPPIES
>;FROM THE DISTRIBUTION BY TYPING @[1,124] INSDIS.
>;
>;DO YOU WISH TO QUIT NOW?[Y/N]:Y
>DMO FWO:
DMO -- TTO:   DISMOUNTED FROM FWO:    ***FINAL DISMOUNT INITIATED***
*** FWO: --  DISMOUNT COMPLETE

>  @<EOF>
>
```

C. SYSTEM GENERATION

Refer to the RSX11M system generation manual for details concerning system generation. Sysgen Phase I and Phase II proceed without deviation from the Sysgen manual. At the end of Phase II, but before you boot your new system you must execute the following command file. Your responses are underlined:

```
>;END OF SYSGEN PHASE II

>@<EOF>

 @[200,200]FWGEN
>;THIS COMMAND PERFORMS THE FOLLOWING FUNCTIONS:
>;      1-ALLOW YOU TO EDIT THE FW CONITIONAL FILE  "PREFW.MAC"
>;      2-ASSEMBLE AND TASK BUILD THE "FW" DRIVER
>;      3-ASSEMBLE AND TASK BUILD THE FW UTILITY "FWU.TSK"
>;      4-RUN VMR TO INSTALL NEW "FW"DRIVER AND "FWU" UTILITY
>* DO YOU WANT TO CONTINUE? [Y/N]:Y
>;
>;
>* ARE YOU BUILDING FW FOR A MAPPED SYSTEM? [Y/N]:Y
>SET /UIC=[1,54]
>;THE FOLLOWING FILES MUST BE PRESENT ON YOUR SYSTEM DISK BEFORE WE CAN
>;PROCEED.
>;
>;      1- [11,10]RSXMC.MAC     (YOUR SYSTEM CONDITIONAL ASSEMBLE FILE)
>;      2- [1,1]EXEMC.MLB       (EXEC MACRO LIBRARY)
>;      3- [1,1]RSXMAC.SML      (SYSTEM MACRO LIBRARY)
>;      4- [1,1]EXELIB.OLB      (EXEC LIBRARY)
>;      5- [1,1]SYSLIB.OLB      (SYSTEM LIBRARY)
>;      6- [1,5X]RSX11M.STB     (MAPPED OR UNMAPPED SYMBOL TABLE)
>;      7- [1,2X]SAV.OLB,BOO.OLB,MCR.OLB
>;
>;ESPECIALLY MAKE SURE THAT THE PROPER RSXMC.MAC IS IN [11,10]
>;
>* ARE ALL THE ABOVE FILES PRESENT IN THE PROPER UIC? [Y/N]:Y
>* DO YOU WANT TO EDIT THE FW CONDITIONAL ASSEMBLY FILE? [Y/N]:Y
>;
>;IT MAY BE NECCESARY TO EDIT THE FW DRIVER BUILD FILE TO CHANGE THE
>;PARITION NAME.
>* DO YOU WANT TO EDIT THE BUILD FILE FOR THE FW DRIVER? [Y/N]:N

>MAC [1,124]FWDRV=[11,10]RSXMC/PA:1,[1,1]EXEMC/ML,[1,124]PREFW,FWDRV
>MAC [1,124]FWTAB=[11,10]RSXMC/PA:1,[1,1]EXEMC/ML,[1,124]PREFW,FWTAB
>TKB @[1,124]FWDRVBLD.CMD
>;
>;IT MAY BE NECCESARY TO EDIT THE FW UTILITY BUILD FILE TO CHANGE THE
>;PARITION NAME.
>* DO YOU WANT TO EDIT THE BUILD FILE FOR THE FW UTILITY? [Y/N]:N

>MAC [1,124]FWU=[1,1]EXEMC/ML,[1,124]FWU,PREFW
>TKB @[1,124]FWUBLD
>TIM
00:10:57 15-MAR-83
```

```
>;
>;YOU WILL NOW HAVE A CHANCE TO RUN VMR TO LOA THE NEW DRIVER AND
>;INSTALL THE NEW COPY OF THE UTILITY PROGRAM. EXECUTE THE FOLLOWING
>;COMMANDS WHEN YOU ARE PROMPTED BY VMR:
>;
>;       ENTER FILENAME:RSX11M          (OR @[1,124]RSX.CMD IF APPROPRIATE)
>;                      ------
>;       VMR>LOA FW:/PAR=GEN            (OR THE APPROPRIATE PARTITION)
>;           -------------
>;       VMR>INS FWU
>;           ---------------
>;       VMR>(CONTROL Z)
>;            -       -
>VMR
ENTER FILENAME:    RSX11M
VMR>  LOA FW:
VMR>  INS FWU
VMR>   Z
```

D. SYSTEM DOES NOT BOOT

If the RSX11M floppy does not boot, insert the FW installation floppy and attempt to boot it. Refer to FW Installation and Test Manual for instructions on running the tests. If the installation floppy boots then the following conditions may have caused RSX11M not to boot:

1) Bad media.

2) Not enough memory in system. (124KB minimum)

3) Memory strapped incorrectly.

4) Floppy disk unit not double sided.

5) Real time clock turned off.

6) Strappings on LSI-11/23 not correct.

7) Vector or control status register address conflict.

IV. COMPLETE RSX11M DISTRIBUTION - CARTRIDGE TAPE

The cartridge tape distribution consists of two RSX-11M tapes plus one FW Installation and Test tape:

(Tape 1) - Complete RSX-11M Distribution. (Tape 2) - BRU Tape with MCR sources and RMS-11. (Tape 3) - Bootable FW Installation and Test.

A. Installation of RSX-11M from tape.

1. Insert and boot FW installation and test tape.

   DRV(FO,WO,T)? T

2. Select command: L
   Select Winchester: WO
   Select Device: TO

3. The load process takes 15 minutes.

B. RSX-11M distribution on tape is set up with multiple logical volumes enables.

The same tape distribution is used for 20,40, or 80 megabyte disks. FW2: is the first 20Mb, FW3: is the second 20Mb, FW4: is the third 20Mb, and FW5: is the last 20Mb.

V. FW DISK UTILITY (FWU)

The FW Disk Utility (FWU) provides the capability for users to dynamically redefine media formats on a per unit basis and, in addition, implements certain general purpose utility functions not available with DEC-supplied software. FWU can be permanently installed into an RSX-11M system as an MCR function or may be invoked by using the MCR RUN command in the manner described in the RSX-11 Utilities Procedures Reference Manual.

A. FWU COMMAND ENTRY

FWU functions are requested by entering command strings through the initiating terminal. Command strings consist of an optional device specifier followed by one or more optional switches and subswitches followed by an optional comment. The general form of a command is:

```
[dduu:] [/sw] [/sw=val] [...] [;comment]
```

where: [ ] (square brackets) indicate optional entry.

dduu is a PHYSICAL device name eg., "FW" or "DK3". No logical device names can be used with FWU.

sw is a switch. Some switches require an argument, others do not.

val is a string following a switch which requires an argument. If a numeric argument is needed, val is interpreted as a decimal number.

comment is any string following the occurance of a semicolon (;) and is ignored by the command string interpreter.

A function specification consists of a slash (/) followed by a variable length switch mnemonic. The switch may be followed by an argument separated from the switch by an equal sign (=). Note that no equal sign is used in the string to separate input and output sides of a command. Instead, each function has an implied input or output condition. This is similar to one-sided PIP commands such as /DE. Note also that any number of switches may be combined on a single command line as long as they all represent operations to be performed on a single device.

B. FWU FUNCTION SUMMARY

Based on the context of the command line, a switch is either a main switch or a subswitch modifying a main sitch. Switches are parsed by the FWU command string interpreter sequentially, but are acted upon according to a hierarchy which the program considers to be appropriate. This allows flexibility when entering commands while insuring that the indicated operations do not destructively interfere with each other.

Table V-1 lists the command switches according to the hierarchy, from highest (first executed) to lowest:

Table IV.A
FWU Command Switches and Subswitches

/HE[LP]	Print command summary on initiating terminal and ignore remainder of command line. This switch must be first on the command line, if entered, and must not follow a device specification. All other command input lines must contain a device specification and cannot have this switch present.
/RX01	Set floppy disk unit to RX01 format.
/RX02	Set floppy disk unit to RX02 format.
/RX03	Set floppy disk unit to RX03 format. (Double sided)
/IBM0	Set floppy disk unit to single side, single density IBM format.
/IBM1	Set floppy disk unit to double side, single density IBM format.
/IBM2	Set floppy disk unit to single side, double density IBM format.
/IBM3	Set floppy disk unit to double side, double density IBM format.
	NOTE:
	The RX0n switches and the /IBMn switches are mutually exclusive.
/IL=n	Set interleave policy for floppy units. The value of n must be 0, 1, or 2. 0 -> no offset or interleave; 2 -> both offset and interleave. This switch may be used as a subswitch of the /RX or /IBM main switches to override the value assumed when either of the latter is specified.
/SL=n	Set sector length for IBM floppy or Winchester units. This is a main switch for Winchester units and a subswitch modifying the /IBM main switch for floppy units. Legal values for n are listed in Table V-2.
/HD=n	Set number of heads/drive for Winchester unit.
/ST=n	Set number of sectors/track for Winchester unit.

/UN[IT]=n — Set physical unit for Winchester device.

/OF[ST]=n — Set logical block offset for floppy or Winchester units. This is a subswitch of the /IBM or /RX main switch if the specified unit is a floppy disk, and a main switch if the specified unit is a Winchester.

NOTE:

The /HD, /ST, /SL, and /SIZE switches, with possible modifying /OF and /UN switches, form a complete specification of control information needed for proper operation of the Winchester.

The /IBMn switch, in conjunction with possible /OF, /SL, and /IL subwitches, form a complete specification of an IBM format floppy unit.

The /RX0n switch, in conjunction with possible /OF, /IL subswitches form a complete specification of an RX01 or RX02 format floppy unit.

/SI[ZE]=N — Set device size in blocks for any FILES-11 device unit on the system. The value of n entered is assumed to be a decimal number. This switch may be used as a subswitch of the /RX or /IBM main switch to overide the value assumed when either of the latter is specified.

/ON=n — Set any device unit logically online (n=1) or logically offline (n=0).

/WP=n — Set any device unit software write-protected (n=1) or software write-enabled (n=0).

/NA[ME]=dd — Set device mnemonic for any device on the system. The value of dd must be two alpha characters.

/DI — Display device information. The information displayed varies according to the device specified in the command.

/FO — Format diskette loaded in floppy disk unit.

/VE — Verify readability of entire device surface. May be used as a subswitch of the /FO command or as a main switch for any device on the system.

/CO — Copy entire device surface. Following the entry of the device name (to be copied), an interactive dialogue is invoked to specify the copy output device and other copy parameters.

C. FWU FLOPPY DISK FUNCTIONS

Some FWU commands are appropriate only for floppy disk units. The operation of these is detailed in this section.

/RX01, /RX02, and /RX03 -- Specify DEC-compatible diskette format

The entry of the /RX01,/RX02, and RX03 switch specifies that subsequent read/write/format operations on the referenced drive should assume a DEC-compatible diskette format. The entered switch sets the device size, interleave and track offset policy, sector length, density, heads/cylinder, and sectors/track control fields which the driver uses when performing I/O operations (see Table IV.B). The device specifier must refer to an unmounted floppy disk unit.

These switches may be modified by the /IL subswitch to override the default interleaving policy or by the /OF subswitch to provide a logical block offset when performing I/O operations. For most applications, neither of these modifiers should be used.

/IBM0, /IBM1, /IBM2, and /IBM3 -- Specify IBM diskette format

The entry of any of the /IBM switches specifies that subsequent read/write/format operations on the referenced drive should assume an IBM-compatible diskette format. The entered switch sets the device interleave and offset policy, sector length, density, heads/cylinder, and sectors/track control fields which the driver uses when performing I/O operations (see Table VI.B). The device specifier must refer to an unmounted floppy disk unit.

These switches may be modified by the /IL subswitch to override the default interleaving policy, by the /OF subswitch to provide a logical block offset when performing I/O operations, or by the SL switch to override the assumed sector length of 512 bytes.

For FILES-11 operation, the 1024-byte sector length is illegal since I/O operations must begin on a block boundary. This makes the 512-byte sector length the most efficient (highest overall data transfer rate because there is more data space relative to control and gap space). Also, the default interleaving policy of no interleaving, 1/4 track offset is the most efficient (least wait incurred by the head crossing a cylinder boundary during a transfer).

/OF=n -- Specify Block Offset for Floppy Unit (subswitch)

This subswitch specifies that subsequent Logical Block read and write operations should be performed by adding the number "n" to the starting block before initiating the operation. This feature could be used when it is desired to make a number of sectors inaccessible to standard FILES-11 I/O. FWU adjusts the device size calculated in the /IBM or /RX main switch downward by the /OF argument (n). This feature is not to be confused with reserving Track 0 of floppy units (which is automatically done by the controller when performing Logical Block I/O).

For most applications, this feature is not useful. However, the specification of /OF=1 enables the FW driver to read and write FILES-11 volumes created by the driver supplied with SMS's FT series of controllers. The FT and 512-byte FW media formats are identical, but the FT driver reserves the first block of storage for firmware bootstrap code.

/IL=n -- Set Interleave Policy for Floppy Unit

This switch is used to control the sector interleave policy and the track offset calculation to be performed for logical block read/write operations. The /IBM and /RX switches establish a default value for the interleaving control field when entered. The /IL switch must be subsequently used if the default is to be overridden. See Table IV.A for legal values of n. The device specifier must refer to an unmounted floppy disk unit.

As explained earlier, overriding the default interleaving for RX01 or RX02 diskette formats improves the performance but makes the media incompatible with the DX or DY driver (see Programming Considerations section). Changing the default interleaving for IBM formatted floppies is not useful in most applications. For a more thorough discussion of interleaving policies, see the FWD0100/FWD1100 OEM Manual.

/SL=n Specify Sector Length for Floppy Unit (subswitch)

This subswitch is used to set the sector length to be assumed by the FW driver for subsequent read/write/format operations on the referenced drive. The device specifier must refer to an unmounted IBM format floppy disk unit.

Execution of this subswitch sets the sector length control field and the device size. The value of "n" affects the device size due to the fact that there is a variable number of sectors per track depending on the sector size selected (see Table IV.B).

/FO -- Format Floppy Diskette

This function if used to initialize the control and data fields of a diskette. A diskette may be formatted in any of the DEC or IBM formats discussed above. Only the density and sector length fields are required in order to format a diskette. The interleaving, track offset, etc., pertain to reading and writing the diskette once it is formatted.

Diskettes which have been formatted may be reformatted at any time. This is called "soft sectoring". When the format operation is invoked, the user is queried as to when and whether to initiate the format operation as a safeguard against destroying diskette data. The device specifier must refer to an unmounted floppy disk unit.

The /VE (verify surface) switch should generally be used in conjunction with the /FO switch. FWU is sensitive to this combination and initiates the verify immediately following the formatting if the two command switches are entered on the same line.

The following table lists device information corresponding to the possible combinations of FWU floppy disk switches:

Table IV.B
FW Floppy Disk Parameters

Format	/SL n Value	Sec/Trk	Sides	Device Size
/IBM0	128 or 0	26	1	494
/IBM0	256 or 1	15	1	570
/IBM0	512 or 2	8	1	608
/IBM0	1024 or 3	4	1	608
/IBM1	128 or 0	26	2	988
/IBM1	256 or 1	15	2	1140
/IBM1	512 or 2	8	2	1216
/IBM1	1024 or 3	4	2	1216
/IBM2	128 or 0	44	1	836
/IBM2	256 or 1	26	1	988
/IBM2	512 or 2	16	1	1216
/IBM2	1024 or 3	8	1	1216
/IBM3	128 or 0	44	2	1672
/IBM3	256 or 1	26	2	1976
/IBM3	512 or 2	16	2	2432
/IBM3	1024 or 3	8	2	2432
/RX01	(128)	26	1	494
/RX02	(256)	26	1	988
/RX03	(256)	26	2	1976

D. FWU WINCHESTER DISK FUNCTIONS

Some FWU commands are appropriate only for Winchester disk units. The operation of these is detailed in this section. Floppy disk hardware formats are currently well defined, thus the FWU floppy disk switches were designed to perform very formalized functions. However, to provide maximum flexibility for various Winchester drive formats (present and future), FWU Winchester-specific switches allow independent settings for each device parameter used by the FW driver. Since not as much error checking can be built into the FWU utility under these guidelines, the user must be responsible for insuring that the control parameter set by these switches agree with the hardware characteristics of the actual drive being used.

/HD=n -- Specify Number of Heads for Winchester Disk

This switch is used to set the number of heads for the specified device. The value of "n" is used only for the Physical Block I/O operations. The controller handles the block number to cylinder/head/sector conversion for Logical Block I/O operations. The device specifier must reference an unmounted Winchester unit.

/ST=n -- Secify Number of Sectors/Track for Winchester Disk

This switch is used to set the number of sectors/track for the specified Winchester unit. The value of "n" is used only for Physical Block I/O operations. The controller handles the block number to cylinder/head/sector conversion for Logical Block I/O operations. The device specifier must reference an unmounted Winchester unit.

/SL=n -- Specify Sector Length for Winchester Disk

This switch is used to set the sector length for the specified device. The device specifier must reference an unmounted Winchester unit. The value of "n" specified MUST agree with that of the actual drive being used. The controller accepts only two values; 256 and 512. On entry, 1 may be used to mean 256, and 2 may be used to mean 512.

/CY=N -- Specify Number of Cylinders/head for Winchester Disk

This switch is used to set the number of cylinders/head for the specified Winchester unit. The value of "n" is used only for Physical Block I/O operations. The controller handles the block number to cylinder/head/sector conversion for Logical Block I/O operations. The device specifier must reference an unmounted Winchester unit.

/OF=n -- Specify Block Offset for Winchester Disk

This switch is used to set a block offset to be applied by the FW driver prior to initiating Logical Block I/O operations. The device specifier must reference an unmounted Winchester unit.

This switch has the effect of reserving disk space and rendering it inaccessible to FILES-11 operations. Thus for most applications, this feature is not useful. However, it is possible to use the /OF switch in conjunction with the /UN switch to break one Winchester drive into two logical device units (see example later in this section).

The desirability of using one surface as two logical units is dependent on the target application. Note that once the Winchester is INItialized to a single logical unit per drive, it is not possible to change the offset and retain usability of the surface without corrupting the data on it.

/UN=n -- Specify Physical Unit Number of Winchester Disk

This switch sets the physical unit number of the referenced device. The value of n must be in the range (2-3). The device specifier must reference an unmounted Winchester unit.

Some Winchester drives do not have a unit select control line. It is therefore not possible to attach two such drives to a single controller. In other cases, the user has only one Winchester drive. If desired, this switch, in conjunction with the /OF switch, could be used to divide the surface of a single Winchester dirve into two separate logical devices.

The following command sequence can be used to make a Q2040 function as two equally-sized units:

```
>FWU FW2:/HD=8/ST=17/SL=512/OF=0/SIZE=34700/UN=2
>FWU FW3:/HD=8/ST=17/SL=512/OF=34700/SIZE=34700/UN=2
```

In this case, FW2: becomes the first (outermost) half of the surface and FW3: becomes the second (innermost) half of the surface. Care should be taken to insure that the addressing space does not overlap. If if does, the integrity of the file structure will not be insured.

The following table lists device information corresponding to the possible combinations of FWU Winchester disk switches.

Table IV-C

CODE	TYPE	/SIZE	/HD	/ST	/CY
1	ST406	10300	2	17	306
2	ST412	20600	4	17	306
3	ST419	30900	6	17	306
4	RO204	43400	8	17	320
5	RO206	65000	6	17	640
6	RO208	86800	8	17	640
10	SA1002	8650	2	17	256
11	SA1004	17200	4	17	256
12	Q2010	17200	2	17	512
13	Q2020	34700	4	17	512
14	Q2030	52000	6	17	512
15	Q2040	69400	8	17	512
16	Q2080	138800	7	17	1172
20	SA4008	51600	8	17	512

E. FWU GENERAL PURPOSE FUNCTIONS

Several general purpose functions are provided for use with the device-specific functions already described. The operation of these is detailed in this section.

/HE[LP] -- Print Help Message

This switch causes a command summary to be exhibited on the initiating terminal. The /HE[LP] command should be the only string entered. No device specifier is allowed. Any remaining command text on the entry line is ignored. Only the most commonly used functions are shown.

/SI[ZE]=n -- Specify Device Size

This switch causes the single precision value of "n" to be set in the device table as the size (in 512-byte blocks) of the referenced unit. This is allowed only for FILES-11 devices (which contain this information in a standard location). The size of each unit of a controller is independent. This switch was designed to be used primarily for setting the size of Winchester units. Use with other devices in the system is not advised.

/ON=n -- Set Device Status Offline/Online

This switch is used to set/reset the offline/online bit in the status of the referenced unit. The entry of "1" for n sets the bit to ONLINE; "0" for n sets the bit to OFFLINE.

The Executive disallows attempted accesses to units which have this bit set. The system startup task (SAV) initially fixes the offline/online status for all devices. Each CSR is examined and if not present, all units on the correspondng controller are set to offline. For some devices, each unit on an existing controller is individually tested by device-specific code. For the remainder, this switch is available.

For the FW driver, specifically, the "Device Not Ready" error code will probably be returned if an access is attempted on a unit which is physically not present unless the /ON=n switch used, in which case the Executive will return the "Device Offline" error code instead. We chose not to insert device-specific code into the SAV task (other than the bootstrap driver) so as not to alter DEC-supplied software.

/WP=n -- Set Device Software Write Protected/Enabled

This switch set/resets the write-protect status bit for the referenced device. The Executive disallows write acceses to a unit if the status is set to protected. The entry of a value of 0 for n sets the status to ENABLED. The entry of a value of 1 for n sets the status to PROTECTED.

This switch is useful whenever it is desired to use a volume in a read-only manner. A FILES-11 volume which is write protected (either by software or

hardware) may be MOUnted and DisMOunted and otherwise used as long as no write is attempted (Caution: INStall requires write access).

/NA[ME]=dd -- Specify Device Name

This switch is used to change the PHYSICAL name of the referenced device. Because of the table structure of RSX-11M, it is not possible to change the name of a specific unit. This function alters the name of ALL units on a controller. The value of "dd" must be two alphabetic characters.

No harmful effects are incurred to the operating system as the result of using this function since logical assignments, redirections, and the like are all done with address pointers (which the function does not change). However, a task which has a file open when the name is changed may suffer subsequent file system errors when a re-open, close, or delete is attempted.

This switch was provided to be used with some DEC-supplied tasks (notably FLX) which have built-in device tables to correlate device names with legal functions, device sizes, etc. For example, it is necessary to use the device name DX or DY when creating/accessing an RT-11 diskette using the FW driver with FLX. The device name DK may be used when reading a DEC diagnostic diskette (actually DOS-11 directoried) using the FW driver with FLX.

The user may determine that the device name FW should be permanently changed to DY. If this is the case, the system should be SAVed with the changed name. If, however, the driver is ever to be UNLoaded, the name must be changed back in order for UNLoad to function properly.

/DI -- Display Device Information

This switch causes a series of informational lines to be exhibited on the initiating terminal. The information displayed varies according to the device referenced. The most information is displayed for FW floppy units, the least for non-FW units. Basically, any parameter that can be affected by the FWU utility is displayed. All of the numeric output is in decimal. If some I/O operation is causing consistent failures, it is a good idea to use the /DI command as a diagnostic tool for examining the characteristics of the unit in question.

/VE -- Verifiy Readability of Entire Device Surface

This switch may be used on any device unit in the system which may be read (eg., not the line printer) to perform a block by block scan of the device. Unlike the VFY utility, this function reads all of the blocks, not just the ones that are used in the file structure. It may be applied to mounted or unmounted volumes. The referenced unit need not be a FILES-11 device, nor need it be a block structured device.

Once initiated, the verify routine will attempt to read the volume sequentially in (potentially) large data blocks. If a large block read failure occurs, a message will be printed indicating the block which the read started with. The function will then re-read the series of blocks one block at a time, reporting

each and every logical block in which an error occurs. If the small block reads are all successful, read failure may be interpreted to be a "soft" error (the function performs large block reads with retries inhibited).

If the /VE switch is used in conjunction with the /FO switch, the verify operation will proceed immediately following the completion of the formatting. Otherwise, the user will be asked to make the input volume ready before attempting to initiate the verify operation.

/CO -- Copy Device Surface

This switch invokes an interactive dialogue which can initiate the copying of (a) volume(s) to (an)other volume(s). The actual execution of the function is dependent on the relative size of the two device units. If the input device unit is large (with respect to the output device unit), the copy is assumed to be a volume backup. If the output device unit is large (with respect to the input device unit), the function is assumed to be a volume restoration. If the device unit sizes are equal, the operation is assumed to be a block-for-block copy operation. No file structure is involved, and neither of the device units may be MOUnted at the time the copy function is initiated. For tape units, the device size is assumed to be infinite, thus a disk-to-tape is a backup and a tape-to-disk is a restore.

This function does not perform any device-specific operations (such as rewind). The user is responsible for having input or output non-file structured volumes at the appropriate locations before a copy is started. If an unequal size condition exists, the function prompts for volume name and sequence numbers in a manner similar to PRESRV. There is an option (question answered by the user) to perform on-line verification of the copy. If answered affirmatively, the output volume must be block structured.

The actual copy is performed using large block I/O. If the online verification is used, each block is 4K bytes long. Otherwise, each block is 8K bytes long. If the operation performed is a restoration, the input volume name (created by the corresponding backup operation) must be known. If not know, the DMP utility may be used (device not mounted) to read the volume name which is in Block 0 of the relatively small volume (DMP TI:=dduu:/BL:0:0/AS command). The volume name should stand out, as it is the only ASCII information in the block.

Once the copy operation is started, errors are reported in a similar manner to that described in the /VE operation. Each long block is read. If not successful, the read is broken into small blocks and each small block is read with retry. The assembled long block is written to the ouput device. If the online verify is not in effect, the sequence repeats. If verify is in effect, the output data just written is read back. The re-read block is compared with the original block still in memory word for word. The sequence is then repeated.

If an error is encountered during any part of the sequence, the user may elect to retry, continue, or abort the copy operation. If the error occurred during a long block read, the retry attempts the long read again; the continue performs the small block re-read to assemble a long output block and prints specific block number error message lines. If the error occurred during a write operation, the retry attempts to write the long block again; the continue skips to the next long

input block. If the error occurred during a verify (read-back after write) the retry attempts both the write operation and the verify again; the continue skips to the next long input block. If the error occurs during a data compare, the hardware (memory, controller, or drive) is suspect unless the compare follows a write or read-back error. This error indicates that data read back from the output device does not match that saved in memory from the input device.

NOTE: The commmand interpreter built into FWU is table driven. It may be desirable for some users to limit the capabilities of the program by creating a version having only a subset of the provided switches (such as /IBM, /RX, /FO, /VE, /CO, and /DI). If so, the remaining switches may be elimnated by removing the table entries for them. The function dispatch table (FUNTBL) is located toward the beginning of the source.

F. FWU UTILITY EXAMPLES

The following examples illustrate FWU usage:

```
>!
>!      Example 1:   Help, Display, and Format
>!
>FWU                     !RUN INSTALLED VERSION OF FWU

FWU Disk Utility V1.0
For Help, Type /HE<CR>

FWU>/HE                 ;LIST HELPFUL INFORMATION
        ..
        ..      (Information is Displayed)
        ..
FWU>FW1:/IBM2/DI        ;SET DBLE DNSTY, 1 SIDE, 512-BYTE FMT
        ..
        ..      (Device Information is Displayed)
        ..
FWU>FW1:/FO/VE          ;FORMAT FW1: AS IBM DD, 512 BYTE

FW1:/Format -- Continue (Y/N): Y
** Begin Formatting **
** Begin Verification **

FWU>^Z          (Exit to MCR)

>RUN $FWU               !RUN UNINSTALLED COPY OF FWU

FWU Disk Utility V1.0
For Help, Type /HE<CR>

FWU>FW1:/IBM2/FO/VE     ;ONE-STEP FORMAT COMMAND ENTRY

FW1:/Format -- Continue (Y/N): Y
** Begin Formattig **
** Begin Verification **

FWU>^Z          (Exit to MCR)

>
>
>!      Initialize Files-11 Volume
>!
>BAD FW1:                !MARK BAD BLOCKS
>INI FW1:TEST            !INIT VOLUME
>MOU FW1:/OVR            !MOUNT FILE-11 VOLUME
>PIP FW1:/FR             !LIST STORAGE USAGE

FW1: HAS 1188. BLOCKS FREE, 28. USED OUT OF 1216. BLOCKS
```

```
>DMO FW1:                !DISMOUNT VOLUME
  ** FW1: DISMOUNT COMPLETE **
>!
>!      EXAMPLE 2:   Use FW Driver with FLX
>!
>FWU FW1:/RX02/NAME=DY          !RUN FWU FROM MCR LEVEL TO
>                               !SET UP FOR FLX RX02 USAGE
>FLX DY1:/ZE/RT                 !INITIALIZE RT-11 DIRECTORY
>FLX DY1:/RT=[11,10]*.MAC       !BACK UP MACRO SOURCE FILES
>FLX DY1:/LI/RT                 !LIST RT-11 DISK DIRECTORY
        ..
        ..      (FLX Prints Listing)
        ..
>FWU DY:/NAME=FW                !CHANGE NAME BACK TO FW:
>!
>!      Example 3:   Set up Winchester Disk Parameters
>!
>FWU FW2:/HD=8/ST=17/SL=2/OF=0/SIZE=34700    !SET FW2:
>                                           !WINCHESTER TO Q2040
>
```

G. FWU MESSAGES

This section contains a listing of FWU messages with explanations and possible action to be taken by the user.

FWU -- Missing Device Specifier

Operation requires device specifier (eg., "FW2:"). Re-enter the command.

FWU -- Command Syntax Error

Extra or invalid character sequence in command line. Re-enter the command.

FWU -- "sw" Is An Invalid Option

The option /sw is not a valid FWU switch. Re-enter the command.

FWU -- Option "sw" Has Invalid Argument

Invalid switch value. Re-enter the command.

FWU -- Invalid Option Combination

Two switches on the same command line conflict (example: /RX02 and /IBM0). Re-enter the command.

FWU -- Device dduu: Not In System

The device is not in the device tables. This error could also be printed if the symbol table used when the FWU task was built does not match the Executive in memory. Check the device table (MCR DEV command)>

FWU -- Device dduu: Already In System

An attempt was made to rename a device when that device was already present. Choose another menmonic and re-enter the command.

FWU -- dduu: is Attached

Certain operations may not be performed if the device is attached by some other user. The FWU check for attached device alleviates the problem of the program getting "stuck" while waiting for the other user to finish using the device.

FWU -- dduu: is Mounted

Certain operations may not be performed if a volume is mounted on the requested device. Dismount the volume and retry the operation.

FWU -- dduu: is Not an FW

The specified operation is valid only for an FW unit. Re-enter the command.

FWU -- dduu: is Not a FILES-11 Device

Certain operations are valid only for file structured units. Re-enter the command.

FWU -- dduu: is Not a Floppy

Certain operations are valid only for FW floppy disk units (example: /FO). Re-enter the command.

FWU -- dduu: is Not a Winchester

Certain operations are valid only for FW Winchester units (example: /ST). Re-enter the command.

FWU -- Directive Execute Error nnnnn

The command was accepted, but some error occurred while executing an RSX-11M directive. Consult the I/O Operations Reference Manual for the meaning of nnnnn. The value printed is the positive decimal equivalent of the low order byte of the error code. Correct the problem, then retry the command.

dduu:/Format -- Continue? (Y/N):

This message is printed before initiating a diskette formatting operation. If the desired diskette is loaded into the drive, respond "Y<CR>". Else enter "N<CR>" or just "<CR>" to cancel the format operation.

** Begin Formatting **

This is an informational message indicating that the FWU utility is initiating a format operation. No user action is required.

dduu/Verify -- Continue? (Y/N):

This message is printed before initiating a volume read check operation. If the desired volume is ready, respond "Y<CR>". Else enter "N<CR>" or just "<CR>" to cancel the verify operation.

** Begin Verification **

This is an informational message indicating that the FWU utility is starting a volume verification. No user action is required.

Verify Error Starting at LBN nnnnn

This message is printed whenever a block scan fails. The utility proceeds to rescan the blocks one at a time following this message. No user action is required.

Block Read Error at LBN nnnnn

This message is printed to identify the specific block(s) at Which a block scan failed. No user action is required.

dduu:/Copy to Output Device:

This message is printed before initiating a copy operation. Respond with the output device desired or else type ^Z to abort the copy.

Enter Volume Name (1-6 Alpha Characters):

This message is printed when a copy operation requires a volume name identifier. Respond with the desired name or else type ^Z to abort the copy.

Verify Copied Data Online? (Y/N):

This message is printed by the copy function to determine whether or not to read copied data back into memory. Respond accordingly or else type ^Z to abort the operation.

Mount Input Volume nnnnn and Type <CR>:

During a volume restoration, a new input volume is required. Load the indicated volume and then respond, or else type ^Z to abort the copy operation.

Volume Header Read Failure nnnnn
<CR> to Preceed, ^Z to Abort:

The volume header is not readable. Respond to the request by making the volume ready or aborting the operation.

Wrong Volume Name
<CR> to Preceed, ^Z to Abort:

The volume name entered does not match the actual name on the volume. Determine which is the problem, then retry or abort the operation.

Wrong Volume Number
<CR> to Proceed, ^Z to Abort:

The volume sequence is incorrect. Load the correct volume or abort the operation.

Warning -- Mismatched Number of Output Blocks
<CR> to Proceed, ^Z to Abort:

This message indicates that the size of the volume being restored disagrees with the size of the volume that was backed up. In general, this is an attempt to restore a backed-up volume to an incorrect device. Determine the problem and respond accordingly.

Type <CR> When Input and Output Devices are Ready:

This message is printed before initiating a block-for-block copy (no volume sequence) operation. Respond with a <CR> when both devices are ready or else type ^Z to abort the copy operation.

Mount Output Volume nnnnn and Type <CR>:

This message is printed when a new pseudo-tape ouput volume is required by a copy operation. Respond <CR> when the output volume is ready, or type ^Z to abort the copy operation.

Volume Lable Write Failure nnnnn
<CR> to Proceed, ^Z to Abort:

The ouput volume is not ready, write protected, or faulty. The value of nnnnn is the driver error code. Determine the problem and respond accordingly.

Input Device Error nnnnn
^Z to Abort, <CR> to Continue, R to Retry:

The input volume is faulty. Respond to the request based on the value of nnnnn which is the driver error code.

Output Device Error nnnnn
^Z to Abort, <CR> to Continue, R to Retry:

The output volume is faulty. Based on the error code, correct the problem, continue, or abort the copy operation.

Read Check Error nnnnn
^Z to Abort, <CR> to Continue, R to Retry:

An attempt to read back a block just written has failed. If retry is selected, the write will be re-attempted, followed by another read-back attempt. If continue is selected, the copy operation will proceed. Else the operation is aborted.

Data Verification Error

The data read back from the output volume does not agree with that saved in memory (just written). If this message is printed after a read check error, the output volume disagrees with the original volume at the point noted. If this message prints under any other circumstance, the computer's memory is faulty or the ouput device has indicated a successful read-back of invalid data (indicating a controller or drive failure).

V. FW: DRIVER FUNCTIONS

The information in this chapter supplements that provided in the RSX-11M I/O Drivers Reference Manual. The FW driver provides the same functions as other disk drivers, plus additional functions required for device-specific operations.

Throughout this document, the word "device" is used in the RSX-11M context, meaning a device-unit combination, such as "FW0", or "DY1". This is not to be confused with the PDP-11 Peripherals Manual interpretation of "device" as a controller with (an) attached peripheral unit(s).

A. THE GLUN$ (GET LUN INFORMATION) MACRO

Word 0 of the buffer returned by the GLUN$ macro contains the ASCII value of the physical device name (ie., "FW").

Word 1, low order byte contains the unit number of the associated device. This is RSX's interpretation of unit, not the device controller's. Thus if two controllers are present, each having four physical units, the unit number returned for physical unit 1 of the second controller will be 5.

Word 1, high order byte is a resident flag byte which will be returned as 200 octal if the driver is resident, or as 0 if the device is not loaded.

Word 2 is the device characteristics word with bit values as defined for all disks.

Word 3 low order byte contains the high 8 bits of the device size in 512-byte blocks. The high order byte is meaningless.

Word 4 contains the low 16 bits of the device size in 512-byte blocks.

Word 5 contains the logical block size in bytes, which is 512(10) for all disk devices.

NOTE: The I/O Drivers Reference Manual is INCORRECT in stating that Words 3 and 4 are undefined. In fact, these words contain the device size for any disk device, as noted in the Guide to Writing an I/O Driver.

B. STANDARD QIO FUNCTIONS

The following functions are valid for the FW device, as with all other disk devices:

QIO$	IO.ATT,...	Attach Device
QIO$	IO.DET,...	Detach Device
QIO$	IO.KIL,...	Kill I/O on Device
QIO$	IO.RLB,...,<stadd,size,,lbnh,lbnl>	Read Logical Block
QIO$	IO.RVB,...,<stadd,size,,lbnh,lbnl>	Read Virtual Block
QIO$	IO.LOV,...,<stadd,size,,lbnh,lbnl>	Load Overlay
QIO$	IO.WLB,...,<stadd,size,,lbnh,lbnl>	Write Logical Block
QIO$	IO.WVB,...,<stadd,size,,lbnh,lbnl>	Write Virtual Block

where: stadd is the starting address of the data buffer (must be on a word boundary).

size is the data buffer size in bytes (must be even and greater than zero).

lbnh is the high order block. For floppy units, this must be zero.

lbnl is the low order block. For floppy units, the range depends on the diskette format selected. For Winchester units, provision has been made to accommodate different device sizes, including those too large to express in 16 bits.

The octal values for the above functions are defined by the FILIO$ macro call.

The Executive processes the Attach and Detach functions without invoking the driver. These functions are allowed only for unmounted volumes.

The Kill I/O function causes all previously queued pending I/O requests (if any) for the referenced device to be cancelled. If an I/O transfer is in progress when the Kill I/O is executed, the Executive calls the driver to cancel the I/O in progress. Most disk drivers ignore this call since the pending operation will complete or time out in a short time.

The Executive (or the file system, if necessary) processes the Read/Write Virtual functions until the starting block number specified is transformed into a logical block number, at which time the driver is invoked with the corresponding Read/Write Logical function. This implies that a volume is mounted on the device, and that a file is open on the associated LUN.

This leaves only the Read/Write Logical functions to be processed by the driver, and then only after the parameters have passed a series of validity checks imposed by the Executive. The logical block I/O functions are valid only for unmounted volumes, or for mounted volumes when the calling task is privileged.

The Read Overlay function is a special case of the Read Logical function in which the Executive adds the starting block of the requesting task to the block number specified. This function is used by the overlay handler built into any task which is overlaid. The Executive bypasses the privilege checks performed by normal logical block requests and instead checks the block number to insure that the request is contained within the limits of the task image on disk.

NOTE: File structured I/O performed by typical utility tasks (GET$, PUT$, READ$, and WRITE$) is handled by File Control Service (FCS) routines linked with these tasks. FCS routines transform user calls into IO.RVB and IO.WVB QIO directives. The OPEN$, CLOSE$, and other control functions are also handled by FCS routines. In these cases, FCS transforms user calls into GLUN$, ALUN$, and QIO directives which are processed by the Executive in conjunction with the File Control Primitive task (F11ACP or other ACP) which performs any required I/O (such as directory block accessing) with the IO.RLB and IO.WLB QIO directives. This eliminates the need for the programmer to perform I/O at the QIO level. The driver itself needs only to be concerned with the IO.RLB and IO.WLB functions in order to support RSX I/O at the FCS, FCP, and QIO levels.

C. DEVICE-SPECIFIC QIO FUNCTIONS

The following device-specific functions are valid for the FW device:

QIO$	IO.RPB,...,<stadd,size,,,pblk>	Read Physical Block
QIO$	IO.WPB,...,<stadd,size,,,pblk>	Write Physical Block
QIO$	IO.WDD,...,<stadd,size,,,pblk>	Write Physical Block Setting the Deleted Data Bit
QIO$	IO.SPF,...,<fmt,hcst,ofsl,ofhdsh,dszl,cph>	Specify Device Format Parameters
QIO$	IO.SPB,...,<,,,ofv,strk,ntrk>	Format Tracks
QIO$	IO.STC,...	Set Media Density

where: pblk is the physical block to be accessed. The driver constructs the cylinder, head, and sector of the start of the data transfer according to the currently defined device parameters.

fmt is the format control word. Six bits of this word are used. They correspond exactly to the Unit Designator parameter fields DS, SL, and IL documented in the FWD0100/FWD1100 Disk Drive OEM Manual, Chapter III F. The other bits are masked out by the FW driver.

hcst is a word containing a value defining the number of sectors/track in the low order byte and the number of sectors/track in the high order byte. These values are used in processing the Physical Block I/O functions.

ofsh is a 16-bit value defining the low order of the block offset to be applied to read/write logical block operations. This is normally zero.

ofhdsh is a word containing a value defining the high order 8 bits of the device size in the low order byte and the high order 8 bits of the block offset in the high order byte.

dszl is a 16-bit value defining the low order of the number of 512-byte blocks on the device.

cph is a word defining the number of cylinders per head on the device.

ofv is a word defining the offset to be performed during a format operation and corresponds exactly to the Head Step and Switch Offsets parameter described in the FWDX100 OEM Manual.

strk is a word defining the starting track to format.

ntrk is a word defining the number of tracks to format.

All of the above functions are valid for unmounted volumes only. The particular I/O function code values are defined by the SPCIO$ macro call. IO.RPB, IO.WPB, IO.WDD, and IO.STC are used by DXDRV and DYDRV. IO.SPF and IO.SPB are actually magtape control function codes whose mnemonics were borrowed so that alterations to the RSXMAC.SML macro library would not be necessary to support the FW driver-specific functions. The octal values corresponding to these codes are subject to change from time to time, but present assignments are documented in the RSX-11 I/O Operations Reference Manual (in an Appendix).

The Read/Write Physical functions allow access at the sector level for floppy and Winchester units. No sector interleaving or track offsetting is performed by the driver or controller. If a physical block I/O request specifies a length greater than the sector size, consecutive physical sectors will be read/written. The controller updates the sector to access without any interleaving or track offset considerations.

The physical block number specified represents a value that is converted from an unsigned single precision number to a head number, cylinder number, and sector number by using the heads/cylinder and sectors/track parameters currently in effect for the selected drive. The driver does not actually know if these parameters correspond to the media loaded in the drive, but simply performs algebraic operations yielding a value in head, cylinder, and sector from the specified block (sector) number supplied by the user. The cylinder number is the most significant value in the calculation, the head number next, and the sector number the least significant. If there is only one head per cylinder on the referenced drive, the formula reduces to a calculation of track and sector.

For RX01 and RX02 formats, the mapping results in exactly the same track and sector as would be arrived at by the DX or DY (DEC-supplied) drivers, since there is only one head per cylinder for these formats. Application software using the Read/Write Physical I/O functions originally written for RX01 or RX02 floppies will perform in a transparent manner using the FW driver instead of the DX or DY driver.

The Specify Format function is provided in order for an application program to dynamically set the assumed format for physical block I/O operations on unmounted volumes. The FWU utility also uses this QIO. Caution should be observed when executing a Specify Format, as parameters passed to the driver which do not correspond to the actual format of the media loaded in the drive will cause subsequent I/O operations to either fail or else produce results other than those desired (eg., corruption of a volume).

The Format Track function is used to initialize the control and data fields of a floppy diskette. A diskette may be initialized to RX01, RX02 or any of a variety of IBM formats, according to the last Specify Format operation performed. It should be mentioned that this function is NOT equivalent to the Set Media Density function implemented for the DY driver. The latter simply fills data fields with either double density or single density data. By contrast, the Format Track operation of the FW driver is able to write new sector headers on the device, thus allowing a diskette to be reformatted among RX01, RX02, and IBM double

density formats. By using the Format Track capability of the FW driver, a diskette whose original sector header data has been corrupted by a faulty drive or other error condition may be restored to a usable condition (the RX01 and RX02 cannot reformat diskettes).

The Set Media Density function is provided for compatibility with the DY driver. This function returns a success condition when invoked. The purpose of this function for the DY driver is to allow diskettes to be initialized to double or single density (data fields only). Since the FWU utility provides the capability of formatting diskettes, and since a wider variety of diskette formats is allowed by the FW driver (thus rendering the Set Media Density function incomplete), the NO-OP interpretation was deemed to be the most useful one for our purposes.

D. QIO FUNCTION MODIFIERS

Any of the standard or device-specific QIO functions may be modified by the inhibit retry (IQ.X) and/or diagnostic (IQ.UMD) subcodes. The octal values for these subcodes are defined by the FILIO$ macro call. The prefix for these modifying subcodes is an abbreviation for "I/O Qualifier".

The IQ.X subcode indicates that the operation is to be performed without retry. That is, of an error is detected by the controller, no attempt will be made to repeat the operation.

The IQ.UMD subcode specifies that the operation is to be terminated by passing the controller registers to the calling task, and that no error analysis is to be performed by the driver. The driver always returns the IS.SUC completion code in the I/O status block after completing a transfer function. If the operation did not involve a transfer, no registers are passed. If diagnostics are not supported by the Executive, the specified operations will be performed in the normal manner, and the subcode will be ignored.

The following are examples of usage of transfer operations with subcode modifiers:

```
QIO$    IO.RPB!IQ.X,...,<BUFF,128.,,0,494.>          Read Physical
                                                     with no Retry

QIO$    IO.RLB!IQ.X!IQ.UMD,...,<BUF,1000,,0,1>       Diagnostic Read
                                                     with no Retry
```

E. I/O AND RELATED ERROR CONDITION PROCESSING

The information in this section was included in order to accurately describe the processing of an I/O request and related status return codes, since it is not available from any other source. The error codes reported by the FW driver were designed to be consistent with the standardized error codes used by other drivers. A complete listing of the values presently assigned to the symbolic codes itemized below is contained in the RSX I/O Operations Reference Manual (in an Appendix). This is the definitive source for function codes and error return codes.

As with any other directive (EMT), the QIO$ macro is processed by the directive dispatcher. If the directive dispatcher determines that the directive parameter block is valid, the directive is passed to the QIO directive processor. If the QIO directive processor determines that the information in the standard portion of the parameter block is valid for the related device, the success return (IS.SUC) is returned to the calling task, and processing continues. Otherwise, the operation is terminated.

The following status conditions may be returned to a task by the directive dispatcher or QIO directive processor in the Directive Status Word ($DSW):

IS.SUC	Directive accepted, request in progress
IE.ADP	Invalid I/O status block specified
IE.IEF	Invalid event flag number specified
IE.ILU	Invalid logical unit number specified
IE.SDP	Invalid data parameter block
IE.ULN	Unassigned logical unit number (LUN)
IE.HWR	Hardware driver not resident
IE.UPN	No pool space for I/O packet or AST

If the QIO$ directive was accepted thus far, an I/O packet is created to store the request parameters, and the specified event flag (if any) is cleared. The QIO directive processor then performs more checks on the I/O request. In certain cases, the request is completed by the QIO directive processor, so the driver is not called.

The following errors may be returned by the QIO directive processor in the I/O status block:

IS.SUC	Directive completed successfully (This code is returned immediately only if the Executive processes the request without calling the driver, such as when the function is IO.KIL.)
IS.PND	Operation is pending in the I/O queue (This is the normal return.)

IE.PRI	Privilege violation
IE.IFC	Invalid function for device
IE.OFL	Device offline
IE.SPC	Buffer space not within calling task limits or zero buffer length
IE.NOD	No pool space available
IE.BYT	Odd byte starting address or length
IE.OVR	Illegal overlay request block number
IE.AIN	File already accessed on LUN
IE.NLN	Un-assigned LUN
IE.BAD	Bad calling parameters

After the QIO directive processor returns one of the above values to the calling task, the calling task is eligible for execution. If the IS.PND code was returned in the I/O status block, the specified event flag (if any) remains clear. Otherwise, it is set.

Assuming that the IS.PND value was returned immediately, the request packet is now in the I/O queue. When the driver dequeues the request, further checks are performed by Executive routines before the driver processes the request. The following error codes may be returned in the I/O status block if the Executive completes the I/O request:

IS.SUC	Directive processed, function complete (This happens only if the request was IO.ATT or IO.DET.)
IE.PRI	Privilege violation
IE.DAA	Device already attached
IE.DNA	Device not attached
IE.IFC	Illegal function under present conditions
IE.ABO	Request aborted by user or Executive (This happens when a Kill I/O has been executed and the I/O request was still in the I/O queue.)

If the Executive checks were completed sucessfully and the operation is still pending, the request is either passed back to the driver or to the file system (FCP). If the function is passed to FCP, file system errors are possible. We will not enumerate these here, as they are documented elsewhere. If the driver regains control of the I/O request, driver processing begins.

The following error codes may be returned in the I/O status block the FW driver (note that other error codes are possible from other drivers; these are specific to the FW driver):

IS.SUC — Operation successfully completed. If the operation was a transfer function, the byte count is returned in the second I/O status word.

IS.WDD — Same as above, but IO.RPB operation resulted in the reading of at least one sector of data with the deleted data address mark.

IE.PRI — Privilege violation. The FW driver attempts to insure the integrity of mounted volumes by disallowing physical block reading and writing, formatting, or parameter setting if the device is mounted.

IE.IFC — Illegal function code. This happens if the actual function was legal but the sub-function bits were incorrect. As an example, the IO.RWD control functon has the same function code as IO.SPF but different sub-function bits. It must therefore be detected. Alternatively, the function is valid only for floppy disk units and the referenced drive was a Winchester.

IE.DNR — Device not ready. This error indicates that the controller detected a drive not ready condition or that an operation could not be completed in a reasonable amount of time. The driver makes different assumptions about what is reasonable based on whether the referenced unit was a floppy or a Winchester.

IE.FHE — Fatal hardware error. This error indicates that an illegal error status was returned by the controller upon completion of an operation or that the controller cannot communicate with the physical device unit.

IE.VER — Unrecoverable error. This is the general error code used by drivers to indicate media problems. Usually, this means that the equivalent of a parity error was detected by the controller. For floppy disk devices, the actual error could be unexpected media density or sector length, invalid sector header data, invalid checksum following the data portion of the sector, or unreadable media. For Winchester units, the flaw map may be unreadable or the media is unreadable.

IE.BLK — Bad block number. This error is detected by the controller if the logical block specified for the start of a transfer is invalid, or by the driver if an invalid physical sector number (larger than 16 bits) is specified for a physical block I/O request.

IE.BAD — Bad Parameters. This error is returned if the controller detects invalid calling parameters. This could be the result of specifying a media format inconsistent with the media loaded in the referenced drive during a physical block I/O request when the request results in the passing of an invalid head, cylinder, or sector number to the controller by the driver.

IE.OFL — Device offline. This error is returned if the controller determines that the referenced drive is physically not present.

IE.WLK — Drive writelocked. For floppy units, the media may be write protected by means of a notch cut at the back of the diskette. The controller returns this error if a diskette is loaded in the drive but the drive indicates that the notch was present (uncovered) when a write operation was attempted.

IE.EOV — End of volume. This error is returned by the controller if a transfer request results in media overrun (ie., the start of the transfer was accessible but the end overflowed the limits of the device.

NOTE: An I/O status block must be present in order for a task to determine if the I/O request completed successfully. If a wait-for (WTSE$) is issued, the task has a positive assurance that the directive was completed, but not that the request was completed successfully. Assuming that the WTSE$ directive was valid, a success value is ALWAYS returned for the wait-for in the directive status word ($DSW). This is NOT an indication that the I/O request is completed successfully; only that the wait-for condition has been satisfied. The task must examine the status block specified in the QIO directive to find out the result of a QIO operation.

Unfortunately, it is impossible for a program to tell whether an error was reported by the Executive or by the driver if the directive was processed to the point at which an I/O packet was created for the request because a task may not have an opportunity to execute instructions between the time that a directive is issued and its completion. Since some of the error codes reported by drivers are the same as some of the error codes reported by the Executive, this is a limitation of RSX-11M. By inspection of the error code list, one can easily see that all of the directive error codes are assigned. More accurate reporting of error conditions by a driver to a task is therefor not possible.

F. DRIVER ENTRY POINTS

Originally, RSX-11M was designed to be a real-time, priority-driven operating system suitable for use in process control (industrial) environments. Accordingly, drivers reported errors immediately so that a task could not get stuck while performing I/O if it executed a wait following the issuance of a QIO$ directive. Today's typical RSX-11M user, however, chooses the operating system (because of its multiuser capabilities) for use in a timesharing environment. Over the years, more and more timesharing features have been incorporated in the Executive as Sysgen options in order to better fit the needs and wishes of customers. Along with these, changes in the operation of drivers were required.

Following the original design goals of RSX-11M, drivers were constructed to have four independent entry points. These are:

Initiator — Called by the Executive to start an I/O operation.

Cancel I/O — Called by the Executive when a task is aborted or an I/O kill directive is issued by a task.

Power Fail — Called when the system is booted or after a power fail to initiate the driver to an idle state and complete formerly pending I/O. For terminals, this entry enables input interrupts.

Timeout — Called as an absolute protection mechanism in order to recover form the circumstance in which an I/O operation is initiated but does not complete within a reasonable amount of time.

In previous releases of RSX-11M, a disk I/O operation in progress when a power fail occured would always terminate with an error completion code following power fail recovery because the related drive would not be immediately ready upon power up. With the advent of Version 3, drivers were changed to special case I/O operations in progress after recovery from a power fail, resulting in the Patient Power Fail Recovery feature. To implement this feature, the power fail and timeout entry have become interrelated.

According to the new scheme, if an operation is pending when a power fail occurs, and if power fail recovery was included as a Sysgen option, the driver will retry the operation using the timeout entry point as an initiator until the drive has had enough time to spin back up to speed. This logic handles the case in which an operation is in progress, but not the case in which an operation is initiated soon after power fail recovery occurs (note that a task does not know whether or not a power fail has occurred unless it issues the SPRA$ directive).

To solve the general case for timesharing users, the FW driver has a provision for complete power fail recovery (P$$WFW) which is independent of the Sysgen definition of power fail recovery (P$$RFL). Since the startup program (SAV) calls drivers at the power fail entry point regardless of the definition of P$$RFL, and since the load MCR function (LOA) does the same, it is possible to incorporate this feature without actually defining the Sysgen power fail support.

The definition of P$$WFW causes the first operation after boot or load (or powerfail recovery if P$$RFL is also defined) on each FW drive to be special cased by the FW driver. If the drive-not-ready return code is desired instead of the possible long delay in completing the first operation on a drive, do not include P$$WFW when the driver is assembled. Otherwise, include this definition (see later section).

H. PROGRAMMING CONSIDERATIONS

The operation of the FW driver is very similar to that of most other RSX-11M file structured devices. However, a few points may be worth mentioning here.

1) Due to the relatively long time required to perform an operation on floppy disk units (up to 10 seconds for a 32K-word format command), the FW driver processes the Kill I/O command instead of ignoring it. This is done by means of the timeout entry point. An operation in progress when a task is aborted (Kill I/O performed by the Executive) or when the IO.KIL directive is issued by a task will complete within one second of the I/O Kill.

2) The FWD controller does not possess a Buffer Address hardware register. It is therefore not possible to report the actual length of and I/O transfer in the I/O status block. The driver therefore sets the transfer length to the request

length if the operation was successful, or to zero if the operation failed. Some part of the user buffer may have been filled or written even if the status block indicates that zero bytes were transferred. This does not affect the operation of DEC-supplied programs since they do not examine the reported transfer length on block oriented devices and they usually terminate upon encountering a hardware error condition.

3) The DEC RX01 format and the IBM single side single density diskette formats are the same, physically. However, the DEC RX01 format implies sector interleaving whereas the IBM format does not. Sector interleaving means that every other sector is skipped when a block is accessed (block equals (4) 128-byte sectors). This means that the actual transfer rate is effectively twice as slow with the DEC format as it would be with the IBM format. If a diskette is written with the IBM format, the performance on the user's system will be better than it would be using the DEC format, but the diskette will not be readable using an RX01 or RX02 drive with the DX or DY driver. Actually, the diskette could be read using DMP, but the data would be "scrambled". The same applies to use of RX02 formatted diskettes with interleaving disabled (FWU FW:/RX02/IL=0 command).

4) When using the Physical Block I/O functions with IBM double density formats, cylinder zero is a special case. Instead of having double density data with potentially 512-byte sectors, the first cylinder has a fixed format of 128-byte sectors. The driver does not take this into consideration when calculating the parameters to pass to the controller for sectors on this track. The result is that invalid parameters may be passed to the controller when accessing sectors on cylinder zero.

The design of the driver was such that it would be easy for a programmer to construct the physical block number to pass to the driver without his taking track zero into account either, according to the following formula:

```
pblk = (cylder*spc) + ((head-1)*spt) + (sector-1)
```

where:

pblk	is the starting physical block (sector) number
cylder	is the physical cylinder (starting with zero)
spc	is a constant defining the number of sectors per cylinder
head	is the physical head number (starting with one)
spt	is a constant defining the number of sectors per track
sector	is the sector number (starting with one)

If the number of heads per cylinder is one, each track is one cylinder (spt=spc), so the formula reduces to:

```
pblk = (track*spt) + (sector-1)
```

where: track is the track number (starting with zero)

Although the formula may look complicated, the effect is that the drive surface is considered to be consecutive sectors starting with sector zero and ending with the last logical sector on the device (in the case of IBM floppy units, there is a "hole" on track zero). It is thus possible for a program to calculate the address of the current operation by simple addition of the prior operation's starting sector and the sector offset between the prior operation and the current operation.